

FUJITSU Software

Interstage Application Server

A decorative horizontal band with a red-to-dark-red gradient. It features abstract, glowing white and red lines that swirl and intersect, creating a sense of motion and energy.

J2EE User's Guide

Windows/Solaris/Linux

B1WS-1096-03ENZ0(00)
April 2014

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:

- People who develop applications using J2EE components
- People who operate applications using J2EE components

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

The following table shows which features are provided in each edition:

| Product | Windows32/64 Solaris32 Linux32/64 Java Transaction Service (JTS) | Java Message Service (JMS) | J2EE Connector Architecture (Connector) | Enterprise JavaBeans (EJB) | HTTP Tunneling | Cluster Service |
|-----------------------|---|-------------------------------|--|----------------------------------|-------------------|--------------------|
| Enterprise Edition | Y | Y | Y | Y | Y | Y |
| Standard-J Edition | Y | Y | Y | Y | N | N |

Intended Readers

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

- Basic knowledge about Windows
- Basic knowledge about Java
- Basic knowledge about J2EE
- Basic knowledge about Web Service
- Basic knowledge about XML
- Basic knowledge about the Internet
- Basic knowledge about the relational database
- Basic knowledge about CORBA
- Basic knowledge about the transaction model (client/server model)
- Basic knowledge about UNIX **Solaris32/64**
- Basic knowledge about Linux **Linux32/64**

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)

Structure of This Manual

The structure of this manual is as follows:

Part 1 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 2 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.

[Chapter 8 Session Recovery](#)

This chapter describes session recovery in the Servlet service.

Part 3 EJB Edition

[Chapter 9 Basic Functions of the EJB Service](#)

This chapter explains the basic functions required to use the EJB Service

[Chapter 10 EJB Application Development](#)

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

[Chapter 11 How to Create Entity Beans](#)

This chapter explains how to create Entity Beans

[Chapter 12 How to Call EJB Applications](#)

This chapter explains how to call EJB applications

[Chapter 13 Customize by EJB Service Operation Command](#)

This chapter explains how to customize the EJB application

Part 4 Web Service Edition

[Chapter 14 Interstage Web Service Functions](#)

This chapter explains the functions of the Interstage Web Service.

[Chapter 15 Developing Web Services](#)

This chapter explains how to develop Web service applications and Web service client applications.

[Chapter 16 Interstage Web Service Operation](#)

This chapter explains how to operate Web service applications and Web service client applications.

Part 5 JTS/JTA Edition

[Chapter 17 Using Java Transaction API \(JTA\)](#)

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

Part 6 JMS Edition

[Chapter 18 Basic Functions of Interstage JMS](#)

This chapter describes the basic functions of Interstage JMS.

[Chapter 19 Environment Settings for Interstage JMS](#)

This chapter explains the environment settings to configure before using the Interstage JMS.

[Chapter 20 Developing a JMS Application](#)

This chapter explains how to develop JMS applications.

Part 7 Connector Edition

[Chapter 21 Basic Functions of the Interstage Connector](#)

The fundamental functions of the Connector are explained.

Part 8 Appendixes

[Appendix A JDK, JRE and FJVM](#)

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

[Appendix B Linkage with Oracle Real Application Clusters](#)

This appendix explains how to implement linkage with Oracle RAC.

[Appendix C Low-level Processing of SOAP Messages](#)

This appendix explains how Web service messages that use SOAP are processed.










[Appendix D Executing Sample Applications](#)

This appendix explains the sample applications that can be executed in Interstage Application Server.

Conventions

Representation of Platform-specific Information

In the manuals of this product, there are parts containing content that relates to all products that run on the supported platform. In this case, an icon indicating the product platform has been added to these parts if the content varies according to the product. For this reason, refer only to the information that applies to your situation.

| | |
|---|--|
|  | Indicates that this product (32-bit) is running on Windows. |
|  | Indicates that this product (64-bit) is running on Windows. |
|  | Indicates that this product (32/64-bit) is running on Windows. |
|  | Indicates that this product (32-bit) is running on Solaris. |
|  | Indicates that this product (64-bit) is running on Solaris. |
|  | Indicates that this product (32/64-bit) is running on Solaris. |
|  | Indicates that this product (32-bit) is running on Linux. |
|  | Indicates that this product (64-bit) is running on Linux. |
|  | Indicates that this product (32/64-bit) is running on Linux. |

Abbreviations

Read occurrences of the following Components as their corresponding Service.

| Service | Component |
|-------------------------------|---------------------|
| CORBA Service | ObjectDirector |
| Component Transaction Service | TransactionDirector |

Export Controls

Exportation/release of this document may require necessary procedures in accordance with the regulations of the Foreign Exchange and Foreign Trade Control Law of Japan and/or US export control laws.

Trademarks

Trademarks of other companies are used in this documentation only to identify particular products or systems.

| Product Trademarks/Registered Trademarks |
|--|
| Microsoft, Active Directory, ActiveX, Excel, Internet Explorer, MS-DOS, MSDN, Visual Basic, Visual C++, Visual Studio, Windows, Windows NT, Windows Server, Win32 are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. |
| Oracle and Java are registered trademarks of Oracle and/or its affiliates. |

Other company and product names in this documentation are trademarks or registered trademarks of their respective owners.

Copyrights

Copyright 2001-2014 FUJITSU LIMITED

| |
|---|
| April 2014 Third Edition November 2012 First Edition |
|---|

Contents

| | |
|--|----|
| Part 1 J2EE Common Edition..... | 1 |
| Chapter 1 Design of J2EE Application..... | 2 |
| 1.1 Environment Where J2EE Applications are Operated (JServer)..... | 3 |
| 1.1.1 What is JServer?..... | 3 |
| 1.1.2 JServer Types..... | 3 |
| 1.1.3 V8.0 Compatible Mode JServers..... | 4 |
| 1.1.4 JServer File Configuration..... | 4 |
| 1.1.5 Startup/Shutdown Execution Class..... | 6 |
| 1.1.6 Environment Settings for Access to the Shared Resources on the Network..... | 8 |
| 1.1.7 Control when Java Heap/Java Permanent Area is Insufficient..... | 9 |
| 1.2 Class Loader..... | 10 |
| 1.2.1 Structure of a Class Loader..... | 11 |
| 1.2.2 Separation of Class Loaders..... | 14 |
| 1.2.3 Changing the Search Order of Class Loaders..... | 17 |
| 1.2.4 Class Settings Used by JServer..... | 18 |
| 1.2.5 Settings of XML Parser..... | 22 |
| 1.2.6 Problem Investigation with the Trace Function..... | 24 |
| 1.2.7 Notes to be Taken when Class Loaders are Used..... | 26 |
| 1.3 Transaction Control..... | 26 |
| Chapter 2 Operating J2EE Applications..... | 29 |
| 2.1 Preparing J2EE Applications..... | 29 |
| 2.2 Creating an JServer..... | 30 |
| 2.3 Deploying and Setting J2EE Applications..... | 30 |
| 2.3.1 XML Parser Settings Required for Deployment..... | 32 |
| 2.3.2 J2EE Application (EAR File) Deployment Descriptor..... | 33 |
| 2.3.3 HotDeploy Function of J2EE..... | 35 |
| 2.3.4 Class Auto-reload Function..... | 41 |
| 2.3.5 Deploy the Web Application Anywhere on the Server..... | 45 |
| 2.3.6 Pre-configuring the Deployment Settings..... | 48 |
| 2.4 Preparation for Servlet Service Operation..... | 51 |
| 2.4.1 Interstage HTTP Server Environment Settings..... | 51 |
| 2.4.2 Microsoft® Internet Information Services 6.0 Environment Settings..... | 51 |
| 2.4.3 Microsoft® Internet Information Services 7.0/7.5/8.0 Environment Settings..... | 55 |
| 2.4.4 Sun Java System Web Server Environment Settings..... | 59 |
| 2.4.5 Procedure for Operation by Separating JServer and Web Server..... | 62 |
| 2.5 Request Distribution Control by Web Server Connector..... | 68 |
| 2.5.1 Distributing Procedure and Viewing the Status Using the Commands..... | 68 |
| 2.5.2 Monitoring Web Server Connector Faults..... | 70 |
| 2.6 Procedure for Using JTS..... | 78 |
| 2.6.1 Flow to Operation Start..... | 78 |
| 2.6.2 Flow to Operation End..... | 80 |
| 2.7 Procedure for Using JMS..... | 81 |
| 2.7.1 Flow to Operation Start..... | 81 |
| 2.7.2 Flow to Operation End..... | 81 |
| 2.7.3 Monitoring the Operational Status of an Event Channel..... | 82 |
| 2.8 Procedure for Using JavaMail..... | 84 |
| 2.8.1 Mail Sending Application..... | 84 |
| 2.8.2 Mail Receiving Application..... | 86 |
| 2.9 Customizing and Checking the Operating Environment..... | 87 |
| 2.9.1 Customizing the Operating Environment..... | 87 |
| 2.9.2 Checking the Operating Environment..... | 88 |
| 2.10 Debugging Applications..... | 91 |
| 2.10.1 Debugging Using Snap..... | 92 |

| | |
|---|------------|
| 2.10.1.1 Method Information of EJB Application Invoked by a Client..... | 95 |
| 2.10.1.2 EJB Application Method Information..... | 97 |
| 2.10.1.3 javax.transaction.UserTransaction API Information..... | 100 |
| 2.10.1.4 Database Manipulation Statement Information..... | 102 |
| 2.10.1.5 EJB Container Transaction Control Information..... | 104 |
| 2.10.1.6 J2EE Application User Debug Information..... | 106 |
| 2.10.1.7 Log Output Method for Support..... | 109 |
| 2.10.2 Using Application Debugging Information..... | 111 |
| 2.10.3 Using the Debugger..... | 111 |
| 2.10.4 Automatic Thread Dump Collection..... | 112 |
| 2.10.5 Debugging Using Java Method Trace..... | 112 |
| Chapter 3 JNDI..... | 113 |
| 3.1 JNDI Service Provider Environment Setup..... | 114 |
| 3.1.1 J2EE Application Client..... | 114 |
| 3.1.2 Applet..... | 118 |
| 3.2 Environment Setup for Referencing EJB..... | 119 |
| 3.2.1 Environment Setup in Client Environment..... | 120 |
| 3.3 Environment Setup when JDBC (Database) is Referenced..... | 122 |
| 3.3.1 Environment Setup when Symfoware is Used..... | 123 |
| 3.3.2 Using Symfoware Connection Pooling..... | 127 |
| 3.3.3 Environment Setup when Oracle is Used..... | 129 |
| 3.3.4 Environment Setup when SQL Server is Used..... | 132 |
| 3.3.5 Environment Setup when Generically-defined Data Sources are Used..... | 135 |
| 3.4 JDBC (Database) Connections..... | 139 |
| 3.4.1 Connection Pooling..... | 139 |
| 3.4.2 Automatic Reconnection Function..... | 140 |
| 3.4.3 Supported APIs..... | 141 |
| 3.5 Environment Setup when JMS is Referenced..... | 142 |
| 3.6 Environment Setup when JavaMail is Referenced..... | 142 |
| 3.7 Environment Setup when URL is Referenced..... | 142 |
| 3.8 Environment Setup when Connector is Referenced..... | 143 |
| 3.9 Description in Deployment Descriptor File..... | 144 |
| 3.10 Referencing Objects..... | 151 |
| 3.11 Name Conversion Function..... | 153 |
| 3.11.1 Name Conversion File..... | 154 |
| 3.11.2 interstage.xml File..... | 156 |
| 3.12 Transaction Function Using the UserTransaction Interface..... | 160 |
| 3.13 J2EE Application Client Deployment Descriptor File Detailed Setup..... | 162 |
| Chapter 4 The J2EE Application Security Function..... | 165 |
| 4.1 The Security Function..... | 165 |
| 4.1.1 User Authentication..... | 165 |
| 4.1.2 Access Constraints..... | 166 |
| 4.1.3 Method Permissions..... | 167 |
| 4.1.4 Security Methods..... | 167 |
| 4.1.5 Resource-connectable User Control Function..... | 167 |
| 4.1.6 Run-as Security Function..... | 168 |
| 4.2 Embedding the Security Function..... | 170 |
| 4.2.1 Setting Up the Security Management Environment Definition Files..... | 170 |
| 4.2.2 User and Security Role Settings..... | 171 |
| 4.2.3 Directory Service Work Procedure..... | 172 |
| 4.2.4 Setting the Security Function into the J2EE Application Client..... | 174 |
| 4.2.5 Setting the Security Function into a Web Application..... | 175 |
| 4.2.6 Setting the Security Function into the EJB Application..... | 177 |
| 4.3 Collecting the Authentication Log of the Security Function..... | 177 |
| 4.4 Action when a Security Function Error Occurs..... | 178 |

| | |
|---|-----|
| Part 2 Servlet/JSP Edition..... | 180 |
| Chapter 5 Functions of the Servlet Service..... | 181 |
| 5.1 Session Recovery Function..... | 181 |
| Chapter 6 Web Application Development..... | 182 |
| 6.1 Configuring the Web Application Directory..... | 182 |
| 6.2 Notes on the Development of Web Applications..... | 182 |
| 6.2.1 Notes when Using Cookies..... | 182 |
| 6.2.2 Cross-site-scripting Fragility Problem..... | 182 |
| 6.2.3 Errors and Exceptions..... | 183 |
| 6.2.4 Specifying an Error Page for the HTTP Error Status Code..... | 183 |
| 6.3 Web Application Environment Definition File (Deployment Descriptor)..... | 184 |
| 6.3.1 Start and End of Web Application Environment Definition Files..... | 189 |
| 6.3.2 The Name of a Servlet Context..... | 189 |
| 6.3.3 Servlet Context Initialization Parameters..... | 189 |
| 6.3.4 Filter Class..... | 190 |
| 6.3.5 Filter Class Application Target..... | 191 |
| 6.3.6 Listener Class..... | 194 |
| 6.3.7 Servlet Attributes..... | 195 |
| 6.3.8 Servlet Mapping..... | 197 |
| 6.3.9 Session Parameter..... | 199 |
| 6.3.10 Mime Types..... | 199 |
| 6.3.11 Welcome Files..... | 204 |
| 6.3.12 Resources During Error Occurrence..... | 205 |
| 6.3.13 Access Limit..... | 206 |
| 6.3.14 User Authentication..... | 207 |
| 6.3.15 Security Role..... | 210 |
| 6.3.16 Mapping of the Locales and the Character Code..... | 210 |
| 6.3.17 Definitions Shared by JSPs in Web Application..... | 211 |
| Chapter 7 How to Call Web Applications..... | 214 |
| 7.1 Calling Servlets..... | 214 |
| 7.1.1 Call that Requires Mapping..... | 214 |
| 7.1.2 Call That Does Not Require Mapping..... | 215 |
| 7.2 Calling JSPs..... | 216 |
| 7.3 Calling HTML, Image and Other Files..... | 217 |
| Chapter 8 Session Recovery..... | 219 |
| 8.1 About Session Recovery..... | 219 |
| 8.1.1 Session Backup..... | 220 |
| 8.1.2 Session Recovery..... | 222 |
| 8.1.3 Enabling and Disabling Session Recovery in the URI..... | 223 |
| 8.1.4 Monitoring Session Recovery..... | 225 |
| 8.1.5 Web Server Connector Fault Monitoring..... | 225 |
| 8.1.6 Maximum Number of Sessions Maintained in the Session Registry Server..... | 226 |
| 8.1.7 Session Serialization..... | 226 |
| 8.1.8 Clearing All Sessions..... | 227 |
| 8.1.9 Session Recovery Log..... | 227 |
| 8.1.10 Destroying Expired (timed out) Sessions Maintained in the Session Registry Server..... | 228 |
| 8.1.11 Session IDs..... | 228 |
| 8.1.12 Session Registry Server Access Restrictions..... | 228 |
| 8.1.13 Specifying the Servlet Container Control Port..... | 229 |
| 8.2 Session Recovery Scope..... | 230 |
| 8.3 Session Registry Server Settings..... | 231 |
| 8.3.1 Creating Session Registry Server WorkUnits (Using the Interstage Management Console)..... | 232 |
| 8.3.2 Creating Session Registry Server WorkUnits (Using the isj2eeadmin Command)..... | 234 |
| 8.3.3 Deploying Session Registry Server (Using the Interstage Management Console)..... | 238 |

| | |
|--|------------|
| 8.3.4 Deploying Session Registry Server (Using the ijsdeployment Command) | 238 |
| 8.3.5 Session Registry Server Environment Definition File Settings | 238 |
| 8.4 Session Registry Client Settings | 241 |
| 8.5 Session Recovery Settings | 243 |
| 8.5.1 Settings for each Timeout | 243 |
| 8.5.2 Concurrency (Number of Simultaneous Processes) Settings | 245 |
| 8.5.3 Example of Setting of IP Address and Port Number | 246 |
| 8.6 Session Recovery Application Method | 248 |
| 8.6.1 Session Registry Server Operations and Information Reference | 248 |
| 8.6.2 Changing the Session Registry Server Start User | 249 |
| 8.6.3 Isolating the Server | 249 |
| 8.6.4 Running More than One Session Registry Server Application | 249 |
| 8.6.5 Restarting Session Registry Server | 250 |
| 8.6.6 Backing Up/Restoring Session Registry Server Resources | 250 |
| 8.7 Application Creation Method | 250 |
| Part 3 EJB Edition | 254 |
| Chapter 9 Basic Functions of the EJB Service | 255 |
| 9.1 Session Bean | 255 |
| 9.1.1 STATELESS Session Bean Web Service | 255 |
| 9.2 Entity Bean | 255 |
| 9.2.1 Managing Entity Bean Instances | 255 |
| 9.2.2 Entity Bean Optimization | 256 |
| 9.2.3 Increasing the Speed of CMP2.0 Multi-Item Searches | 257 |
| 9.2.4 Correspondence of Entity Bean and a Database | 260 |
| 9.2.5 EJB QL | 261 |
| 9.3 Message-driven Bean | 261 |
| 9.3.1 Durable Subscription Function | 262 |
| 9.3.2 Message Backup Function in Abnormal Circumstances | 262 |
| 9.4 Time Monitoring Functions Supported by EJB Service | 264 |
| 9.4.1 Maximum Time Monitoring Function for Application Processing | 266 |
| 9.4.2 Waiting Time Monitoring Function for Server Return | 267 |
| 9.4.3 Idle-time Monitoring Function of STATEFUL Session Bean | 267 |
| 9.4.4 Timer Deletion of EJB Object | 267 |
| 9.5 EJB Timer Service | 268 |
| 9.5.1 EJB Timer Service Access Method | 270 |
| 9.5.2 The Monitoring Start Method | 270 |
| 9.5.3 The Time Monitoring Processing Execution Method | 271 |
| 9.5.4 The Timer Cancellation/Status Reference Method | 271 |
| 9.5.5 Other | 272 |
| 9.6 Notes in EJB Service | 272 |
| Chapter 10 EJB Application Development | 273 |
| 10.1 Application Development Flow | 273 |
| 10.2 Developing an EJB Application | 273 |
| 10.3 Deployment of an EJB Application | 273 |
| 10.4 Debugging an EJB Application | 273 |
| 10.5 Using the Development Environment of Other Companies | 274 |
| Chapter 11 How to Create Entity Beans | 275 |
| 11.1 CMP Definitions | 275 |
| 11.2 Notes on Instance Management Modes | 275 |
| 11.3 Correspondence between Data Types Defined in a CMP, and DBMS SQL Data Types | 276 |
| 11.3.1 Standard Data Types | 276 |
| 11.3.2 Other Classes | 279 |
| Chapter 12 How to Call EJB Applications | 281 |

| | |
|---|------------|
| 12.1 Calling Session Beans..... | 281 |
| 12.2 Calling Entity Beans..... | 281 |
| 12.2.1 Specifying Search Processing..... | 281 |
| 12.3 Relationship between Enterprise Bean Instance, EJB Object, and EJB Home..... | 283 |
| 12.4 Using Java Applets..... | 284 |
| 12.4.1 Development Procedure (Pre-installed Version Java Library)..... | 285 |
| 12.4.2 Client Setup (Pre-installed Version Java Library)..... | 287 |
| 12.4.3 Development Procedure (Portable-ORB)..... | 288 |
| 12.4.4 Setting Client Environment (Portable-ORB)..... | 293 |
| 12.5 Digital Signature in Applets..... | 297 |
| 12.5.1 Digital Signature..... | 297 |
| 12.6 Notes..... | 300 |
| Chapter 13 Customize by EJB Service Operation Command..... | 301 |
| 13.1 Customize Flow..... | 301 |
| 13.2 Export and Import of Enterprise Bean Definition Information..... | 301 |
| 13.3 Contents of Enterprise Bean Definition File..... | 302 |
| 13.4 Enterprise Bean Definition File Example..... | 318 |
| Part 4 Web Service Edition..... | 321 |
| Chapter 14 Interstage Web Service Functions..... | 322 |
| 14.1 Web Service Standards..... | 322 |
| 14.2 Interstage Web Service Basic Functions..... | 323 |
| 14.3 Execution Environment for Web Services..... | 324 |
| 14.3.1 Execution Environment for Web Service Applications..... | 324 |
| 14.3.2 Execution Environment for Web Service Clients..... | 324 |
| Chapter 15 Developing Web Services..... | 326 |
| 15.1 Developing Web Services (Server Function)..... | 326 |
| 15.1.1 WAR/ejb-jar File Configuration for Web Service Applications..... | 327 |
| 15.1.2 Developing Web Service Applications..... | 327 |
| 15.1.3 Editing the Deployment Descriptors..... | 330 |
| 15.1.4 Packaging the Files into a WAR File or ejb-jar File/EAR File..... | 331 |
| 15.1.5 Settings Relating to HTTP Connections..... | 331 |
| 15.1.6 Providing Interface Information for the Web Service..... | 332 |
| 15.2 Developing Applications that Call Web Services (Client Function)..... | 332 |
| 15.2.1 Obtaining Interface Information for the Web Service..... | 333 |
| 15.2.2 Generating a Stub..... | 333 |
| 15.2.3 Developing the Web Service Client Application..... | 334 |
| 15.2.4 Editing the Deployment Descriptor..... | 336 |
| 15.2.5 Settings Relating to HTTP Connections..... | 336 |
| 15.3 Mapping of XML and Java Data Types..... | 338 |
| 15.3.1 Simple Types..... | 338 |
| 15.3.2 Structure Type/ Bean Type..... | 340 |
| 15.3.3 Array Type..... | 342 |
| 15.3.4 Attachment File Type..... | 343 |
| 15.3.5 Using Data Types as out/inout Parameters..... | 345 |
| 15.4 WSDL that Can be Used..... | 347 |
| 15.5 Developing Applications Based on the WS-I Basic Profile and Attachments Profile..... | 348 |
| 15.6 Web Service Environment Definition Files (Deployment Descriptors)..... | 349 |
| 15.6.1 Format for webservices.xml..... | 350 |
| 15.6.2 webservices.xml Tag..... | 350 |
| 15.6.3 web.xml..... | 353 |
| 15.6.4 Service Reference Description..... | 353 |
| 15.7 Storage Location for the Sample Application..... | 355 |
| Chapter 16 Interstage Web Service Operation..... | 356 |

| | |
|--|------------|
| 16.1 How to Operate Web Services (the Server Function)..... | 356 |
| 16.2 How to Operate Web Services (the Client Function)..... | 358 |
| 16.2.1 Client Function Logs..... | 360 |
| 16.2.2 Stub Settings File..... | 361 |
| 16.2.3 Connection via Proxy Servers..... | 362 |
| 16.3 Web Service Settings File..... | 362 |
| 16.3.1 Web Service Client Log File Path..... | 364 |
| 16.3.2 Maximum Size for Web Service Client Log Files..... | 365 |
| 16.3.3 Maximum Number of Generations for Web Service Client Log Files..... | 365 |
| 16.3.4 SSL Definitions Used by Web Service Clients..... | 365 |
| 16.3.5 Location of Temporary Files Created for Attachment Files..... | 366 |
| 16.3.6 Maximum Size for Processing Using Memory Only (not Creating Temporary Files) when Attachment Files are Received..... | 367 |
| 16.3.7 Deletion of Attachment File Data Received by the Web Service Application at the Time of Returning a Response (Releasing Resources)..... | 367 |
| 16.3.8 Default Character Code of Attachment Files Specified in 'text/plain' in WSDL..... | 368 |
| 16.3.9 Non-ASCII Character Sending Format..... | 368 |
| 16.3.10 Enabling Authentication Using the Directory Service..... | 369 |
| 16.3.11 Web Service Client Connection Timeout..... | 369 |
| Part 5 JTS/JTA Edition | 371 |
| Chapter 17 Using Java Transaction API (JTA) | 372 |
| 17.1 JTA | 372 |
| 17.2 User Transaction Interface | 372 |
| 17.2.1 Functions Provided by the User Transaction Interface | 372 |
| 17.2.2 Environment Setup for the User Transaction Interface | 373 |
| 17.2.3 Acquiring the User Transaction Interface | 374 |
| 17.3 Generating a JTA Application | 374 |
| 17.3.1 Application Configuration | 374 |
| 17.3.2 Performing Initialization Process and Acquiring the UserTransaction Object | 375 |
| 17.3.3 From Transaction Start to Transaction Stop | 375 |
| 17.3.4 JTA Application Example | 376 |
| 17.3.5 Precautions | 377 |
| Part 6 JMS Edition..... | 379 |
| Chapter 18 Basic Functions of Interstage JMS..... | 380 |
| 18.1 Publish/Subscribe Messaging Model (1-to-n Messaging Model)..... | 380 |
| 18.2 Point-To-Point Messaging Model (1-to-1 Messaging Model)..... | 381 |
| 18.3 Message Guarantee..... | 382 |
| 18.4 Message Selector Function..... | 383 |
| 18.5 Queue Browser Function..... | 384 |
| Chapter 19 Environment Settings for Interstage JMS..... | 386 |
| 19.1 Environment Settings for the Event Channel Operation Machine before Operation..... | 386 |
| 19.1.1 Starting Interstage..... | 386 |
| 19.1.2 Creating and Starting a Unit..... | 386 |
| 19.1.3 Creating a Static Event Channel..... | 387 |
| 19.1.4 Changing the Event Channel Operating Environment..... | 389 |
| 19.2 Environment Deletion for the Event Channel Operation Machine after Operation..... | 390 |
| 19.2.1 Deleting the Static Event Channel..... | 391 |
| 19.2.2 Stopping and Deleting a Unit..... | 391 |
| 19.2.3 Stopping Interstage..... | 391 |
| 19.3 Environment Settings for the JMS Application Operation Machine before Operation..... | 391 |
| 19.3.1 Setting JNDI Environment Definitions..... | 393 |
| 19.3.2 Registering ConnectionFactory Definition..... | 394 |
| 19.3.3 Registering Destination Definition..... | 394 |
| 19.4 Environment Deletion for the JMS Application Operation Machine after Operation..... | 396 |

| | |
|--|------------|
| 19.4.1 Deleting ConnectionFactory Definition..... | 396 |
| 19.4.2 Deleting Destination Definition..... | 396 |
| 19.4.3 Deleting Durable Subscriber..... | 396 |
| Chapter 20 Developing a JMS Application..... | 398 |
| 20.1 Designing an Application..... | 398 |
| 20.2 Creating a JMS Application..... | 398 |
| 20.2.1 Publish/Subscribe Messaging Model and Point-To-Point Messaging Model..... | 398 |
| 20.2.2 Create a Message Listener..... | 401 |
| 20.2.3 Durable Subscription Function..... | 402 |
| 20.2.4 Message Priority and Lifetime..... | 404 |
| 20.2.5 Message Persistent Function..... | 404 |
| 20.2.6 Local Transaction..... | 404 |
| 20.2.7 Global Transaction | 406 |
| 20.2.8 Linkage with a CORBA Application..... | 409 |
| 20.2.9 Message Selector Function..... | 409 |
| 20.2.10 Queue Browser Function..... | 412 |
| 20.2.11 Notes on Executing an Application..... | 414 |
| 20.3 Interface..... | 415 |
| 20.3.1 API List of the Package javax.jms (Part 1)..... | 415 |
| 20.3.2 API List of the Package javax.jms (Part 2)..... | 416 |
| 20.3.3 API List of the Package javax.jms (Part 3)..... | 418 |
| 20.3.4 API List of the Package javax.jms (Part 4)..... | 419 |
| 20.3.5 API List of the Package javax.jms (Part 5)..... | 420 |
| 20.3.6 API List of the Package javax.jms (Part 6)..... | 421 |
| 20.3.7 API List of the Package javax.jms (Part 7)..... | 423 |
| 20.3.8 API List of the Package javax.jms (Part 8)..... | 424 |
| Part 7 Connector Edition..... | 426 |
| Chapter 21 Basic Functions of the Interstage Connector..... | 427 |
| 21.1 Resource Adapter Types..... | 427 |
| 21.2 Stopping/Starting the Resource Adapter..... | 428 |
| 21.3 Connection Management..... | 429 |
| 21.4 Transaction Management..... | 429 |
| 21.5 Security Management..... | 430 |
| 21.6 Work Management..... | 430 |
| Part 8 Appendixes..... | 432 |
| Appendix A JDK, JRE and FJVM..... | 433 |
| Appendix B Linkage with Oracle Real Application Clusters..... | 434 |
| B.1 Oracle Settings..... | 434 |
| B.2 Interstage Settings..... | 436 |
| B.3 Notes on Linkage with Oracle RAC..... | 439 |
| Appendix C Low-level Processing of SOAP Messages..... | 440 |
| C.1 Structure of SOAP Message..... | 440 |
| C.2 Creating a SOAP Message..... | 441 |
| C.3 SOAP Envelope Processing..... | 441 |
| C.4 Fault Processing..... | 442 |
| C.4.1 Analyzing Fault Information..... | 443 |
| Appendix D Executing Sample Applications..... | 446 |
| D.1 The Sample Environment..... | 446 |
| D.1.1 Overview of the Sample Environment and Explanation of Functions..... | 446 |
| D.1.2 Advance Preparation..... | 446 |
| D.1.3 Starting the Sample Environment..... | 447 |

Part 1 J2EE Common Edition

| | |
|---|-----|
| Chapter 1 Design of J2EE Application..... | 2 |
| Chapter 2 Operating J2EE Applications..... | 29 |
| Chapter 3 JNDI..... | 113 |
| Chapter 4 The J2EE Application Security Function..... | 165 |

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 supplied with Interstage provides various functions required to implement enterprise-class multilayered services.

Functions to be Provided as J2EE Components

The Interstage J2EE components provide services for J2EE applications created according to the following specifications:

| Specification name | Specification version |
|--------------------|-----------------------|
| JSP | 2.0 |
| Servlet | 2.4 |
| EJB | 2.1 |
| JMS | 1.1 |
| JTA | 1.0 |
| JavaMail | 1.4 |
| JAF | 1.1 |
| JAXP | 1.2 |
| Connector | 1.5 |
| Web Services | 1.1 |
| JAX-RPC | 1.1 |
| SAAJ | 1.2 |

Interstage provides the following services.

- Servlet service

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

- EJB service

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

- Interstage Web Service

The Interstage Web Service is a component for controlling the execution of Web services.

- 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 the "JNDI" chapter.

- Java Transaction Service (JTS) Windows32/64 Solaris32 Linux32/64

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. SMTP, IMAP, and POP3 are also included as providers.

1.1 Environment Where J2EE Applications are Operated (IJSERVER)

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

1.1.1 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 or isj2eeadmin command.

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

Refer to the Reference Manual (Command Edition) for details of the isj2eeadmin command.

1.1.2 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 Java VM. This way, EJB can be invoked quickly from Servlet/JSP, and the memory resources can be saved since applications operate on the same Java VM.

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 Java VMs. Although memory resources are used up by operating the applications on separate Java VMs, a process concurrency can be set for each Java VM, and a process failure risk can be distributed.

Contains Web Applications only

The Java VM can be used with Web applications only.

Contains EJB Applications only

The Java VM can be used with EJB applications only.



Note

- When Web and EJB applications are operated on the same Java VM, EJB applications cannot be invoked from another IJSERVER application (or EJB client application).
- When Web and EJB applications are operated on the same Java VM, 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 Java VMs, only the EJB Java VM is started.

- When only the Web applications are deployed on the IJServer where both Web and EJB applications are operated on separate Java VMs, only the Web Java VM 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 Java VMs.

1.1.3 V8.0 Compatible Mode IJServers

In V9, a compatible Tomcat4.1 based Servlet Service, JDK/JRE1.4 and a V8.0 compatible mode IJServer were provided.

In V10 and later, the V8.0 compatible mode IJServer cannot be created. For this reason, migrate to the IJServer provided in this version.

1.1.4 IJServer File Configuration

IJServers created and applications deployed by the Interstage Management Console or isj2eeadmin command are managed as follows.

Windows32/64

J2EE common directory\ijserver\

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

Solaris32/64 Linux32/64

J2EE common directory/ijserver/

(The default J2EE common directory is /opt/FJSVj2ee/var/deployment.)

| <- Upper directory lower directory -> | | Meaning |
|---------------------------------------|-----------------------|--|
| IJServer name | apps | Files of deployed applications are deployed in the module name directory. |
| | module name | When the EAR file is deployed, the WAR/ejb-jar/RAR file in the EAR file is deployed. |
| | | Shared/lib |
| | Shared/classes | The jar files under the Shared/lib directory and class files under the Shared/classes directory are shared by applications in EAR. |
| | | These are loaded even when they are created after deployment. |
| | current | In the default setting, the IJServer current directory is created under the current directory For details, refer to the "IJServer Current Directory". |
| | distribute | module name |
| conf | | This contains the IJServer internal resource. |
| log | process serial number | This is the IJServer log directory. The IJServer has process serial number directories equivalent to the process concurrency number. Log files are output to these directories. The location of the log directory can be changed as follows: <ul style="list-style-type: none"> - Click on Interstage management console [WorkUnit] > [WorkUnit name] > [Environment Settings] > [Detailed Settings] > [WorkUnit Settings] > [Log Output Directory] - isj2eeadmin command |

| <- Upper directory lower directory -> | | | Meaning |
|---------------------------------------|---------|--|--|
| | | | For details, refer to the Reference Manual (Command Edition). The log file can be referenced by clicking [WorkUnit] > [WorkUnit name] > [Log Reference] |
| Shared | classes | | Refer to "1.2.1 Structure of a Class Loader" and "1.2.4 Class Settings Used by IJServer" in "1.2 Class Loader". |
| | lib | | Jar files under the Shared/lib directory cannot be overwritten or deleted while the IJServer is running. |
| work | | | The following directory is generated, and the container temporary files (such as java source generated from JSP, and compile results) are stored in it. Windows32/64 [Process serial number]\Catalina\<localhost>\Web application name Solaris32/64 Linux32/64 [Process serial number]/Catalina/<localhost>/Web application name |
| ext | | | Refer to "1.2.1 Structure of a Class Loader" and "1.2.4 Class Settings Used by IJServer" in "1.2 Class Loader". |



Note

For the application resources authority, refer to "Deploying and Setting J2EE Applications" and "Application File Protection Level" in the "Operating J2EE Applications" chapter.

Current Directory of IJServer

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

Windows32/64

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

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

Solaris32/64 Linux32/64

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

(The default J2EE common directory is /opt/FJSVj2ee/var/deployment.)

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. The isj2eadmin command can also be used to implement these settings. For details, refer to the Reference Manual (Command Edition).

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.

| Option of 'current directory' | Directory Name | 'Unique current directory in IJServer' | Current Directory |
|-------------------------------|--|--|--|
| Default directory structure | (Specification is impossible.) Windows32/64 | (Invalid) | Windows32/64 J2EE common directory\ |

| Option of 'current directory' | Directory Name | 'Unique current directory in IJServer' | Current Directory |
|-------------------------------|---|--|---|
| | J2EE common directory\ijserver Solaris32/64 Linux32/64 J2EE common directory/ijserver | | ijserver\[IJServer-name]\current\ [IJServer-name *]\[Process-ID] Solaris32/64 Linux32/64 J2EE common directory/ ijserver/[IJServer-name]/current/ [IJServer-name *]/[Process-ID] |
| Specification by user | Specification example: Windows32/64 C:\tmp\current Solaris32/64 Linux32/64 /tmp/current | When checked | Windows32/64 C:\tmp\current Solaris32/64 Linux32/64 /tmp/current |
| | | When not checked | Windows32/64 C:\tmp\current\[IJServer-name]\ current\[IJServer-name *]\ [Process-ID] Solaris32/64 Linux32/64 /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.
- Solaris32/64 Linux32/64

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.

1.1.5 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 Java VM 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

From the execution class, JNDI can reference the following:

- EJBHome object
 - EJB Home object
 - EJB LocationHome object
- Resources for Reference
 - JDBC data sources
 - Using the JDBC data sources, it is possible to obtain or release database access connections.

The following EJB applications can be called from the execution class:

- STATEFUL Session Bean
- STATELESS Session Bean
- Entity Bean BMP
- Entity Bean CMP1.1
- Entity Bean CMP2.0

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.
- 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 (Java VM) 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. The `isj2eadmin` command can also be used to implement these settings. For details, refer to the Reference Manual (Command Edition).

1. Execution class setting
Store execution class setting information.
2. Class path setting
Set an execution class in the `WorkUnit` class path.

Note

- 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 'Separation of class loaders' is 'Separate between EARs' or 'Separate all', EJB applications cannot be called from a startup time execution class or a shutdown time execution class.
- When processing is performed from the execution class to an EJB application on another Java VM, 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)', a process for an EJB application on the same IJServer cannot be performed from the execution class that is set in the Web container. A call for a process for an EJB application that was made from an execution class on the same IJServer cannot be guaranteed.

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: <ul style="list-style-type: none">- 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. |

- The EJB timer service functionality cannot be used in EJB applications called from execution classes while the execution class is starting or stopping.

1.1.6 Environment Settings for Access to the Shared Resources on the Network



The procedure for referencing and changing shared resources on the network from an application deployed to the IJServer is described below.

Operating the server (server A) running the IJServer containing the shared resources to be accessed

1. Stop Interstage.
2. Create a user with "Administrators" authority.
3. Specify the user created in Step 2 in the TransactionDirector logon account.

To specify the TransactionDirector logon account, log on with "Administrators" authority, and select [Control Panel] - [Services] or [Control Panel] - [Administrative Tools] - [Services]. Select "TransactionDirector", and then click [Logon].

- In Windows Server(R) 2008 or Windows Server (R) 2012, specify the user in Step 2 for the INTERSTAGE and Interstage Operation Tool logon accounts

To do that, log in as a user with "Administrators" privileges, click [Control Panel] - [Services] or [Control Panel] - [Administrative Tools] - [Services], select "INTERSTAGE" and "Interstage Operation Tool", and then click [Logon].

4. Start Interstage.

Operating the server (server B) containing the resources to be shared

1. With the user name and password set up in Step 2 for server A, create a user with "Administrators" authority.
There is no need for this user to have "Administrators" authority if the user has authority to access shared resources.
2. Implement the settings required to share the relevant resources.

Application implementation method

Access to shared resources using a UNC connection

Example

```
\\[ip address]\[shared name]\a.txt OR \\[host name]\[shared name]\a.txt
```

Add escape characters when doing the actual coding.

Note

- In the above procedure, user creation on each server and the implementation of resource sharing settings must be performed on a Windows(R) OS, and Windows(R) OS security should be considered for these operations. This would include, for example, management of the created user's ID and password and access control to prevent access by unintended users
- These settings cannot be used in the Cluster Service.
- Users created in the machine running the IJServer that accesses the common resources (machine A) must not belong to the Guests group. Interstage might not run correctly if a user in the Guests group is specified as the TransactionDirector logon account.

1.1.7 Control when Java Heap/Java Permanent Area is Insufficient

If the Java Heap area or Java Permanent area is insufficient, the following IJServer control options are available. If the IJServer type is "Operate Web application and EJB application on separate Java VMs", settings can be made for the Servlet container and the EJB container.

- Reactivate the process
- Return java.lang.OutOfMemoryError to the application

Each option is explained as follows:

Reactivate the process

When the Java Heap area, Java Permanent area or C Heap area is insufficient, the process belonging to the insufficient area is reactivated.

(Note)

By reactivating the process, the operation can be reopened. By running the IJServer on multiple processes, the processing is performed by another process during process reactivation so that the operation continues and is not interrupted.

The EXTP4435 message is output to the system log when the process is terminated. If the message is generated frequently, specify the Java VM option by referring to the user action in "EXTP4435" in the "Messages Beginning with 'EXTP'" chapter of the Messages.

Return java.lang.OutOfMemoryError to the application

As with normal Java applications, java.lang.OutOfMemoryError is returned to the application and the Servlet container or EJB container.

When java.lang.OutOfMemoryError is returned to the Servlet container or EJB container, the control differs for each container.

- Servlet Container:

Processing is continued as much as possible. Normal operation may not be possible because of missing or insufficient resources or the timing of generation (status 500 due to request processing failure, lost session data, internal errors, etc.)

- EJB Container:

The process is reactivated (**Note**) but the EXTP4435 message is not output.

Unlike when "Reactivate the process" is specified, the process may not be terminated.

Note: Specify the number of reactivation attempts.

In either case, refer to "Tuning of IJServer" in the Tuning Guide for details on the resources used by the Java process, and perform verification and tuning before operation.

1.2 Class Loader

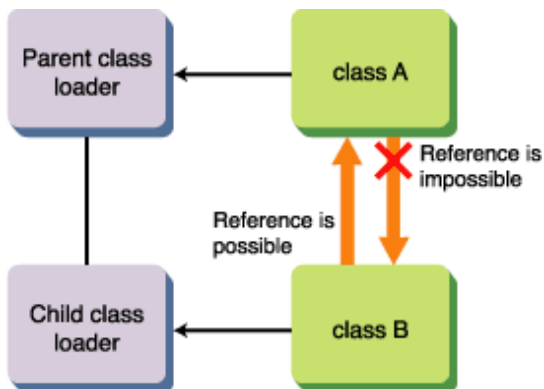
Class loaders of Java provide a function to search for a class file and load classes on the memory.

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

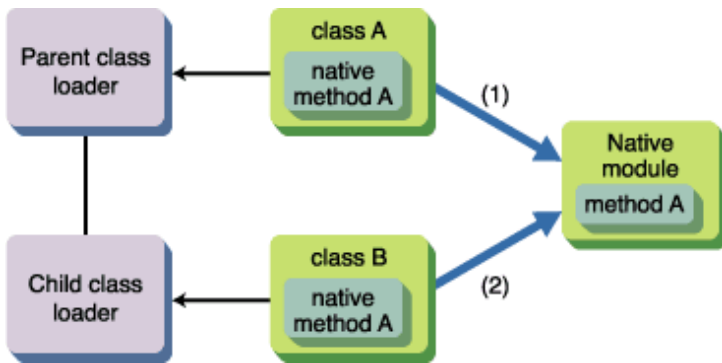
Layer of a Class Loader

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. This 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.

1.2.1 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 "1.2.2 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 "1.2.3 Changing the Search Order of Class Loaders" for the customization method.

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

| 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 | Common classes in IJServer (class file) | application-specific library path |
| | Classes used by Connector that are not Connector classes (class file) Note) When a connector is deployed for IJServer | Manifest classpath in a RAR file |
| | EJB application classes (class file) | ejb-jar file |
| | Classes used by EJB applications that are not EJB application classes (class file) | Manifest classpath in a ejb-jar file |
| | Classes that are commonly used in applications (class file) | Shared/classes directory in an EAR file |
| | Common classes in IJServer (Jar file) | application-specific library path |
| | Connector class (Jar file) Note) When a connector is deployed for IJServer | RAR file |
| | Classes used by Connector that are not Connector classes (Jar file) | Manifest classpath in a RAR file |

| Class Loader | Resources to be Loaded | Setting Method |
|-------------------------|---|--|
| | Note) When a connector is deployed for IJServer | |
| | Classes used by EJB applications that are not EJB application classes (Jar file) | Manifest classpath in a ejb-jar file |
| | Classes that are commonly used in applications (Jar file) | Shared/lib directory in an EAR file |
| Webapp class loader | Web application classes (class file) | WEB-INF/class directory in a WAR file |
| | Classes used by the Web application that are not Web application classes (class file) | Manifest classpath in a WAR file |
| | Note) When included in EAR | |
| | Web application classes (Jar file) | WEB-INF/lib directory in a WAR file |
| | Classes used by the Web application that are not Web application classes (Jar file) | Manifest classpath in a WAR file |
| 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 '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 | Common classes in IJServer (class file) | application-specific library path |
| | Note) When an EAR or ejb-jar is deployed for IJServer | |
| | Classes used by Connector that are not Connector classes (class file) | Manifest classpath in a RAR file |
| | Note) When a Connector is deployed for IJServer | |
| | EJB application classes (class file) | ejb-jar file |
| | Classes used by EJB applications that are not EJB application classes (class file) | Manifest classpath in a ejb-jar file |
| | Classes that are commonly used in applications (class file) | Shared/classes directory in an EAR file |
| | Common classes in IJServer (Jar file) | application-specific library path |
| | Note) When an EAR or ejb-jar is deployed for IJServer | |
| | Connector classes (Jar file) | RAR file |
| Note) When a connector is deployed for IJServer | | |
| Classes used by Connector that are not Connector classes (Jar file) | Manifest classpath in a RAR file | |
| Note) When a Connector is deployed for IJServer | | |
| Classes used by EJB applications that are not EJB application classes (Jar file) | Manifest classpath in a ejb-jar file | |

| Class Loader | Resources to be Loaded | Setting Method |
|--------------------------------------|--|--|
| | Classes that are commonly used in applications (Jar file) | Shared/lib directory in an EAR file |
| Webapp class loader | Common classes in IJServer (class file) Note) When a WAR is deployed for IJServer | application-specific library path |
| | Web application classes (class file) | WEB-INF/class directory in a WAR file |
| | Classes used by the Web application that are not Web application classes (class file) Note) When included in EAR | Manifest classpath in a WAR file |
| | Common classes in IJServer (Jar file) Note) When a WAR is deployed for IJServer | application-specific library path |
| | Web application classes (Jar file) | WEB-INF/lib directory in a WAR file |
| | Classes used by the Web application that are not Web application classes (Jar file) | Manifest classpath in a WAR file |
| 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 |
| | Common classes in IJServer | application-specific library path |
| | Classes common to multiple IJServers | 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 (class file) | WEB-INF/class directory in a WAR file |
| | Classes used by the Web application that are not Web application classes (class file) Note) When included in EAR | Manifest classpath in a WAR file |
| | Web application classes (Jar file) | WEB-INF/lib directory in a WAR file |

| Class Loader | Resources to be Loaded | Setting Method |
|--------------|---|----------------------------------|
| | Classes used by the Web application that are not Web application classes (Jar file) | Manifest classpath in a WAR file |

Note

- The class stored in the IJServer ext directory is set in the system class loader, and used with priority over the above classes. For details, refer to "[1.2.4 Class Settings Used by IJServer](#)".
- In addition to the above, the resources required to run the IJServer are loaded automatically.

1.2.2 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 isj2eeadmin command can also be used to implement these settings. For details, refer to the Reference Manual (Command Edition).

| 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'). |

Note

- 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.

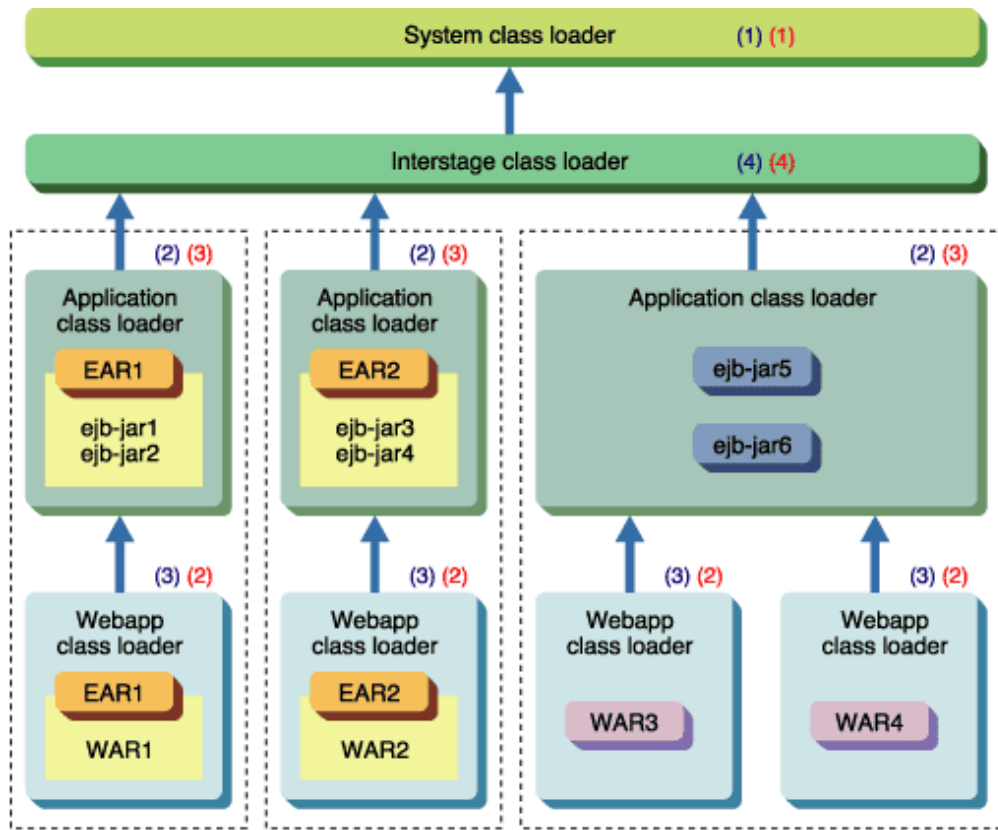
Separation Pattern of Class Loaders

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.

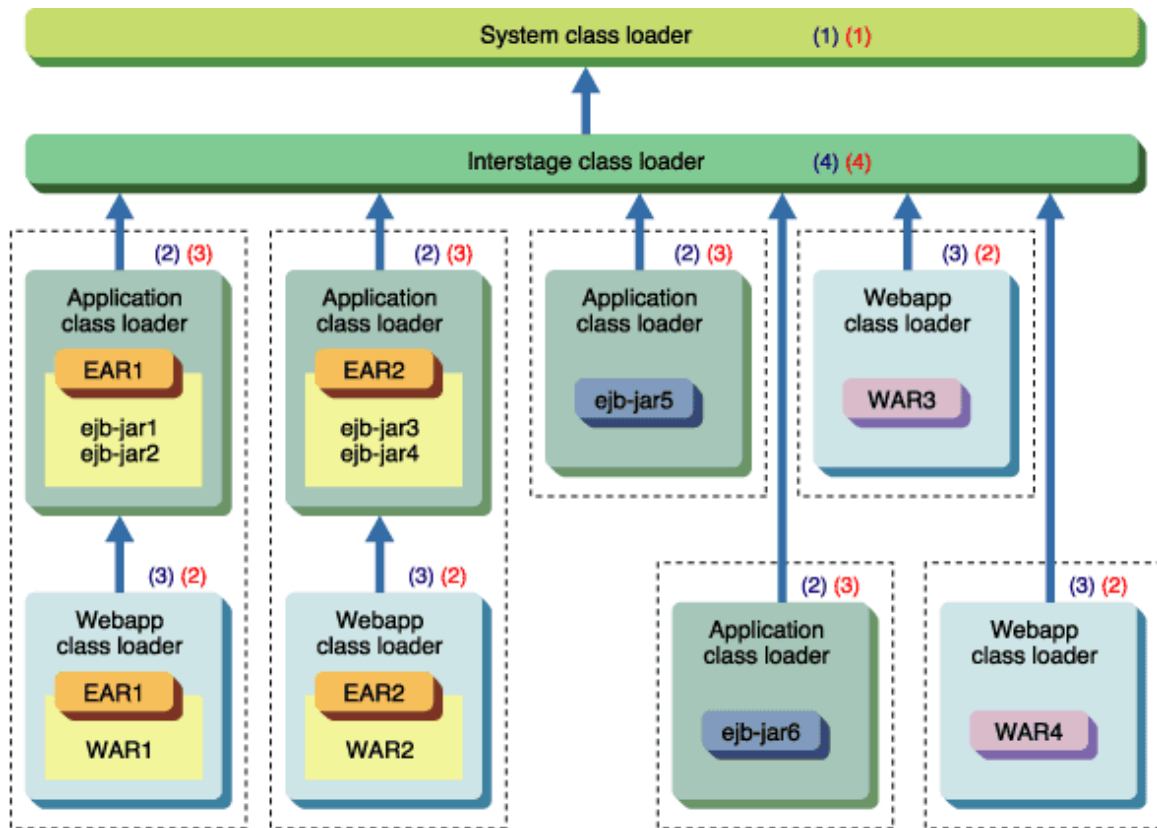


↑ in a figure expresses the child-parent relationship of a class loader. (Terminal points are parents)
 (1), (2), (3), and (4) express the search order of the class loader at the time of setting up "Parent is first".
 (1), (2), (3), and (4) express the search order of the class loader at the time of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



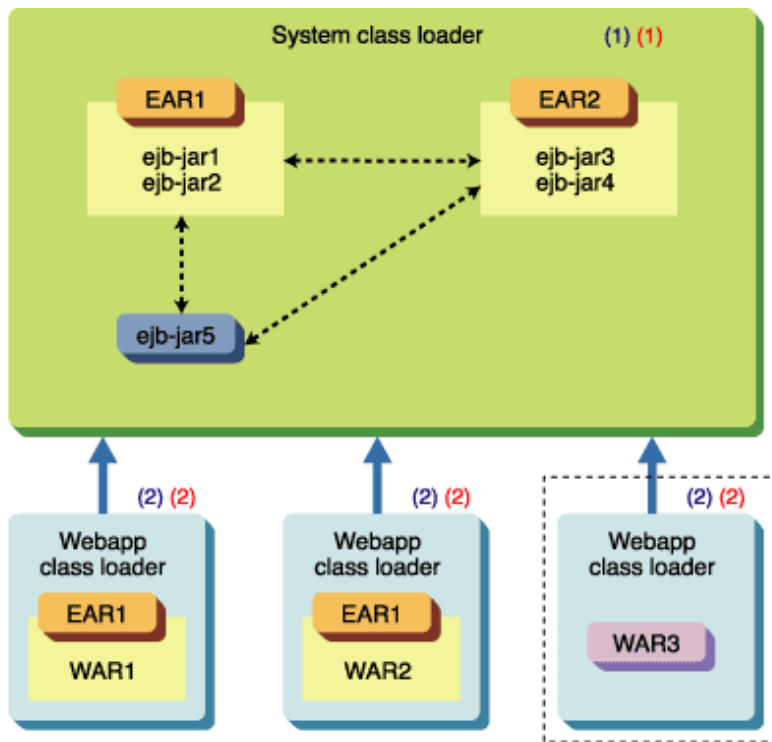
↑ in a figure expresses the child-parent relationship of a class loader. (Terminal points are parents)
 (1), (2), (3), and (4) express the search order of the class loader at the time of setting up "Parent is first".
 (1), (2), (3), and (4) express the search order of the class loader at the time of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

1.2.3 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 isj2eeadmin command can also be used to implement these settings. For details, refer to the Reference Manual (Command Edition).

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 |

| Setting Value | Search Order |
|---------------|--------------------------|
| | 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 are 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) |

1.2.4 Class Settings Used by IJServer

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

Refer to "[1.2.1 Structure of a Class Loader](#)" and "[1.2.3 Changing the Search Order of Class Loaders](#)" for details on the order in which the classes being set are referenced.

For details on the handling by the HotDeploy function and the auto reload function of the classes that have been set, refer to "HotDeploy Function of J2EE" or "Class Auto-reload Function" respectively.

XML Parser

Refer to "[1.2.5 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]

The isj2eadmin command can also be used to implement these settings. For details, refer to the Reference Manual (Command Edition).

Common Classes in IJServer

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

There are four setup methods for the classes commonly used in IJServer: setting up the WorkUnit classpath, storing information in the Shared directory of IJServer, setting up an application-specific library path, and setting up in the IJServer ext directory.

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]

The `isj2eeadmin` command can also be used to implement these settings. For details, refer to the Reference Manual (Command Edition).

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.

Method for Specifying an Application-Specific Library Path

Specify the absolute path to the directory containing the jar file or class file of the application-specific library in the application-specific library path of the WorkUnit.

An application-specific library path for a WorkUnit can be set up from the Interstage Management Console using one of the following methods:

- [WorkUnit] > [Create] > [WorkUnit Configuration] > [application-specific library path]
- [WorkUnit] > [WorkUnit Name] > [Environment Settings] > [WorkUnit Configuration] > [application-specific library path]

The `isj2eeadmin` command can also be used to implement these settings. For details, refer to the Reference Manual (Command Edition).

Method for Setting up in the IJServer ext Directory

By storing the jar files in the IJServer ext directory, they can be used with priority over the class paths required for the container to operate.

Note, however, the following:

- When multiple jar files are stored in the ext directory, they are set for the class path in an arbitrary order.
- Reactivate the IJServer to load the jar files in the ext directory.
- When a class with the same name as the class used by the container is stored in the jar files of the ext directory, the container operates using the class of jar files of the ext directory. This may cause a container operation error, so fully verify container operation before using the function.

To avoid interference by the class required to execute the IJServer and that required to execute the application, the "Method for specifying an application-specific library path" or inclusion of the class in the J2EE application is recommended. For details on including the class in the J2EE application, refer to "Application Classes".

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.

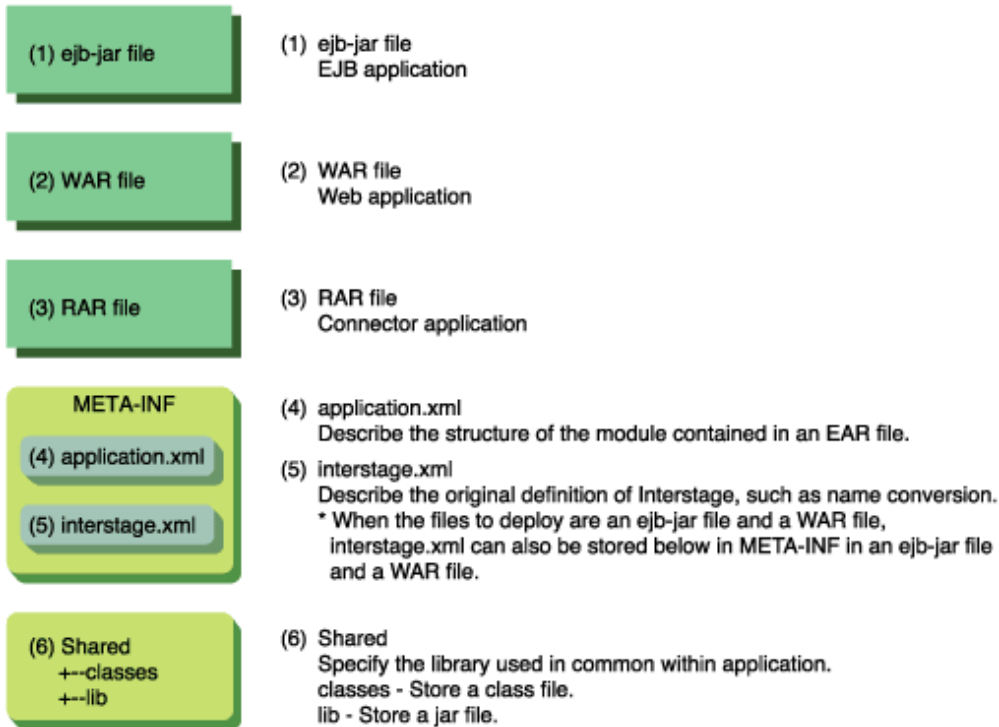
Application Classes

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.
```



Example

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.
- Add the "/" (slash) at the end of the directory in the description in Class-Path.

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.

Confirming the Setting

When the IJServer is activated, the class path information valid at the time of activation is output to the following activation information (info.log) and container log (container.log). The setting of the class path can be confirmed by referring to the activation information and container log. The activation information and container log can be referenced by clicking [WorkUnit] > [WorkUnit name] > [Log Reference] on the Interstage management console.

Windows32/64

Activation information: IJServer-name\log\[process-serial-number]info.log

Container log: IJServer-name\log\[process-serial-number]container.log

Solaris32/64 Linux32/64

Activation information: IJServer-name\log\[process-serial-number]info.log

Container log: IJServer-name\log\[process-serial-number]container.log

1.2.5 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:

- Xerces2
- Fujitsu XML processor Windows32/64 Solaris32 Linux32/64

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

- 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]

The isj2eeadmin command can also be used to implement these settings. For details, refer to the Reference Manual (Command Edition).

Setting values are as follows:

- Xerces2 (default)
- Fujitsu XML processor Windows32/64 Solaris32 Linux32/64
- 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 an implemented class of XML parser into the specified directory. IJServer cannot be started in one of cases.

If there is a defect in the settings (for example the XML parser jar file is deleted) after the settings are configured, the default XML parser is used.

Solaris32/64 Linux32/64

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 | XML Parser Set in the Parameter |
|--|---|
| Xerces2 | None * The XML server contained in JDK is used. |
| Windows32/64 Solaris32 Linux32/64 Fujitsu XML processor | The following is set in "-classpath" when the IJServer is activated. Windows32/64 Windows-shared-file-folder\FujitsuXML\xmlpro.jar Windows-shared-file-folder \FujitsuXML\xmltransx.jar Solaris32 Linux32/64 /opt/FJSVxmlpc/lib/xmlpro.jar /opt/FJSVxmlpc/lib/xmltransx.jar |
| Others | Set to the Java VM option: '-Djava.endorsed.dirs' at the start of IJServer. |

- 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.

Note

- 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

Windows32/64

When JDK is used : C:\Interstage\JDK6\jre\lib\jaxp.properties

When JRE is used : C:\Interstage\JRE6\lib\jaxp.properties

Solaris32/64 **Linux32/64**

When JDK is used : /opt/FJSVawjbc/jdk6/jre/lib/jaxp.properties

When JRE is used : /opt/FJSVawjbc/jre6/lib/jaxp.properties

1.2.6 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

| Item | Description | |
|---------------------|---|-----------------------|
| Time stamp | Date and hour of the loading of the class | |
| Class name | Class name of the loaded class (including the package name) | |
| Repository | Storage directory name or jar file name of the loaded class | |
| Class loader type | Name of the class loader that loaded the class | |
| | Class Loader | Displayed Name |
| | System class loader | System |
| | Interstage class loader | Interstage |
| | Catalina class loader | Catalina |
| | Note) Used in the Interstage system | |
| | Application class loader | Application |
| Webapp class loader | Webapp | |

Note

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 "[1.2.1 Structure of a Class Loader](#)".

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

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

Set the following option in the Java VM option of the work unit to output the trace data:

```
-Dcom.fujitsu.interstage.j2ee.ijserver.loader.trace=true
```

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. Set the following option in the Java VM option of the work unit:
-Dcom.fujitsu.interstage.j2ee.ijserver.loader.trace=true
2. Execute a reproduction test.
3. Search for the trace information that is output in the container log using ClassA. This detects the following two 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 determined that ClassA is stored in two locations, the EJB application and the Web application.

Investigation

Since the EJB application call from Servlet failed because of the class loading, 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'?

1.2.7 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.

The JDBC drivers set in the class path must not be stored in the "WEB-INF/lib" directory in Web application directory structure. If the JDBC drivers are stored in this directory, it may not be possible to use the following functions:

- Pre-opened connection count
- Reconnect on Failure

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.

Windows32/64

Path

Solaris32/64 Linux32/64

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.

1.3 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 the "JNDI" chapter for details of using UserTransaction.

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

Default Transaction

A J2EE application on the same Java VM 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 Java VM, 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 Windows32/64 Solaris32 Linux32/64

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

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

Note

- Because the distributed transaction can be used only with an EJB application, specify one of the following two as the IIServer operation type.

- Process where only EJB applications are operated
- Operating both Web and EJB applications (on separate Java VMs)

Specify the distributed transaction when the IIServer is defined.

Refer to "Part V, 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.
- IIServers 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 Windows32/64 Solaris32 Linux32/64

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 allows the user to start/stop Interstage and IJServer (operation) or set up the environment.

Version 8.0 provides the *isj2eeadmin* command, which has features that support large-scale system construction, such as the ability to create the same IJServer on multiple servers.

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

- Preparing J2EE Applications
- Creating an IJServer
- Deploying and Setting J2EE Applications
- Preparation for Servlet Service Operation
- Request Distribution Control by Web Server Connector
- Procedure for Using JTS Windows32/64 Solaris32 Linux32/64
- Procedure for Using JMS
- Procedure for Using JavaMail
- Customizing and Checking the Operating Environment
- Debugging Applications

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

- Refer to the "JNDI" chapter for details of the resource definition/reference function using JNDI.
- Refer to the "The J2EE Application Security Function" chapter 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.

Note

For details on accessing files on the network drive from an application deployed to the IJServer, refer to "Environment Settings for Access to the Shared Resources on the Network" in the "Design of J2EE Application" chapter.

2.1 Preparing J2EE Applications

This section explains how to prepare J2EE applications.

Developing J2EE Applications

J2EE applications need to be developed.

Refer to the following for details on development:

- "Referencing Objects" in the "JNDI" chapter, for referencing resources and EJB.
- The "Web Application Development" chapter, for Web application.
- The "EJB Application Development" chapter, for EJB application.

Refer to the "How to Create Entity Beans" chapter and the "How to Call EJB Applications" chapter, for details on development in each runtime environment.

- The "Developing Web Services" chapter, for Web service applications.

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 Setup" in the "JNDI" chapter, for J2EE application clients.
- "Web Application Environment Definition File (Deployment Descriptor)" in the "Web Application Development" chapter, for Web applications.
- For EJB applications, use the EJB Deployment Descriptor editor of Interstage Studio. Refer to the Interstage Studio User's Guide.
- "Editing the Deployment Descriptors" in the "Developing Web Services" chapter, for Web service applications.

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.

2.2 Creating an IJServer

The following two methods can be used to create an IJServer:

- Interstage Management Console

Select **WorkUnit** and then **Create a new WorkUnit**.

- *isj2eeadmin* command

Create the Work Unit with the J2EE operation command "isj2eeadmin". Refer to the Reference Manual (Command Edition) for a detailed description of the isj2eeadmin command.

2.3 Deploying and Setting J2EE Applications

This section explains how to deploy and set J2EE applications.

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 Commands" in the Reference Manual (Command Edition) for details of the *ijsdeployment* command.

Web applications can be run without being copied to the IJServer directory. For details, refer to "[2.3.5 Deploy the Web Application Anywhere on the Server](#)".

- 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 the "Design of J2EE Application" chapter, for details of the J2EE application client expansion destination.

Note

- It is also used when the XML parser specified in the IJServer environment settings is deployed. The JAXP version to which the XML parser required at the time of deployment must conform depends on the module type deployed and the version. For details, refer to "[2.3.1 XML Parser Settings Required for Deployment](#)".
- EJB application deployment runs Javac and therefore fails in a JRE environment. Install a JDK environment for this purpose.

See

Refer to "[2.3.3 HotDeploy Function of J2EE](#)" for details of the HotDeploy function.

Refer to "[2.3.4 Class Auto-reload Function](#)" for details of the class auto-reload function.

Refer to "[2.3.5 Deploy the Web Application Anywhere on the Server](#)" for details on deploying web applications that execute on arbitrary server locations.

Setting J2EE Applications

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.

JSP Pre-compile

The first JSP access causes deterioration in response time because it triggers the JSP compile program to run. This can be avoided by compiling the JSP before the IJServer starts.

If pre-compile is used together with the JSP auto reload function, it is possible to overwrite the JSP while the IJServer is running. The changes are not necessarily effective immediately after the JSP pre-compile is completed. The time at which changes take effect depends on the settings of JSP reload function.

For command details, refer to "*ijscompilejsp*" in the Reference Manual (Command Edition).

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 the "JNDI" chapter for details.

HTTP Tunneling

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

Application File Protection Level Solaris32/64 Linux32/64

Access rights to an application file are set according to the application file protection level in the IJServer definition, as follows:

- High (administrator only)
 - Directory: 755
 - File: 644
- Low (ordinary users)
 - Directory: 777
 - File: 666

Access rights to the following resources can be changed:

- IJServer directory or apps directory (and resources in that directory)
- IJServer directory or Shared/lib directory

- IJServer directory or Shared/classes directory
- IJServer directory or ext directory

The default setting is "High (administrator only)". The application file protection level can be changed as required.

If this option is selected, the file and directory owner do not change. For example, if [High (Administrator only)] is selected when a file is stored by a general user, only the file owner and administrator can access the file.

Note

When the IJServer is activated, 022 is set in umask. To generate files from the IJServer, follow the umask setting.

2.3.1 XML Parser Settings Required for Deployment

The JAXP version to which the XML parser is required to conform at the time of module deployment depends on the module type and the version. The relationship is shown as follows:

| Type and version of the module | Required JAXP version |
|--------------------------------|-----------------------|
| J2EE1.4(EAR) | 1.2 or later |
| Servlet2.4(WAR) | 1.2 or later |
| EJB2.1(ejb-jar) | 1.2 or later |
| Connector1.5(RAR) | 1.2 or later |
| J2EE1.3(EAR) | 1.1 or later |
| Servlet2.3(WAR) | 1.1 or later |
| EJB2.0(ejb-jar) | 1.1 or later |
| Connector1.0(RAR) | 1.1 or later |
| J2EE1.2(EAR) | 1.1 or later |
| Servlet2.2(WAR) | 1.1 or later |
| EJB1.1(ejb-jar) | 1.1 or later |

Refer to the above table, and set the XML parser that is appropriate to the module deployed in the IJServer environment settings.

For the method of setting the XML server, refer to "Settings of XML Parser" in the "Design of J2EE Application" chapter.

The relationship between the XML parser that can be set in the IJServer environment settings and the JAXP version is shown in the following table:

| XML parser | JAXP version |
|-----------------------|---------------------------------|
| Xerces2 | 1.3 |
| Crimson | 1.1 |
| Fujitsu XML Processor | 1.2 |
| Other | Depends on the parser specified |

Refer to HELP on the Interstage management console for the IJServer environment settings.

Note

- The XML parser selected in the IJServer environment settings is valid, if deployed, only when the IJServer of V9.0 or later version is used. The XML parser is used when Crimson is deployed if the IJServer of V8.0 compatible mode or that created by Interstage V8 or earlier version is used.

- Encoding that is defined as follows in the deployment descriptor depends on the encoding supported by each XML parser. When a module that was deployed in other environment is deployed and the deployment descriptor cannot be read, check if the XML parser being used supports the encoding specified. Use of UTF-8, which is supported by most of XML parsers, is recommended.

```
<?xml version="1.0" encoding="UTF-8"?>
...
```

2.3.2 J2EE Application (EAR File) Deployment Descriptor

Set the operation environment for J2EE applications for the J2EE application (EAR file) deployment descriptor (application.xml).

How to describe application.xml is explained as follows:

How to Describe application.xml

application.xml is a deployment descriptor of the J2EE application (EAR file)

It describes the path for the configuration items in the J2EE application or the security role. The path described is the path in the module that was valid at the time of EAR file packaging.

The description format of the deployment descriptor is the XML format. The description format of the deployment descriptor is shown as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/application_1_4.xsd"
  version="1.4">
  <display-name>display_name</display-name>
  <module>
    <connector>uri</connector>
    <alt-dd>uri</alt-dd>
  </module>
  <module>
    <ejb>uri</ejb>
    <alt-dd>uri</alt-dd>
  </module>
  <module>
    <java>uri</java>
    <alt-dd>uri</alt-dd>
  </module>
  <module>
    <web>
      <web-uri>uri</web-uri>
      <context-root>context-root</context-root>
    </web>
    <alt-dd>uri</alt-dd>
  </module>
  <security-role>
    <role-name>role-name</role-name>
  </security-role>
</application>
```



Note

Notes on Description

- Because <?xml...> specified at the beginning describes the XML declaration, it may not be omitted from the beginning of the deployment descriptor file.

- Because the application tag and the application tag attributes describe the version of the J2EE namespace application schema, they must always follow the XML declaration.
- <application> and </application> are root tags marking the beginning and the end of the XML file. They must always be specified.
- Describe the tags in the order shown above.
- The description is case sensitive.
- The deployment descriptor of J2EE 1.2 or J2EE 1.3 can also be used.

application.xml Tag

The following tags can be specified in the J2EE application (EAR file) deployment descriptor (application.xml):

Contents to be Defined

| Tag name | Meaning | Omitting the tag | Specifying in multiples |
|---------------|---|------------------|-------------------------|
| Application | The beginning and the end of the deployment descriptor of the J2EE application (EAR file) are defined. | No | No |
| display-name | The J2EE application (EAR file) name is defined. | Yes | No |
| module | Definitions related to one module are described. Note: One of connector, ejb, java or web must always be defined. | No | Yes |
| connector | The connector (RAR file) path is defined. | No | No |
| ejb | The EJB module (ejb-jar file) path is defined. | No | No |
| java | The J2EE application client (client-jar file) path is defined. | No | No |
| web | The definitions related to the Web module are described. | No | No |
| web-uri | The Web module (WAR file) path is defined. | No | No |
| context-root | The Web application name is defined. Note: The definition must be a value unique in the EAR file. | No | No |
| alt-dd | The module deployment descriptor path is defined. If omitted, the deployment descriptor contained in the module is used. | Yes | No |
| security-role | The security role used for access restriction is used. | Yes | Yes |
| role-name | The security role name is defined. The role name specified must be the security role name defined in the security function operation settings (Interstage directory service). Note: The definition must be a value unique in the EAR file. | No | No |

Note

- A tag with "No" displayed in the "omitting the tag" column cannot be deployed if omitted.
- A tag with "No" displayed in the "specifying in multiples" column cannot be deployed if specified in duplication.

Example

The example of the deployment descriptor with the following module configuration is shown as follows:

The EAR file includes the following:

- connector.rar
- ejb.jar
- client.jar
- web.war
- connector.xml
- ejb.xml
- client.xml
- web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/application_1_4.xsd"
  version="1.4">
  <display-name>J2EEApplication</display-name>
  <module>
    <connector>connector.rar</connector>
    <alt-dd>connector.xml</alt-dd>
  </module>
  <module>
    <ejb>ejb.jar</ejb>
    <alt-dd>ejb.xml</alt-dd>
  </module>
  <module>
    <java>client.jar</java>
    <alt-dd>client.xml</alt-dd>
  </module>
  <module>
    <web>
      <web-uri>web.war</web-uri >
      <context-root>WebApplication</context-root>
    </web>
    <alt-dd>web.xml</alt-dd>
  </module>
  <security-role>
    <role-name>Administrator</role-name>
  </security-role>
  <security-role>
    <role-name>Operator</role-name>
  </security-role></application>

```

2.3.3 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 "2.3.4 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

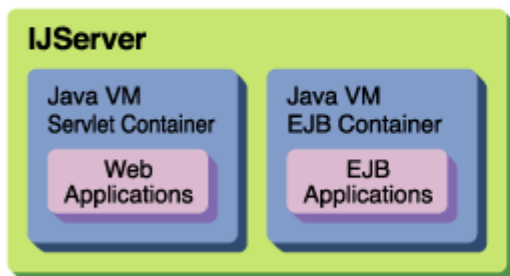
If the IJServer type is 'Web and EJB applications are operated in different JavaVMs', the behavior is as shown below.

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

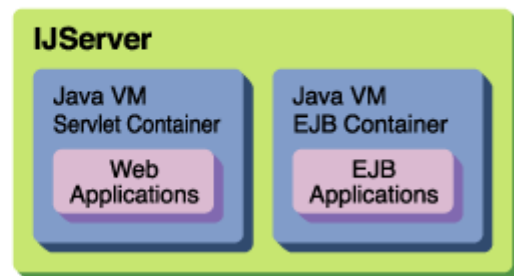
- Deploying Web and EJB applications

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.

● In case the HotDeploy function is not used



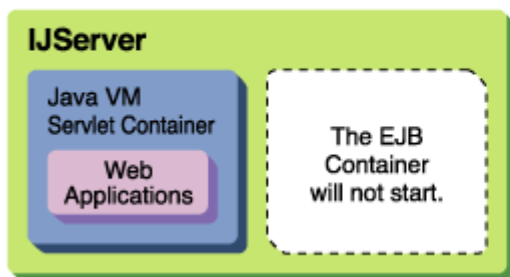
● In case the HotDeploy function is used



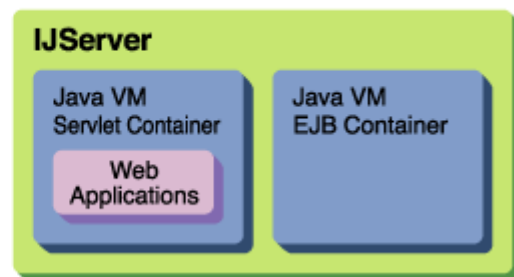
- Deploying Web applications only

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

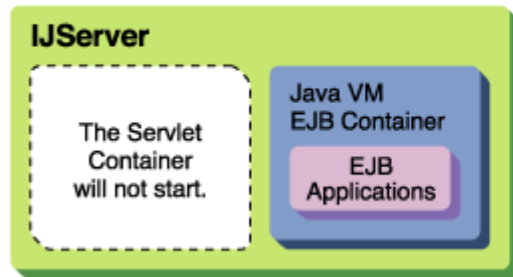


● In case the HotDeploy function is used

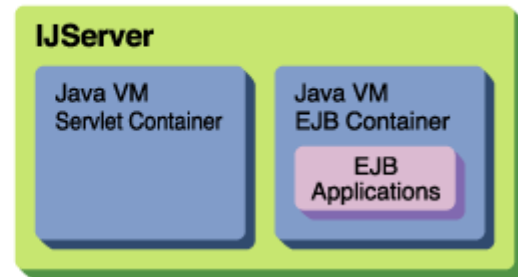


Java VMs on which applications have been deployed start. If the HotDeploy function is used, all Java VMs (Servlet container) start regardless of whether or not a Web application has been deployed in a Servlet container.

● **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
EJB applications for which activation was successful start.
- The HotDeploy function is not used
No EJB applications 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.

| Deployed Module Status/ Type | | The HotDeploy Function is not used | | The HotDeploy Function is used | |
|---------------------------------|-----------------|--|------------------------------|--------------------------------|------------------------------|
| | | 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 | Starts | | | |
| | 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.

For deployment or redeployment, use the *ijsdeployment* command. For undeployment, use the *ijsundeployment* command.

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.

Execute the *ijsdeployment* command as follows:

```
ijsdeployment -n IJServer-WorkUnit -f deployment-target-module
```

2. Redeployment (redployment 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.

Execute the *ijsdeployment* command as follows:

```
ijsdeployment -n IJServer-WorkUnit -f deployment-target-module -r
```

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.

Execute the *ijsundeployment* command as follows:

```
ijsundeployment -n IJServer-WorkUnit -f deployment-module
```

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 restart the IJServer.

Refer to "[Status of Deployed Modules](#)" for the status of deployed modules. Refer to 4. above 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 being activated and going to start to accepting requests. | - |
| Deactivating | The deployed module is being deactivated and 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. An unexpected error has been generated during the activation, deactivation, or auto-reload processing of the deployed module. | Refer to the container log and remove the cause of the error. 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 the "Design of J2EE Application" chapter, 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 | (*1) | (*1) | (*1) | (*1) |

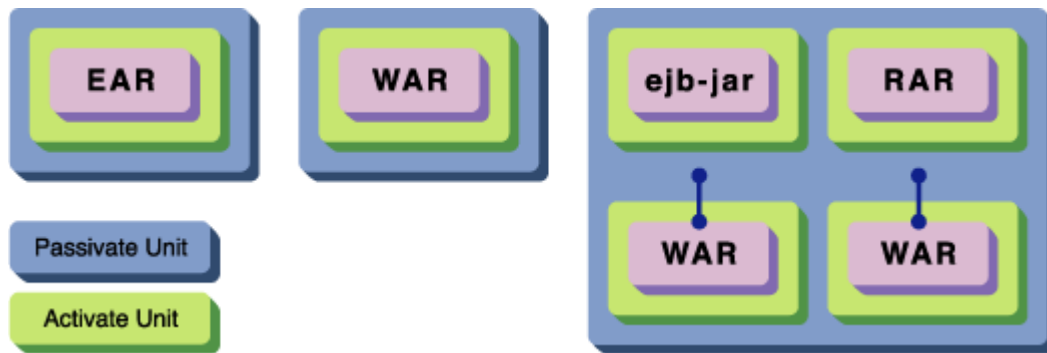
*1 The HotDeploy function cannot be used.

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.

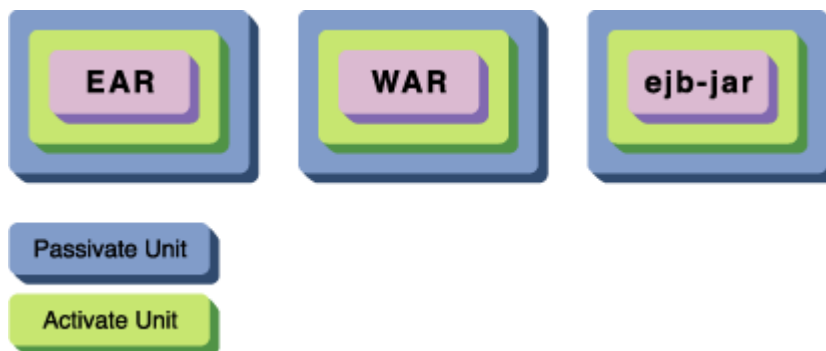
- Separating between EARs

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.
Note that the Servlet session is backed up and recovered automatically If the session recovery function is in use, and so can continue.
 - 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 to be Taken when Class Loaders are Used".
 - If you are running a Web service client on IJServer, do not use the HotDeploy function in the application.
 - Overwrite deployment may fail if the Java VM such as another IJServer is directly referring to EJB application client distribution items. Change this so that the client distribution items are referred to after a copy is made.
-

2.3.4 Class Auto-reload Function

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

Class auto-reload is performed if the following changes are made:

- If a jar file in the application was overwritten
- If a class file that was loaded when the application was executed was overwritten
- If a new jar file was added to WEB-INF/lib or Shared/lib in ear

To make the settings for using the class auto-reload function, in the Interstage Management Console click [WorkUnit] > [IJServer name] > [Environment Settings] . Select 'Yes' for the Use of auto-reload function.

If the class auto-reload function is used, a corrected application class or Jar 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.

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 "[2.3.3 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.

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 the "Design of J2EE Application" chapter, for details of the copy destination (deployment directory).

Windows32/64

- 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 rar files are deployed
 - [J2EE common directory]\ijserver\[IJServer name]\apps\[RAR file name]\lib is '.jar'
- 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'

Solaris32/64 Linux32/64

- If WAR files are deployed
 - [J2EE common directory]/ijserver/[IJServer name]/apps/[Web module name]/WEB-INF/lib is '.jar'
 - [J2EE common directory]/ijserver/[IJServer name]/apps/[Web module name]/WEB-INF/classes lib is '.class'
- If ejb-jar files are deployed
 - [J2EE common directory]/ijserver/[IJServer name]/apps/[EJB module name]/ lib is '.class'
- If rar files are deployed
 - [J2EE common directory]/ijserver/[IJServer name]/apps/[RAR file name]/lib is '.jar'
- 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]/[WAR file name]/WEB-INF/lib is '.class'
 - [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 is '.jar'
 - [J2EE common directory]/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 Settings | 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 | All classes of ejb-jar, RAR, and WAR deployed individually | 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 | None (This is not a target of auto-reload because deployment cannot be performed) | All classes of modules included in EAR |
| Don't separate | - The class auto-reload function cannot be used. | | | | | | |

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.



Note

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.
- Even when the application is in the deactive state, the class files or the jar files contained in the application or referenced by the application can be loaded by the class loader, and become subject to the monitoring for reloading.

This applies to the following class files and jar files:

- Class files already loaded by extended application activation processing
- jar files in RAR

- jar files specified in the class-path of the ejb.jar or RAR manifest files.

Therefore, the files are checked for reloading when updated, and the application operating on the same class loader is reloaded.

Non-active applications are not activated in this instance.

To activate them, redeploy or reactivate using the HotDeploy function or reactivate the work unit.

- 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.
- The class auto-reload function only switches classes. For this reason, the result of changing a definition such as deployment descriptor (except for web.xml of Web application) in the Interstage Management Console system, resources or definitions specified in IJServer is invalid.
- The deployment descriptor (web.xml) of the Web application is not subject to monitoring but is re-read if the class files or the jar files are updated.

To reflect the updating of the deployment descriptor (web.xml), update the deployment descriptor(web.xml), then update other class files or jar files already loaded, and run the auto-reload function.

To update only the deployment descriptor (web.xml), reactivate the module or the work unit.

Even if the Servlet definition is lost due to the deployment descriptor (web.xml) update, data up to that point is accumulated and displayed on the Web application monitor of the Interstage management console. To reset the data, reactivate the module or the work unit.

- 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" in the "Design of J2EE Application" chapter.

For details of how to edit interstage.xml, refer to "interstage.xml File" in the "JNDI" chapter.

- 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 | Works | 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 to be Taken when Class Loaders are Used".
- If you are running a Web service client on IJServer, do not use the auto-reload function in the application.

Windows32/64

When the IJServer has been activated, the jar files are being used by the process and so it may not be possible to overwrite or delete them.

In this instance, after reflecting the jar files, terminate the IJServer and reactivate. Alternatively, undeploy and redeploy using the HotDeploy function.

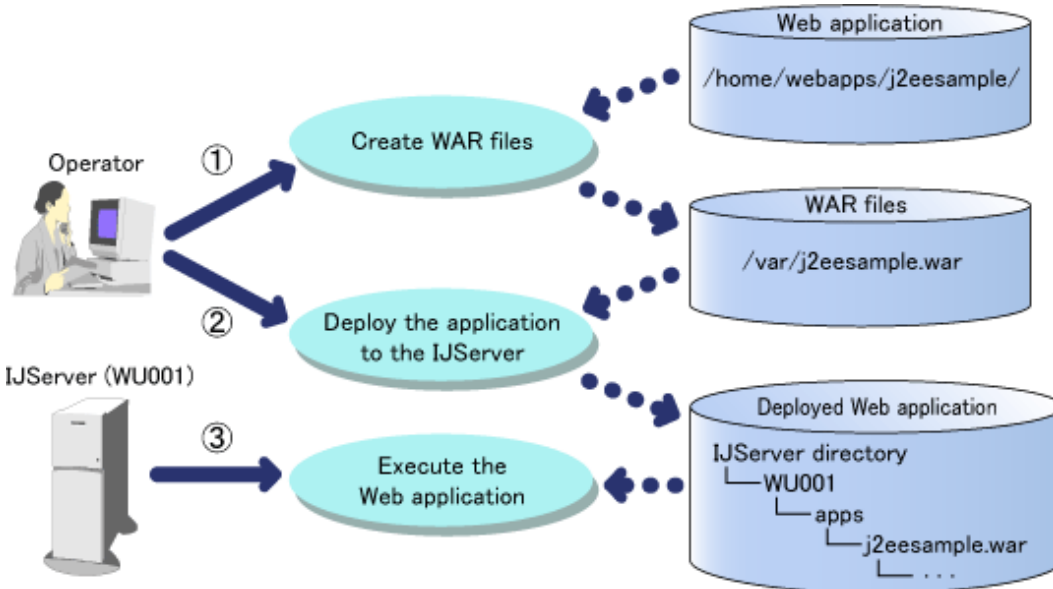
Solaris32/64 Linux32/64

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.



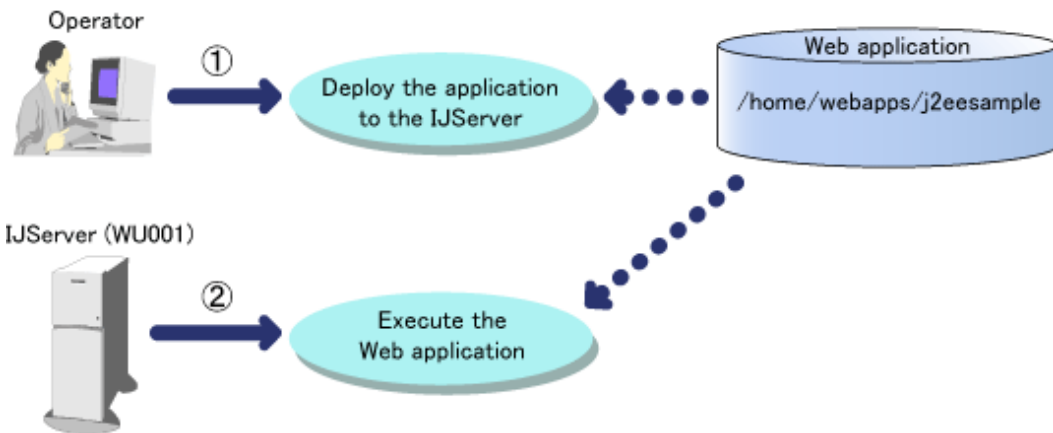
2.3.5 Deploy the Web Application Anywhere on the Server

To deploy a Web application to the IJServer in a version of Interstage Application Server prior to V7, set "Create WAR files" in the Web application deployed to the server, as shown in the figure below. Set "Deploy to the IJServer" for the created WAR files to deploy the Web application to a directory on the IJServer. Once this has been done, set "Execute the Web application" for the Web application deployed to the directory on the IJServer.



In Interstage Application Server 8.0 or later, a Web application developed for a specific directory on a server can be operated on IJServer without copying it to the IJServer directory.

When deploying an application using the Interstage Management Console, select "Deploy the Web Application Anywhere on the Server". For details on deploying applications using the `ijsdeployment` command, refer to the Reference Manual (Command Edition).

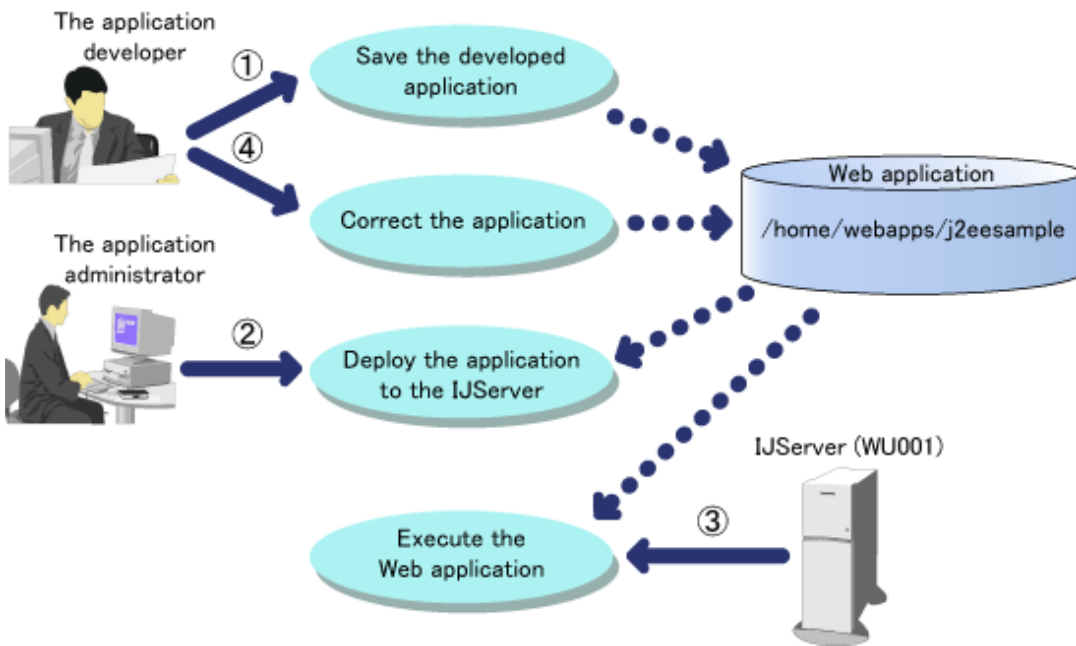


Note

- Assign the required authority to the IJServer start user for the resources in the directory to which the Web application is deployed. If the IJServer start user does not have Read authority for classfiles, JAR files, JSP, and static resources (such as html), any attempt to read the resources will fail.
- This deployment method cannot be used for the Web application that contains the Web services.

Application Method for Executing One Application on One IJServer

A Web application developed for a specific directory on a server can be operated on IJServer, without copying it to the IJServer directory.



1. Save the developed application

The application developer saves the developed application in an arbitrary directory on the server.

2. Deploy the application to the IJServer

The application administrator deploys the Web application saved in an arbitrary directory on the server to the IJServer.

3. Execute the Web application

The IJServer is used to execute the Web application saved in an arbitrary directory on the server.

4. Correct the application

If it is necessary to correct the application, the application developer can overwrite the resources of the Web application saved in an arbitrary directory on the server.

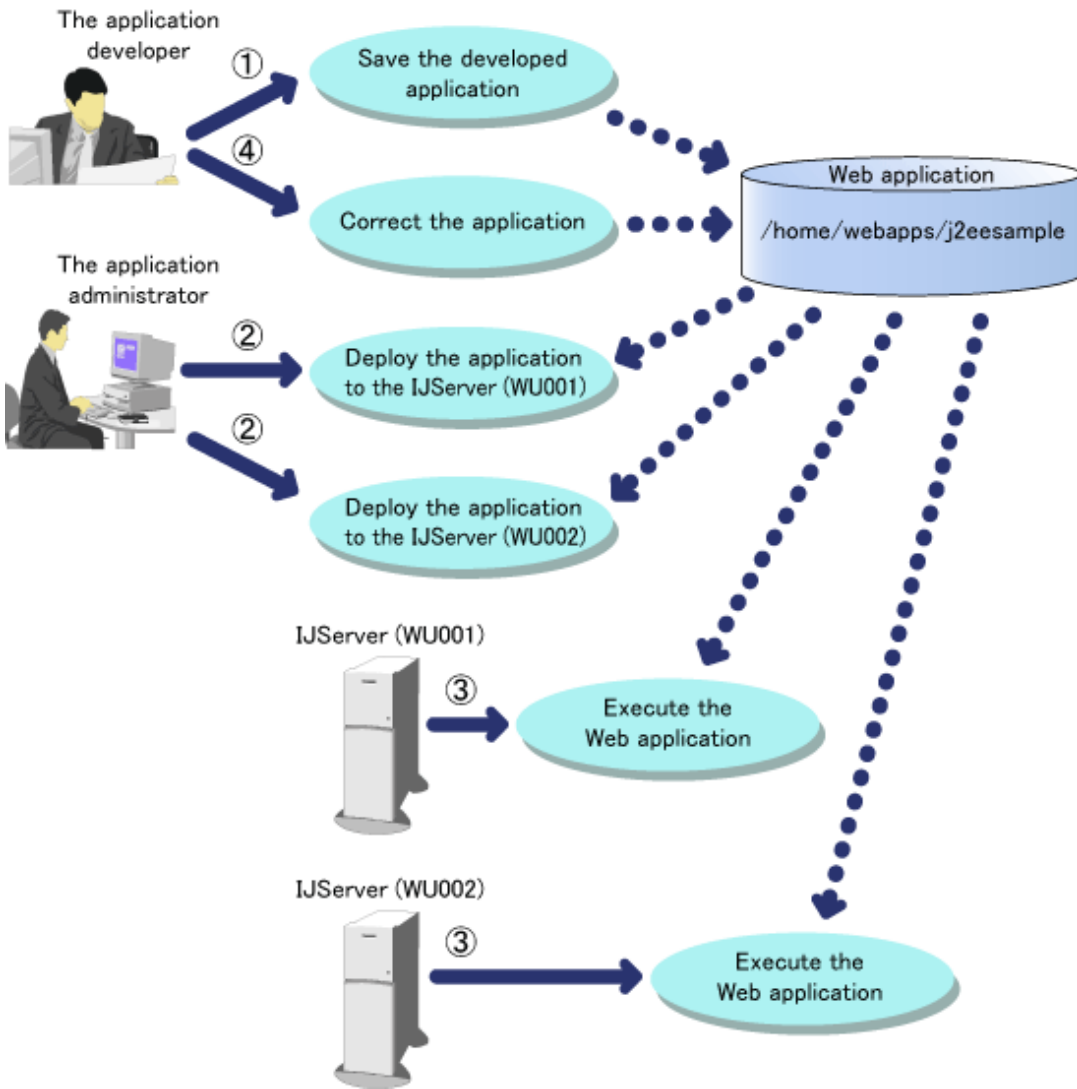
If classfiles or JAR files are overwritten, the changes are enabled after the class is auto reloaded or reactivated, or after the IJServer is restarted.

If JSP files are overwritten, the changes are enabled after the JSP file is reloaded or after the IJServer is restarted. If a static file (such as html) is overwritten, the changes are enabled as soon as it is overwritten.

Application Method for Executing One Application on more than One IJServer

As shown below, one Web application can be executed on more than one IJServer.

In this case, a single correction can be applied to more than one IJServer.



1. Save the developed application

The application developer saves the developed application in an arbitrary directory on the server.

2. Deploy the application to the IJServer

The application administrator deploys the Web application saved in an arbitrary directory on the server to two IJServers (WU001 and WU002).

3. Execute the Web application

The two IJServers (WU001 and WU002) are used to execute Web applications saved in the same directory.

Note: In this case, more than one IJServer is used to execute the same Web application. If the resources in the Web application are overwritten, there is exclusive access control for those resources.

4. Correct the application

If it is necessary to correct the application, the application developer can overwrite the resources of the Web application saved in an arbitrary directory on the server.

In this case, the overwritten resources are updated on two IJServers (WU001 and WU002).

If classfiles or JAR files are overwritten, the changes are enabled after the class is auto reloaded or reactivated, or after the IJServer is restarted.

If JSP files are overwritten, the changes are enabled after the JSP file is reloaded or after the IJServer is restarted.

If a static file (such as html) is overwritten, the changes are enabled as soon as it is overwritten.

2.3.6 Pre-configuring the Deployment Settings

To pre-configure Web application settings using the Web module definition file, the file (interstage-web.xml) should be stored in the META-INF directory of the deployment file.

When the Web module definition file is deployed, the Web Application Settings specified in the file will be configured. These are the same settings found in [System] > [WorkUnit] > [WU001] > [Deploy] > [Detailed Settings] > [Web Application Settings] in the Interstage Management Console.

This method of pre-configuring the Web application settings can also be used when deployment is performed according to the information in "2.3.5 Deploy the Web Application Anywhere on the Server". In this case, the Web module definition file should be stored in the META-INF directory of the Web application storage destination directory.

The Web module definition file can only be used when deployment is performed in a stand-alone environment. It cannot be used in deployment using application management in a multiserver environment.



Note

If the settings in a Web module definition file stored in the module deployment destination are changed after deployment, the settings become invalid.

To change the settings, use either of the following methods:

- Using the Interstage Management Console, enter the settings in the [Application Settings] window of the corresponding module, and then click [Update].
- Change the settings of the Web module definition file stored in the deployment file, and then perform the deployment again.

If deployment was performed by selecting "Deploy the Web Application Anywhere on the Server", change the settings of the Web module definition file stored in the directory, and then perform deployment again.

Web Module Definition File (interstage-web.xml)

The Web module definition file (interstage-web.xml) is described in the following sections:

- Web module definition file description format
- List of Web module definition file tags
- Web module definition file tag details
- Web module definition file examples

Web Module Definition File Description Format

The format used for Web module definition file description is XML. The description format is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<interstage-web-app>
  <war>
    <context-root>context-root</context-root>
    <shared-context>shared-context</shared-context>
    <session>
      <cookie>cookie</cookie>
      <web-browser>web-browser</web-browser>
      <cookie-security>cookie-security</cookie-security>
    </session>
    <encoding>encoding</encoding>
    <authentication>
      <web-server>web-server</web-server>
    </authentication>
  </war>
</interstage-web-app>
```

Note

- For details about the order in which each tag is written, refer to the above.
- `<?xml...>` describes the XML declaration, and therefore must not be omitted from the beginning of the file. To use multibyte characters (including comments), specify an appropriate encoding type, such as "UTF-8", in the "encoding=" part.
- `<interstage-web-app>` and `</interstage-web-app>` are the XML file 'start' and 'end' root tags. These tags must be specified.
- Control characters (such as spaces, tabs, and line feed characters) specified at the beginning or end of a specified value are ignored.
- To use the default value for each item, omit the tag itself. Do not set a null value in the tag.

List of Web Module Definition File Tags

The following table lists the Web module definition file tags.

The column "Support in the Interstage Management Console window" refers to the [System] > [WorkUnit] > [JServer] > [Deploy] > [Detailed Settings] window in the Interstage Management Console.

| Tag | Optional | Can be specified more than once | Support in the Interstage Management Console window |
|-----------------|----------|---------------------------------|--|
| War | Yes | No | Web Application Settings |
| context-root | Yes | No | Web Application Name |
| Shared-context | Yes | No | Shared Context |
| Session | Yes | No | Session |
| Cookie | Yes | No | Store session information in cookies? |
| web-browser | Yes | No | Store session information in web browser |
| cookie-security | Yes | No | Always add the Secure attribute to cookies |
| Encoding | Yes | No | Encoding |
| Authentication | Yes | No | Certification method |
| web-server | Yes | No | Use/Do not use Web server authentication information |

Web Module Definition File Tag Details

The values specified for each tag are explained below.

context-root

Specify the Web application name using a maximum of 64 characters.

If the Web application name is specified in the `-c` option of the `ijsdeployment` command and deployment is performed, the value specified in the `-c` option is valid, not the value of this tag.

For details about the `ijsdeployment` command, refer to "J2EE Operation Commands" in the Reference Manual (Command Edition).

shared-context

Select whether to allow dispatch to other Web applications. Select either of the options shown below. No distinction is made between upper and lower case.

- True (Dispatch is allowed)
- False (Dispatch is not allowed) (This is the default value)

cookie

Specify one of the following options to specify whether to manage sessions using cookies. No distinction is made between upper and lower case.

- True (Cookies are used) (This is the default value)
- False (Cookies are not used)

web-browser

Use this tag to specify if, when the Web browser is briefly closed and then restarted, reconnection is enabled if the session is still valid. Select one of the following options. No distinction is made between upper and lower case.

- True (Session information is stored in the web browser)
- False (Session information is not stored in the web browser) (This is the default value)



If 'False' is specified in the 'cookie' tag, this setting is ignored.

cookie-security

Select one of the following options to specify whether to use session management. No distinction is made between upper and lower case.

- Always (This is always assigned)
- Auto (This is determined by the method used for the connection and is the default value)



If 'False' is specified in the 'cookie' tag, this setting is ignored.

encoding

Specify the encoding used to process the Web client request body.

Values that can be specified are those for which encoding is supported in Java, such as "SJIS" and "EUC_JP".

web-server

Select whether to use Web server authentication information. Select either of the options shown below. No distinction is made between upper and lower case.

- True (Web server authentication information is used)
- False (Web server authentication information is not used) (This is the default value)

Web Module Definition File Examples

Specifying only the Web Application Name

```
<?xml version="1.0" encoding="UTF-8"?>
<interstage-web-app>
  <war>
    <context-root>sampleWarModule</context-root>
  </war>
</interstage-web-app>
```

Specifying Items Other than the Web Application Name

```
<?xml version="1.0" encoding="UTF-8"?>
<interstage-web-app>
  <war>
```

```

<context-root>sampleWarModule</context-root>
<shared-context>True</shared-context>
<session>
  <cookie>True</cookie>
  <web-browser>True</web-browser>
  <cookie-security>Always</cookie-security>
</session>
<encoding>EUC_JP</encoding>
<authentication>
  <web-server>True</web-server>
</authentication>
</war>
</interstage-web-app>

```

2.4 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
- **Windows32/64**
Microsoft® Internet Information Services
- **Solaris32**
Sun Java System Web Server

2.4.1 Interstage HTTP Server Environment Settings

The method to configure the environment settings for linking IJServer and Interstage HTTP Server are explained in this section.

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 "2.4.5 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 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 edit it.

Windows32/64

```
LoadModule ihs2_redirector2_module "C:/Interstage/F3FMjs5/gateway/ ihs2/mod_ihs2_redirector2.so "
```

Solaris32/64 **Linux32/64**

```
LoadModule ihs2_redirector2_module "/opt/FJSVjs5/gateway/ ihs2/mod_ihs2_redirector2.so "
```

2.4.2 Microsoft® Internet Information Services 6.0 Environment Settings

Windows32/64

The method to configure the environment settings for linking IIServer and Microsoft® Internet Information Services 6.0 is explained in this section.

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 IIServer and the Web server on different machines, appropriate environment settings must be made in the Web server connector. For details, refer to "[2.4.5 Procedure for Operation by Separating IIServer 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 the Interstage HTTP Server with the Web server name "FJapache " 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 the Interstage HTTP Server with the Web server name "FJapache ", it is necessary to make the Interstage HTTP Server and Web server connector settings using the Interstage Management Console.

.....

Note

.....

On the same machine, the Interstage HTTP Server with the Web server name "FJapache " and Microsoft ® Internet Information Services cannot use a Web server connector of the Interstage HTTP Server with the Web server name "FJapache " simultaneously, although it is possible for them to exist together by setting up a different port number in each Web server.

Also, the Web server connector and JK2 connector cannot be used with Microsoft® Internet Information Services at the same time on the above machine.

.....

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. Restarting the IIS Admin Service Service or the Server Machine
4. Microsoft® Internet Information Services Environment Settings
5. 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 and Interstage.

When the Interstage HTTP Server with the Web server name "FJapache" is removed after installing the Interstage HTTP Server, Microsoft® Internet Information Services cannot be linked to. In this instance, the Interstage HTTP Server must be recreated with the name "FJapache".

.....

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] > [FJapache]. Click the [Status] tab, and then click the [Stop] button. This stops Interstage HTTP Server.

2. Prevent linkage of Interstage HTTP Server

The auto-activation of the Interstage HTTP Server is inhibited. For details, refer to "Setting the Auto-activation" of "Operation and Maintenance" in the Interstage HTTP Server Operator's Guide.

Note

To have the Interstage HTTP Server and Microsoft® Internet Information Services co-exist, when Interstage is reactivated, activate separately the Web servers except for the Interstage HTTP Server with the Web server name "FJapache".

Restarting the IIS Admin Service Service or the Server Machine

If the IIS 5.0 process dissociation mode is used in Microsoft® Internet Information Services 6.0, restart the IIS Admin Service service or the server machine.

- Using the IIS 5.0 isolation mode in Microsoft® Internet Information Services 6.0

Restart the IIS Admin Service service. To restart the service, log in as a user with Administrator authority. Click [Control Panel], and then [Services], or click [Control Panel] - [Administrative Tools] - [Services]. After startup, select "IIS Admin Service", and then select [Restart] from [Action].

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, perform the following using the IIS Internet Service Manager:

- 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

The Web server connector is set in the "ISAPI filter" on the Web site created under the console tree, [Web site].

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

```
C:\Interstage\F3FMjs5\gateway\isapi\isapi_redirector2.dll
```

Registering in an ISAPI Extension

The "virtual directory" is created on the Web site created under [Web site] on the console tree,.

The values that are set are shown below.

| | |
|--------------|-------------------------------------|
| Local path | C:\Interstage\F3FMjs5\gateway\isapi |
| Name (alias) | F3FMjs5 |

Allow permission to execute the ISAPI application using the access authority to the virtual directory.

Registering with the Web Service Extension

The files of the Web server connector set by registering in the ISAPI filter is added to the Web service extension, and the status is set as "Allowed".

Setting the Worker Process

The maximum worker process number is assigned with 1 and the worker process recycle is set invalid.

To invalidate the worker process recycle, cancel all the selection by clicking [Property] - [Recycle] of the application pool set in the virtual directory created by registering with the ISAPI extension.

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 run Interstage Application Server on Windows Server® x64 Editions (32-bit compatible), enable 32-bit mode by executing the following command at the command prompt:

```
cscript %SystemDrive%\inetpub\AdminScripts\adsutil.vbs set w3svc/AppPools/Enable32bitAppOnWin64 1
```

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 IIS Internet Service Manager.



Note

- To use SSL communication between the Web server connector and the Servlet container, the user who executes Microsoft® Internet Information Services 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 directory 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".

To access the Web Server connector log output directory and log file as a general user, assign "Full Control" permission.

- When creating the work unit, "FJapache" must be selected for the Web server. The Web server connector of Microsoft® Internet Information Services cannot be used if "FJapache" is not selected for the Web server or if the Web server is changed in the work unit environment settings from "FJapache".

2.4.3 Microsoft® Internet Information Services 7.0/7.5/8.0 Environment Settings

Windows32/64

The method to configure the environment settings for linking IIServer and Microsoft® Internet Information Services 7.0/7.5/8.0 are explained in this section.

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 IIServer and the Web server on different machines, appropriate environment settings must be made in the Web server connector. For details, refer to "[2.4.5 Procedure for Operation by Separating IIServer 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 the Interstage HTTP Server with the Web server name "FJapache " 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 the Interstage HTTP Server with the Web server name "FJapache ", it is necessary to make the Interstage HTTP Server and Web server connector settings using the Interstage Management Console.

.....

Note

.....

On the same machine, the Interstage HTTP Server with the Web server name "FJapache " and Microsoft® Internet Information Services cannot use a Web server connector of the Interstage HTTP Server with the Web server name "FJapache " simultaneously, although it is possible for them to exist together by setting up a different port number in each Web server. Also, the Web server connector and JK2 connector cannot be used with Microsoft® Internet Information Services at the same time on the above machine.

.....

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 and Interstage.

When the Interstage HTTP Server with the Web server name "FJapache" is removed after installing the Interstage HTTP Server, Microsoft® Internet Information Services cannot be linked to. In this instance, the Interstage HTTP Server must be recreated with the name "FJapache".

.....

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] > [FJapache]. Click the [Status] tab, and then click the [Stop] button. This stops Interstage HTTP Server.

2. Prevent linkage of Interstage HTTP Server

The auto-activation of the Interstage HTTP Server is inhibited. For details, refer to "Setting the Auto-activation" of "Operation and Maintenance" in the Interstage HTTP Server Operator's Guide.



To have the Interstage HTTP Server and Microsoft® Internet Information Services co-exist, when Interstage is reactivated, activate separately the Web servers except for the Interstage HTTP Server with the Web server name "FJapache".

Microsoft® Internet Information Services Environment Settings

Configure the settings for operating the Web server connector on Microsoft® Internet Information Services. Perform the following operations once you have logged on with Administrator permissions.



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

Installing ISAPI Extensions and ISAPI Filters

- If using Microsoft(R) Internet Information Services 7.0/7.5

Select [Server Manager] > [Roles] > [Web Server (IIS)]; in "Role Services", click [Add Role Services], then in "Application Development", select "ISAPI Extensions" and "ISAPI Filters", click [Next], and then click [Install].
- If using Microsoft(R) Internet Information Services 8.0

Select [Server Manager] > [Dashboard] > [Add roles and features] to start the Add Roles and Features Wizard.

From "Installation Type", select "Role-based or feature-based installation", and then click [Next].

From "Server Selection", click [Next].

From "Server Roles", select "Web Server (IIS)". Add features if required. Click [Next].

From "Web Server Role (IIS)", click [Next].

From "Role Services", select "ISAPI Extensions" and "ISAPI Filters" under "Application Development", click [Next], and then click [Install].

Stopping the Service

- If using Microsoft(R) Internet Information Services 7.0/7.5

Select [Server Manager] > [Roles] > [Web Server (IIS)]; in "System Services" stop "World Wide Web Publishing Service" and "Windows Process Activation Service".
- If using Microsoft(R) Internet Information Services 8.0

Select [Server Manager] > [Dashboard].

In "Services", stop "World Wide Web Publishing Service" and "Windows Process Activation Service".

Adding the ISAPI Filter

1. In the IIS manager, double-click [Site] > "Site Name" > "ISAPI Filter" to open the ISAPI Filter window.
2. In the operation pane, click "Add". In the "Add ISAPI Filter" dialog box that is displayed, enter the following values and then click the [OK] button.

| | |
|-------------|--------------------|
| Filter name | F3FMjs5 (optional) |
|-------------|--------------------|

| | |
|---------------------------|---|
| File that can be executed | C:\Interstage\F3FMjs5\gateway\isapi\isapi_redirector2.dll |
|---------------------------|---|

Adding the Application

1. In the IIS manager, right-click [Site] > "Site Name" > "Add Application".
2. In the "Add Application" dialog box that is displayed, enter the following values and then click the [OK] button.

| | |
|------------------|--|
| Alias | F3FMjs5 |
| Application Pool | DefaultAppPool or any application pool |
| Physical Path | C:\Interstage\F3FMjs5\gateway\isapi |

Note

The Web server connector executes the URL rewrite process using the ISAPI filter.

Also, in Microsoft® Internet Information Services, the settings must be configured so that both the original URL and the post-rewrite URL are executed in the same application pool.

For this reason, configure the settings according to one of the examples shown below.

- Using the default application pool (DefaultAppPool) in the default site (Default Web Site)

Set the default application pool (DefaultAppPool) in the default site (Default Web Site) and "F3FMjs5".

Example: Using the default application pool (DefaultAppPool) in the default site (Default Web Site)

```
IIS Manager
+[Application Pool]
|  +[DefaultAppPool]
+[Site]
  +[Default Web Site]:Set DefaultAppPool
    +[F3FMjs5]:Set DefaultAppPool
```

- Using any application pool in the default site (Default Web Site)

Set any application pool in the default site (Default Web Site) and "F3FMjs5".

Example: Using any application pool (IASPool) in the default site (Default Web Site)

```
IIS Manager
+[Application Pool]
|  +[DefaultAppPool]
|  +[IASPool]
+[Site]
  +[Default Web Site]:Set IASPool
    +[F3FMjs5]:Set IASPool
```

- Using any application pool in any site (IAS Web Site)

Set any application pool in any site (IAS Web Site) and "F3FMjs5".

Example: Using any application pool (IASPool) in any site (IAS Web Site)

```
IIS Manager
+[Application Pool]
|  +[DefaultAppPool]
|  +[IASPool]
+[Site]
  +[IAS Web Site]:Set IASPool
    +[F3FMjs5]: Set IASPool
```

3. In the IIS manager, double click [Site] > "Site Name" > [F3FMjs5] > "Handler Mapping" to open the handler mapping window.
4. Select "ISAPI-dll". In the operation pane, click "Edit".

5. In the "Edit Module Map" dialog box that is displayed, enter the following values and then click the [OK] button.

| | |
|---------------------------|---|
| Request path | *.dll |
| Module | IsapiModule |
| File that can be executed | C:\Interstage\F3FMjs5\gateway\isapi\isapi_redirector2.dll |

6. In the dialog box that is displayed next, click [Yes].

7. Select "ISAPI-dll" again. In the operation pane, click "Edit functionality access permissions".

8. In the "Edit functionality access permissions" dialog box that is displayed, select all the check boxes and then click the [OK] button.

Worker Process Settings

1. In [Application Pool] of the IIS manager, select "Name of application pool selected when the above application was added". In the operation pane, click "Recycle Settings".

2. In the "Edit application recycle settings" dialog box that is displayed, de-select all the check boxes and then click the [Next] button. In the next window, click the [Close] button without selecting anything.

3. Select the application pool that was selected in 1. again. In the operation pane, click "Detailed Settings".

4. In the "Detailed Settings" dialog box that is displayed, enter the following value and then click the [OK] button.

| | |
|------------------------------------|------|
| Maximum number of worker processes | 1 |
| Validate 32-bit applications (*1) | True |

*1 This is only set when Interstage Application Server is run on Windows Server® x64 Editions (32-bit compatible).

Starting the Service

- If using Microsoft(R) Internet Information Services 7.0/7.5

Select [Server Manager] > [Roles] > [Web Server (IIS)]; in "System Services" start "World Wide Web Publishing Service" and "Windows Process Activation Service".

- If using Microsoft(R) Internet Information Services 8.0

Select [Server Manager] > [Dashboard].

In "Services", start "World Wide Web Publishing Service" and "Windows Process Activation Service".

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 IIS 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 Services 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 directory 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".

- Assign access permissions that enable full control in general user permissions for the Web server connector log output directory and Web server connector log file.
- When creating the work unit, "FJapache" must be selected for the Web server. The Web server connector of Microsoft® Internet Information Services cannot be used if "FJapache" is not selected for the Web server or if the Web server is changed in the work unit environment settings from "FJapache".

2.4.4 Sun Java System Web Server Environment Settings

Solaris32

The method to configure the environment settings for linking IJServer and Sun Java System Web Server are explained in this section.

The Web server connector of Sun Java System Web Server 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 "2.4.5 Procedure for Operation by Separating IJServer and Web Server".

Point

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

Note

On the same machine, the Interstage HTTP Server with the Web server name "FJapache" and Sun Java System Web Service cannot use a Web server connector of the Interstage HTTP Server with the Web server name "FJapache" simultaneously, although it is possible for them to exist together by setting up a different port number in each Web server.

Also, the Web server connector and JK2 connector cannot be used with Sun Java System WebServer at the same time on the above machine.

Link Interstage and Sun Java System Web Server 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.

When the Interstage HTTP Server with the Web server name "FJapache" is removed after installing the Interstage HTTP Server, Sun Java System Web Server cannot be linked to. In this instance, the Interstage HTTP Server must be recreated with the name "FJapache".

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 HTTP Server Operator's Guide.

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] > [FJapache]. Click the [Status] tab, and then click the [Stop] button. This stops Interstage HTTP Server.

2. Prevent linkage with Interstage HTTP Server

The auto-activation of the Interstage HTTP Server is inhibited. For details, refer to "Setting the Auto-activation" of "Operation and Maintenance" in the Interstage HTTP Server Operator's Guide.

Note

To have the Interstage HTTP Server and Sun Java System Web Server co-exist, when Interstage is reactivated, activate separately the Web servers except for the Interstage HTTP Server with the Web server name "FJapache".

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.



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
```



Example

If the number of Servlet containers that can be processed at the same time is set as 64, set the value of MaxProcs x 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/FJSVjs5/gateway/nsapi/ijs_nsapi_redirector.so"
Init fn="ijs_nsapi_init" conf="/opt/FJSVjs5/conf/jk2/FJapache/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.

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".

- When creating the work unit, "FJapache" must be selected for the Web server. The Web server connector of the Sun Java System Web Server cannot be used if "FJapache" is not selected for the Web server or the Web server is changed from "FJapache" in the work unit environment settings.

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

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.

The *isj2eeadmin* command can also be used to implement these settings. For details, refer to the Reference Manual (Command Edition).

- 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 for accepting requests], and specify all the IP addresses that are to be set in the Web server.

The *isj2eeadmin* command can also be used to implement these settings. For details, refer to the Reference Manual (Command Edition).

- Even if IJServer and the Web server are running on the same server machine, to link them with IJServer and the Web server on another server machine, it is necessary to specify separate settings for each program as described in this section, so that they run on separate server machines.

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 or the *isj2eeadmin* command.

2. Setting connection destination IJServer

Using the Interstage Management Console on the Web server machine or the *isj2eeadmin* command, 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 or the *isj2eeadmin* command, 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.

In the description below, the Interstage Management Console is used. This description also applies to the *isj2eeadmin* command.

- 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

In the Interstage Management Console of the Web server machine, specify [Web Server] > [Web Server name] > [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 name] > [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 name] > [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 name] > [Web Server Connector] > [Create New]
- [Services]> [Web Server] > [Web Server name] > [Web Server Connector] > 'WorkUnit Name' > [Environment Settings]

The *isj2eeadmin* command can also be used to implement these settings. For details, refer to the Reference Manual (Command Edition).

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.

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].

In the procedure described below, the Interstage Management Console is used. This description also applies to the *isj2eeadmin* command. For details, refer to the Reference Manual (Command Edition).

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

[Requirements]

| | |
|--|----------------------------|
| Server machine A (IP Address: 192.0.2.1) | Web server must be running |
| Server machine B (IP Address: 192.0.2.2) | IJServer must be running |

[Setting procedure]

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 |
| Web server IP address for accepting requests | 192.0.2.1 |
| IP address for Servlet container | 192.0.2.2 |
| Port number | 9000 |

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 name] > [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 | 192.0.2.2:9000 |

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.

- 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 name] > [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: 192.0.2.1) | Web server must be running |
| Server machine B (IP Address: 192.0.2.2) | Web server must be running |
| Server machine C (IP Address: 192.0.2.3) | IJServer must be running |

[Setting procedure]

- 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 |
| Web server IP address for accepting requests | 192.0.2.1 192.0.2.2 |
| IP address for Servlet container | 192.0.2.3 |
| Port number | 9000 |

- Setup of server machine A for the Web server.

From the Interstage Management Console on server machine A, select [Services] > [Web Server] > [Web Server name] > [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 | 192.0.2.2:9000 |

- 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.

- 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.

- 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 name] > [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.

- 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: 192.0.2.1) | Web server must be running |
|--|----------------------------|

| | |
|--|--------------------------|
| Server machine B (IP Address: 192.0.2.2) | IJServer must be running |
| Server machine C (IP Address: 192.0.2.3) | IJServer must be running |

[Setting procedure]

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 |
| Web server IP address for accepting requests | 192.0.2.1 |
| IP address for Servlet container | 192.0.2.2 |
| Port number | 9000 |

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 192.0.2.3 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 |
| Web server IP address for accepting requests | 192.0.2.1 |
| IP address for Servlet container | 192.0.2.3 |
| Port number | 9000 |

- [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 |
| Web server IP address for accepting requests | 192.0.2.1 |
| IP address | 192.0.2.3 |
| Port number | 9000 |

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 name] > [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 | 192.0.2.2:9000 192.0.2.3:9000 |

- [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 | 192.0.2.2:9000 |

<Connection destination information separately for server machines C>

| Item Name | Setting Value Example |
|--|-----------------------|
| WorkUnit name | MyIJSERVER_2 |
| IP address for Servlet container : Port number | 192.0.2.3:9000 |

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 name] > [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: 192.0.2.1) | Load balancing is running |
| Server machine B (IP Address: 192.0.2.2) | Web server is running |
| Server machine C (IP Address: 192.0.2.3) | Web server is running |
| Server machine D (IP Address: 192.0.2.4) | IJSERVER is running |
| Server machine E (IP Address: 192.0.2.5) | IJSERVER is running |

[Setting procedure]

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 |
| Web server IP address for accepting requests | 192.0.2.2 192.0.2.3 |
| IP address for Servlet container | 192.0.2.4 |
| Port number | 9000 |

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 192.0.2.5 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 name] > [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 | 192.0.2.4:9000 192.0.2.5:9000 |

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 name] > [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.

2.5 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.

2.5.1 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 the IJServer WorkUnit with the lowest number of requests currently being processed. 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.

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.

```
ijsdspatchcont 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 a and container b) 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.

```
ijsdspatchcont OFF 123.123.123.110:9000  
ijsdspatchcont OFF 123.123.123.110:9001
```

ap101, which was distributed to containers a and b of machines 1 and 2, now runs on containers a and b 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.

```
ijsdspatchcont ON 123.123.123.110:9000  
ijsdspatchcont 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.

```
ijsdspatchcont 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.

```
ijsdspatchcont ON 123.123.123.110:9000
```

2.5.2 Monitoring Web Server Connector Faults



This function can be used with the following products:

- Interstage Application Server Enterprise Edition
- Interstage Application Server Standard-J Edition
- Web Package included in the Interstage Application Server Enterprise Edition

This function cannot be used in the Web Package included in Standard-J Edition.

The Web Package is not provided for 64-bit products.

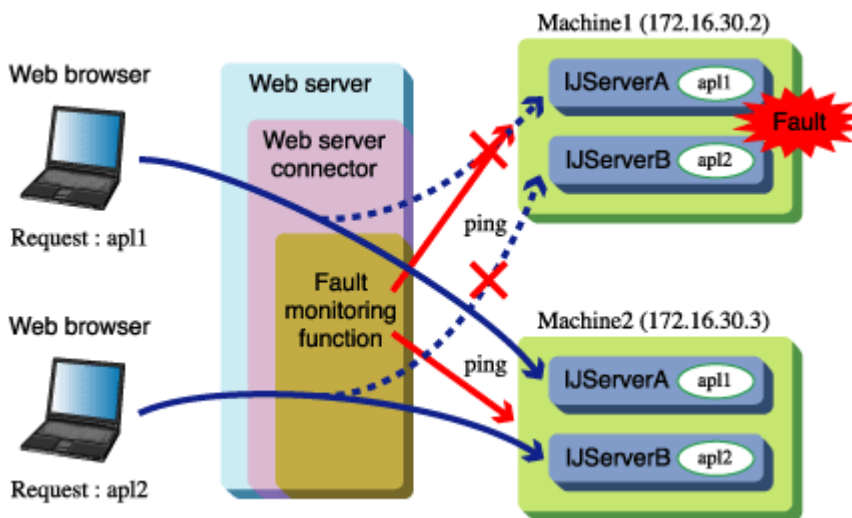
When a balancing operation is executed for IJServer and the Web server that have been separated on different server machines, you can monitor the operation status of the IJServer machine and the Servlet container to stop an IJServer machine or a faulty Servlet container from being distributed automatically,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

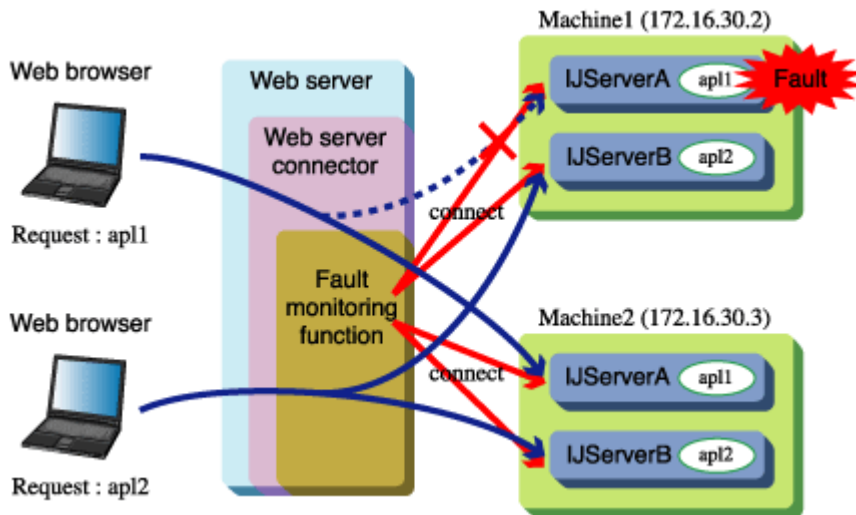


For example, as shown in the figure above, the fault monitoring function issues regular (such as every 60 seconds) pings to the IP addresses of machine 1 and machine 2 and monitors whether there is a response. If the ping response stops because of a fault in the hardware of machine 1, machine 1 is stopped 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. The *isj2eadmin* command can also be used to implement these settings. For details, refer to the Reference Manual (Command Edition).

Settings Items

To make the settings for the fault monitoring function, click the [Services] > [Web Server] > [Web Server name] > [Web Server Connector] > [Fault Monitoring Settings] tabs. The *isj2eadmin* command can also be used to implement these settings. For details, refer to the Reference Manual (Command Edition).

The settings items are as follows:

| Item name | Meaning |
|---------------------|---|
| Monitoring Method | Select one of the following monitoring methods: <ul style="list-style-type: none"> - 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. |

| Item name | Meaning |
|-------------------|---|
| | 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

In the procedure described below, the Interstage Management Console is used. This description also applies to the *isj2eadmin* command. For details, refer to the Reference Manual (Command Edition).

Monitoring the Operation Status of two IJServer Machines in a Balancing Configuration

[Requirements]

| | |
|--|----------------------------|
| Server machine A (IP Address: 192.0.2.1) | Web server must be running |
| Server machine B (IP Address: 192.0.2.2) | IJServer must be running |
| Server machine C (IP Address: 192.0.2.3) | 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 |
| Create the IJServer of V8.0 compatible mode | No |
| Web server IP address for accepting requests | 192.0.2.1 |
| IP address for Servlet container | 192.0.2.2 |
| Port number | 9000 |

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 |
| Create the IJServer of V8.0 compatible mode | No |
| Web server IP address for accepting requests | 192.0.2.1 |
| IP address for Servlet container | 192.0.2.3 |
| Port number | 9000 |

- Using the Interstage Management Console for server machine A, click the [Services] > [Web Server] > [Web Server name] > [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 | 192.0.2.2:9000 192.0.2.3:9000 |

- Using the Interstage Management Console for server machine A, click the [Services] > [Web Server] > [Web Server name] > [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 |

- Using the Interstage Management Console for server machine A, click the [Services] > [Web Server] [Web Server name] > [status] 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: 192.0.2.1) | Web server must be running |
| Server machine B (IP Address: 192.0.2.2) | IJSERVER must be running |
| Server machine C (IP Address: 192.0.2.3) | IJSERVER must be running |

[Setting Procedure]

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

| Item Name | Setting Value Example |
|--|-----------------------|
| WorkUnit name | MyIJSERVER1 |
| Create the IJSERVER of V8.0 compatible mode | No |
| Web server IP address for accepting requests | 192.0.2.1 |
| IP address for Servlet container | 192.0.2.2 |
| Port number | 9000 |

| Item Name | Setting Value Example |
|--|-----------------------|
| WorkUnit name | MyIJSERVER2 |
| Create the IJSERVER of V8.0 compatible mode | No |
| Web server IP address for accepting requests | 192.0.2.1 |
| IP address for Servlet container | 192.0.2.2 |
| Port number | 9001 |

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

| Item Name | Setting Value Example |
|--|-----------------------|
| WorkUnit name | MyIIServer1 |
| Create the IIServer of V8.0 compatible mode | No |
| Web server IP address for accepting requests | 192.0.2.1 |
| IP address for Servlet container | 192.0.2.3 |
| Port number | 9000 |

| Item Name | Setting Value Example |
|--|-----------------------|
| WorkUnit name | MyIIServer2 |
| Create the IIServer of V8.0 compatible mode | No |
| Web server IP address for accepting requests | 192.0.2.1 |
| IP address for Servlet container | 192.0.2.3 |
| Port number | 9001 |

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

| Item Name | Setting Value Example |
|--|----------------------------------|
| WorkUnit name | MyIIServer1 |
| IP address for Servlet container : Port number | 192.0.2.2:9000 192.0.2.3:9000 |

| Item Name | Setting Value Example |
|--|----------------------------------|
| WorkUnit name | MyIIServer2 |
| IP address for Servlet container : Port number | 192.0.2.2:9001 192.0.2.3:9001 |

4. Using the Interstage Management Console for server machine A, click the [Services] > [Web Server] > [Web Server name] > [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] [Web Server name] > [status] 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 IIServer in the Web server machine. Refer to the Reference Manual (Command Edition) for details of the command.

```
svmondspstat
```

Viewing the Status of all IIServers

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

Conditions:

- Distributing is made to 2 IIServers, the IIServer names of which are MyIIServer1 and MyIIServer2
- The IP address and port number of the distributing destination Servlet container are as follows:
 192.0.2.2:9000
 192.0.2.3:9000
 192.0.2.2:9001
 192.0.2.3:9001
- The fault monitoring method is port monitoring
- Only 192.0.2.2:9001 in MyIIServer2 is faulty

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

| Status | IP Address | Port Number | WorkUnit Name | WebServer Name |
|--------|------------|-------------|---------------|----------------|
| ACTIVE | 192.0.2.2 | 9000 | :MyIIServer1 | : FJapache |
| ACTIVE | 192.0.2.3 | 9000 | :MyIIServer1 | : WEB001 |
| DOWN | 192.0.2.2 | 9001 | :MyIIServer2 | : FJapache |
| ACTIVE | 192.0.2.3 | 9001 | :MyIIServer2 | : WEB001 |

Viewing the Status of a Specific IIServer

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 IIServer.

Example

```
svmondspstat -i MyIIServer1
```

Conditions:

- Distributing is made to 2 IIServers, the IIServer names of which are MyIIServer1 and MyIIServer2
- The IP address and port number of the distributing destination Servlet container are as follows:
 192.0.2.2:9000
 192.0.2.3:9000
 192.0.2.2:9001
 192.0.2.3:9001
- The fault monitoring method is port monitoring
- All the Servlet containers in MyIIServer1 are under operation

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

| Status | IP Address | Port Number | WorkUnit Name | WebServer Name |
|--------|------------|-------------|---------------|----------------|
| ACTIVE | 192.0.2.2 | 9000 | :MyIIServer1 | : FJapache |
| ACTIVE | 192.0.2.3 | 9000 | :MyIIServer1 | : WEB001 |



Note the following when using the fault monitoring function:

- All the following IIServer clusters are monitored according to the conditions set for Fault Monitoring Settings. It is not possible to monitor according to conditions set for a separate IIServer cluster.
 - Distribution destinations for Web Server Connector (for Interstage HTTP Server)
 Configure with the "Servlet container IP address: Port number" entry in the Interstage Management Console

- Distribution destinations for Web Server Connector (for Interstage HTTP Server 2.2)

Configure with the wscadmin command --add-instance-ref subcommand

For details about Web Server Connector (for Interstage HTTP Server 2.2), refer to the Java EE Operator's Guide (Java EE 6 Edition).

- 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 IPCOM 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].

- If a fault occurs on a server, then a 'fault-detected' message followed immediately by a second message notifying the user that recovery has started, may be output when:

- fault monitoring is active
- a fault monitoring condition changed, or
- a server targeted for monitoring is dynamically added.



2.6 Procedure for Using JTS Windows32/64 Solaris32 Linux32/64

This section explains the procedure for using JTS.

2.6.1 Flow to Operation Start Windows32/64 Solaris32 Linux32/64

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 Application Server Operator's Guide.

For details on the distributed transaction function and JTA interface distributed by JTS, refer to "Part V, 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



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.



When an Oracle database is used

ojdbc6.jar

- Resource adapter class library

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



Notes on setting resource manager environment

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

```
sqlplus "sys/password@ORACLE_SID AS SYSDBA"
```

2. Execute the following SQL.

```
grant select on DBA_PENDING_TRANSACTIONS to username(*1)
```

*1 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 and to use the JDBC data source in the resource environment settings, select the "Use distributed transaction", and, to use the connector, select "Yes" for the "Use for global transactions", and apply.

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 distributed 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 to use the DBC data source in the resource environment settings, check the "Use distributed transaction", and, to use the connector, select "Yes" for the "Use for global transactions", and apply.



Note

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

With Symfoware or SQL Server, 'Use distributed 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.

Windows32/64

```
%SystemRoot%\system32\drivers\etc\hosts
```

Solaris32 **Linux32/64**

```
/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

Example of server URL

```
Datasource1 Server URL:jdbc:oracle:thin:@oracle_db1:1521:ORCL
Datasource1 Server URL:jdbc:oracle:thin:@oracle_db2:1521:ORCL
```

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 `otsstarttrsc` command to start the JTS resource management program.

6. Starting the Application

Start the WorkUnit using the Interstage Management Console.

2.6.2 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 setup

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

2.7 Procedure for Using JMS

This section explains the procedure for using JMS.

2.7.1 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 'Installation (Server Package)' 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 "[19.1 Environment Settings for the Event Channel Operation Machine before Operation](#)" in "Environment Settings for Interstage JMS" 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 "[19.3 Environment Settings for the JMS Application Operation Machine before Operation](#)" in "Environment Settings for Interstage JMS" 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.



.....
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'.Description (if necessary)
.....

4. Operation Start of the JMS Application Operation Machine

Start the JMS application by using directly the java commands.

2.7.2 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 "[19.4 Environment Deletion for the JMS Application Operation Machine after Operation](#)" in "Environment Settings for Interstage JMS" 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 "[19.2 Environment Deletion for the Event Channel Operation Machine after Operation](#)" in "Environment Settings for Interstage JMS" for details.

2.7.3 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. <ul style="list-style-type: none">- 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 <ul style="list-style-type: none">- 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 |
| 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. <ul style="list-style-type: none">- 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. (*1) |
| 9 | SupplierCount | Number of publishers or senders that can be connected to the event channels. |

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

- Point-To-Point messaging model

Windows32/64

```
Number of connected consumers = 1
```

Solaris32/64 Linux32/64

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 (for unit control) 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 (for unit control) 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 |

2.8 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:

- [2.8.1 Mail Sending Application](#)
- [2.8.2 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 the "JNDI" chapter.

After Operation

1. Stopping the Mail Server

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

2.8.1 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) { }
```

2.8.2 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) { }
```

2.9 Customizing and Checking the Operating Environment

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

Windows32/64

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.

2.9.1 Customizing the Operating Environment

Setting for using the Fujitsu XML Processor Windows32/64 Solaris32 Linux32/64

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:

Windows32/64

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.

Solaris32 Linux32/64

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 "[1.2.5 Settings of XML Parser](#)" in "Design of J2EE Application", and customize.

[For J2EE application Client]

Set the following environment variable before the setting of the 'isj2ee.jar' path.

Windows32/64

| Environment variable | Setting Value |
|----------------------|--|
| CLASSPATH | %ProgramFiles%\Common Files\FujitsuXML\xmlpro.jar %ProgramFiles%\Common Files\FujitsuXML\xmltrans.jar |

Solaris32 Linux32/64

| Environment variable | Setting Value |
|----------------------|-------------------------------|
| CLASSPATH | /opt/FJSVxmlpc/lib/xmlpro.jar |

2.9.2 Checking the Operating Environment

Setting the Environment Variable

In environment variable CLASSPATH, set the following value if it has not already been set.

Windows32/64

C:\Interstage\J2EE\lib\isj2ee.jar

Solaris32/64 Linux32/64

/opt/FJSVj2ee/lib/isj2ee.jar

Environment Setup of Java

(1) Installing Java

When the Interstage server package is installed

- JDK6 is installed when J2EE is installed.

(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.

Windows32/64

When applying in JDK environment: C:\Interstage\JDK6\jre\bin

When applying in JRE environment: C:\Interstage\JRE6\bin

Solaris32/64 Linux32/64

When applying in JDK environment: /opt/FJSVawjbc/jdk6/jre/bin

When applying in JRE environment: /opt/FJSVawjbc/jre6/bin



Note

- Depending on the used shell, 'path' must be used as the environment variable instead of 'PATH'. Make the settings according to the environment.
- 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

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 or Linux custom installation
- If Java is installed additionally using a Solaris or Linux custom installation

The Java environment setup file is allocated to the following directory:

Windows32/64

```
C:\Interstage\J2EE\etc\java_config.txt
```

Solaris32/64 Linux32/64

```
/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 JDK6: JDK60DIR
- When using JRE6: JRE60DIR

- Java installation directory

Specify the absolute path of the Java installation directory.



Example

Windows32/64

Example of statement for using JDK6 installed in C:\Interstage\jdk6:

```
JDK60DIR = C:\Interstage\jdk6
```

Solaris32/64 Linux32/64

Example of statement for using JDK6 installed in /opt/FJSVawjkb/jdk

```
JDK60DIR = /opt/FJSVawjkb/jdk6
```



Note

- Comments cannot be added to the Java environment settings file (java_config). Statements with a prefix such as "#" or "!" will be treated as invalid.

Windows32/64

- 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 JDK60DIR for 'Java version used' and use JDK as the Java environment.

Solaris32/64 Linux32/64

- 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 JDK60DIR is specified for 'Java version used' in the Java environment setup file, do not set JRE under control of JDK6 for 'Java installation directory.' Instead, set the directory in which JDK6 is installed.

Settings for Use of the EJB Service Run Command

Before the EJB service run command can be used, the EJB Java environment settings file 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 or Linux
- Java is installed as an add-on

The Java environment settings file is created in the following directory.

Windows32/64

```
C:\Interstage\EJB\etc\java_config.txt
```

Solaris32/64 Linux32/64

```
/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 JDK6: JDK60DIR
 - When using JRE6: JRE60DIR
- Java installation directory
Specify the absolute path of the Java installation directory.



Example

Windows32/64

Example of statement for using JDK6 installed in C:\Interstage\jdk6

```
JDK60DIR = C:\Interstage\jdk6
```

Solaris32/64 Linux32/64

Example of statement for using JDK6 installed in /opt/FJSVawjkb/jdk6

```
JDK60DIR = /opt/FJSVawjkb/jdk6
```

Note

- Comments cannot be added to the Java environment settings file (java_config). Statements with a prefix such as "#" or "!" will be treated as invalid.

Windows32/64

- 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 JDK60DIR for 'Java version used' and use JDK as the Java environment.

Solaris32/64 Linux32/64

- 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 JDK60DIR is specified for 'Java version used' in the Java environment setup file, do not set JRE under control of JDK6.0 for 'Java installation directory.' Instead, set the directory in which JDK6.0 is installed.

2.10 Debugging Applications

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 "Design of J2EE Applications" for the log output location.

- Container (container.log)
 - Standard output or standard error output of application
 - Output of log method of GenericServlet class
 - Output of log method of ServletContext class
 - EJB container log or Servlet container log
 - EJB container or Servlet container error message
 - EJB snap output
 - Web service log
- Container information log (info.log)
 - JavaVM process start information (ARGV, ENV)
 - JavaVM process start error message
 - Thread dump
 - Messages that cannot be output to the container log

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 Interstage Studio, 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.

2.10.1 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

The following table shows the types of information that are output to Snap.

Information Output to Snap

| Type | Output information |
|---|---|
| Method information of EJB application invoked by a Client | <p>Output the following types of method information of the EJB application that is invoked by a Client application:</p> <p>Method invocation information</p> <p>Method return information</p> <p>Method exception information</p> <p>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.</p> <p>In addition, only when the following method is invoked, the information is output.</p> <p>[Session Bean]</p> <p>Home interface method</p> <p>create</p> <p>remove(handle)</p> <p>remove(primarykey)</p> <p>Remote interface method</p> <p>business method</p> <p>remove</p> <p>LocalHome interface method</p> <p>business method</p> |

| Type | Output information |
|---|--|
| | <p>remove</p> <p>Local interface method</p> <p>business method</p> <p>remove</p> <p>[Entity Bean]</p> <p>Home interface method</p> <p>create</p> <p>remove(handle)</p> <p>remove(primarykey)</p> <p>findByPrimaryKey</p> <p>find <Enumeration type></p> <p>find <Collection type>, find<Object></p> <p>ejbHome method</p> <p>Remote interface method</p> <p>business method</p> <p>remove</p> <p>LocalHome interface method</p> <p>create</p> <p>remove(primarykey)</p> <p>findByPrimaryKey</p> <p>find <Enumeration type></p> <p>Local interface method</p> <p>business method</p> <p>remove</p> <p>Refer to "2.10.1.1 Method Information of EJB Application Invoked by a Client" for details of output information, formats, and examples.</p> |
| EJB application method information | <p>The following types of information are output:</p> <p>Method invocation information</p> <p>Method return information</p> <p>Method exception information</p> <p>For details of output information, formats, and examples, refer to "2.10.1.2 EJB Application Method Information".</p> |
| javax.transaction.UserTransaction API information | <p>The following types of information are output:</p> <p>Method invocation information</p> <p>Method return information</p> <p>Method exception information</p> <p>For details of output information, formats, and examples, refer to "2.10.1.3 javax.transaction.UserTransaction API Information".</p> |
| Database manipulation statement information | <p>The following types of database manipulation information stored in the Container are output:</p> |

| Type | Output information |
|---|---|
| (Only when the Entity Bean mode is CMP) | Database manipulation invocation information Database manipulation return information Database manipulation exception information For details of output information, formats, and examples, refer to " 2.10.1.4 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 " 2.10.1.5 EJB Container Transaction Control Information ". |
| J2EE application user debug information | This outputs the debug information that the J2EE application outputs. Refer to " 2.10.2 Using Application Debugging Information ". |

As shown in following table, the output information varies depending on the output level specified when the IIServer starts.

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 IIServer that collects snapshots.

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.



Note

- When the rapid invocation function is used, Snap information output by every J2EE application deployed in IIServer is stored in container log.
- If J2EE applications run in thread multiplex mode, all Snap information is output to the container log. 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.

2.10.1.1 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 following table.

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. | Yes | Yes |
| Time | The time the method was invoked or returned is indicated in 'hour: minute: second. millisecond' format. | Yes | Yes |

| Output item | Output information | Output level | |
|---|--|--------------|---------|
| | | Level 1 | Level 2 |
| 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. | Yes | Yes |
| Bean name | The name of the EJB application that invoked the method is indicated. | Yes | Yes |
| Method name | The name of an invoked method is indicated. | Yes | Yes |
| 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. | Yes | Yes |
| 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. | No | Yes |
| 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. | No | Yes |
| 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. | No | Yes |

Yes: Item output at the specified output level

No: 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 09:49:20.159 : 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
```

2.10.1.2 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 following table.

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. | Yes | Yes |
| Time | The time the method was invoked or returned is indicated in 'hour: minute: second. millisecond' format. | Yes | Yes |
| 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. | Yes | Yes |
| Bean name | The name of the EJB application that invoked the method is indicated. | Yes | Yes |
| Method name | The name of an invoked method is indicated. | Yes | Yes |
| 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. | Yes | Yes |
| 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. | No | Yes |

| 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. | No | Yes |
| 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. | No | Yes |
| 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. | No | Yes |

Yes: Item output at the specified output level

No: 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

```

2.10.1.3 javax.transaction.UserTransaction API Information

javax.transaction.UserTransaction API information is output when the javax.transaction.UserTransaction method is used from a 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 the table below.

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. | Yes | Yes |
| Time | The time the method was invoked or returned is indicated in 'hour: minute: second. millisecond' format. | Yes | Yes |
| 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. | Yes | Yes |
| Method name | The name of an invoked method is indicated. | Yes | Yes |
| 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. | Yes | Yes |
| Parameter information (Param) | Parameter information for a method is indicated in the '(type) parameter' format. | No | Yes |
| Return value information (ReturnValue) | Method return value information is indicated in the '(type) return value' format. | No | Yes |
| 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 | No | Yes |

Yes: Item output at the specified output level

No: 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
```

2.10.1.4 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 the table below.

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. | No | Yes |
| Time | The time the database manipulation statement was invoked or returned is indicated in 'hour: minute: second. millisecond' format. | No | Yes |
| 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. | No | Yes |
| Class name | The class name used in execution of the database manipulation statement is indicated. | No | Yes |
| Method name | The method name used in execution of the database manipulation statement is indicated. | No | Yes |
| 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. | No | Yes |
| Parameter information (Param) | The SQL statement used in execution of a database manipulation statement is output in the following format: (Type) parameter | No | Yes |
| 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 | No | Yes |
| 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 | No | Yes |

Yes: Item output at the specified output level

No: 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
Return Value              :(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
```

2.10.1.5 EJB Container Transaction Control Information

If the EJB application transaction type is Container and the transaction attribute is Required or RequiresNew, an EJB container uses a method of the `javax.transaction.TransactionManager` interface.

The following information is output as container transaction control information, when an 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
Return Value : 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 the table below:

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. | Yes | Yes |
| Time | The time the transaction control was starts or ends is indicated in the 'hour:minute: second. millisecond' format. | Yes | Yes |
| 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. | Yes | Yes |
| Method name | The name of an invoked method is indicated. | Yes | Yes |
| 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. | Yes | Yes |
| Parameter information (Param) | Parameter information for a method is indicated in the '(type) parameter' format. | No | Yes |
| Return value information (ReturnValue) | Method return value information is indicated in the '(type) return value' format. | No | Yes |
| 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 | No | Yes |

Yes: Item output at the specified output level

No: 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
8/10/2000 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
```

2.10.1.6 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 "[2.10.1.7 Log Output Method for Support](#)". This chapter explains the procedure of writing user debug information to a container log, 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 container log 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 java.util.logging.*; // 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.ejb.CreateException,
    javax.ejb.RemoteException {
    if(isDebug) { // Debug Mode ?
        // Output the snap log
        logger.log (Level.FINE, "ejbCreate START"); // -----3
    }
}

...

```

Inhibiting the log output except for the user debug information

When the user debug information of the J2EE applications (when -DFJSNAP=10, -DFJSNAP=11 or -DFJSNAP=12 is specified in the IJServer Java VM option) is output, information logged using the java.util.logging package provided by JDK of modules operating in the IJServer process or other manufacturers' modules (JDBC driver) is output to the container log.

Define as follows to prevent outputting of the log other than the J2EE application user debug information:

| | |
|-----------------------------------|--|
| Definition file storage directory | <div style="border: 1px solid black; padding: 2px; display: inline-block;">Windows32/64</div> C:\Interstage\EJB\etc <div style="border: 1px solid black; padding: 2px; display: inline-block;">Solaris32/64 Linux32/64</div> /opt/FJSVejb/etc |
| Definition file name | FJlogging.properties |
| Definition to change (*1) | .level=OFF |
| Definition to add (*2) | "Logger name"=ALL Example of definition: (when the logger name is UseLoggerCMP) UseLoggerCMP.level=ALL |

*1 The initial setting value of .level is ALL. Change it to OFF.

*2 Specify the logger name used by the J2EE application at the time of the logger creation.

Refer to the open documents and confirm the logger name before adding the definition when the open source utility module (i.e. Commons-logging) and/or framework used by a J2EE application or other manufacturer's product that is operating in the IJServer process (JDBC driver etc.) is to obtain the log information output using the java.util.logging package.

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 "2.10.1.7 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:

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_EJBL 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
```



Note

- There is no need to describe any description in the program to output the snap, 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 JServer is activated, debug information on the J2EE application is never sent to a container log. 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.
- When J2EE application user debug information is obtained, the debug information generated by Interstage is also output. This does not affect the behavior, however.

2.10.1.7 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.

Log Output Method for Support

| Support method | Output debug information |
|----------------------------------|---|
| <code>config(String msg)</code> | Log level = '[CONFIG]' Message = String specified for msg |
| <code>fine(String msg)</code> | Log level = '[FINE]' Message = String specified for msg |
| <code>finer(String msg)</code> | Log level = '[FINER]' Message = String specified for msg |
| <code>finest(String msg)</code> | Log level = '[FINEST]' Message = String specified for msg |
| <code>info(String msg)</code> | Log level = '[INFO]' Message = String specified for msg |
| <code>severe(String msg)</code> | Log level = '[SEVERE]' Message = String specified for msg |
| <code>warning(String msg)</code> | Log level = '[WARNING]' Message = String specified for msg |

| Support method | Output debug information |
|---|--|
| 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 |
| 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 |
| logrb(Level level, String sourceClass, String sourceMethod, String bundleName, | Log level = String for the level Message = String specified for msg Parameter = String specified for params |

| Support method | Output debug information |
|--|---|
| String msg, Object[] 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 |

2.10.2 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(R) Windows(R) Explorer or an alternative file management tool to delete standard output and standard error output files that are no longer required.
- When using SQL Server, refer to "[JDBC Driver Logging Function](#)" under "Environment Setup when SQL Server is Used".

2.10.3 Using the Debugger

This method employs the debugger provided by Interstage Studio.

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 Interstage Studio User's Guide.



Note

- When IJServer is debugged, two or more processes cannot run concurrently. Be sure to set 1 for 'Process concurrency' of the WorkUnit.
- Multiple IJServers set in debug start mode cannot be started at the same time.

2.10.4 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 when IJServer starts up

There may be a problem with the processing that runs when the class or servlet init method is executed when IJServer starts up, or the processing may take a while. 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 when IJServer is stopped by force

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.

The timeout value can be set at 'Process forced stop time'.

2.10.5 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 Setup when JNDI Service Provider is Used

In Interstage, a JNDI service provider to enable access to J2EE objects is implemented. Refer to "[3.1 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

The JNDI service provider implemented by Interstage enables J2EE applications to use JNDI interfaces to refer to the following object types:

- EJB Home objects registered in the naming service
- EJB Local Home objects of EJB applications that run on the same process
- Various J2EE resources configured on your own PC (such as JDBC data source)
- Environment entries described in the application deployment descriptor
- Fixed-name specific objects (UserTransaction and ORB)

Objects that can be referenced in JNDI are listed in the following table.

| Category | Object name | Web application | EJB application | J2EE application client | Applet |
|--------------------------------|---------------------------------|-----------------|-----------------|-------------------------|--------|
| EJB object | EJB Home object | Yes | Yes | Yes | Yes |
| | EJB Local Home object | Yes | Yes | No | No |
| Resource reference | JDBC data source | Yes | Yes | Yes | No |
| | JMS connection factory | Yes | Yes | Yes | No |
| | JavaMail mail session | Yes | Yes | Yes | No |
| | URL (Uniform Resource Locator) | Yes | Yes | Yes | No |
| | connector connection factory | Yes | Yes | No | No |
| Resource environment reference | JMS Destination | Yes | Yes | Yes | No |
| others | Environment entry | Yes | Yes | Yes | No |
| | UserTransaction | Yes | Yes | Yes | No |
| | ORB | Yes | Yes | Yes | Yes |

Writing to "deployment descriptor"

Write the information of objects to be referenced to the deployment descriptor file of each application. For details, refer to "[3.9 Description in 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 "[3.10 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 "[3.11 Name Conversion Function](#)" for details of the name conversion function.

Setting when Fujitsu XML Processor is Used Windows32/64 Solaris32 Linux32/64

For usual analysis of the deployment descriptor file or name conversion file, use Oracle'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 the "Operating J2EE Applications" chapter.

Using UserTransaction

The transaction start and end can be controlled by using a looked up UserTransaction object. For details, refer to "[3.12 Transaction Function Using the UserTransaction Interface](#)".

3.1 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.

- 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 "[3.1.1 J2EE Application Client](#)".

- Applet

Refer to "[3.1.2 Applet](#)" for details of the setting required for Applet.

3.1.1 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.

1. 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.

Windows32/64

```
C:\Interstage\J2EE\etc
```

Solaris32/64

```
/etc/opt/FJSVj2ee/etc
```

Linux32/64

/etc/opt/FJSVj2ee/etc

2. The argument in javax.naming.InitialContext(Hashtable environment)
3. 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 2 is given the top priority).

1. Command line argument (-D) when the application is started
2. The argument in javax.naming.InitialContext(Hashtable environment)

The values that can be set in an environment property are given in the table below.

Windows32/64

The character strings of the environment property and 'java.naming.factory.initial' values are case-sensitive.

Solaris32/64 Linux32/64

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.InitialContextFactory ForClient | The InitialContext factory class name to access JNDI SP is specified. | All |
| FJUserID (*2) | Optional character string | A user name used in Directory Service 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 Directory Service 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 |
| HTTPGW | (1) For Interstage HTTP Server [Format for using SSL communication] https://host-name/url-name [Format for not using SSL communication] | The gateway that processes the HTTP tunneling is specified. If this value is omitted, HTTP tunneling is not used. (*4) | All |

| Environment property | Value | Setting contents | OS applied |
|---|--|--|------------|
| | <p>http://host-name/url-name</p> <p>host-name: Specifies the Web server that runs the HTTP-IIOP gateway.</p> <p>url-name: Specifies od-httpgw. For url-name, specify the URL of the Location directive.</p> <p>Windows32/64</p> <p>(2) For Internet Information Services</p> <p>[Format for using SSL communication]</p> <p>https://host-name/cgi-ID/gateway-name</p> <p>[Format for not using SSL communication]</p> <p>http://host-name/cgi-ID/gateway-name</p> <p>host-name: Specifies the Web server that runs the HTTP-IIOP gateway.</p> <p>cgi-ID: Specify "Virtual directory" alias name.</p> <p>gateway-name: Specifies ODhttp.dll (HTTP-IIOP gateway).</p> | | |
| VerificationMethod | <ul style="list-style-type: none"> - 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 |
| Windows32/64 Solaris32 Linux32/64 com.fujitsu.ObjectDirector. CORBA.GlobalTransaction Mode | <ul style="list-style-type: none"> - True Performed - False Not performed | Specify whether distributed transaction control is performed or not. | All |

*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 To specify a deployment descriptor file of J2EE 1.4 or later, use JDK/JRE 6.
- *4 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
http://[IPv6-address]/url-name
http://[IPv6-address]:Port_number/url-name
https://ipv4address_host-name/url-name
https://ipv4address_host-name:Port_number/url-name
```

2. For Internet Information Services

```
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.



Example

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.

Windows32/64

```
java -Djava.home=x:\aaaa\bbbb myapplication
```

Solaris32/64 **Linux32/64**

```
java -Djava.home=/aaaa/bbbb myapplication
```



Example

Description Example of FJjndi.properties File

Windows32/64

```
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
```

Solaris32/64

```
deployment descriptor file name:/export/home/j2eeapl/application-client.xml
Name Conversion file name      :ClientebeProperties.xml

com.fujitsu.interstage.j2ee.DeploymentDescriptorClient=/export/home/j2eeapl/application-client.xml
EBEproperties=ClientebeProperties.xml
```

Linux32/64

```
deployment descriptor file name:/home/j2eeapl/application-client.xml
Name Conversion file name      :ClientebeProperties.xml

com.fujitsu.interstage.j2ee.DeploymentDescriptorClient=/home/j2eeapl/application-client.xml
EBEproperties=ClientebeProperties.xml
```

3.1.2 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`.



Note

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 "[12.4 Using Java Applets](#)" in "How to call EJB Applications" for details of developing Java Applets.

When the Portable-ORB is used, refer to "[3.2.1 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 J Business Kit Online Manual of Interstage Studio 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 JDK 6
grant codeBase "file:/C:/Interstage/JDK6/jre/lib/-" {
permission java.security.AllPermission;
};
In the case of JRE 6
grant codeBase "file:/C:/Interstage/JRE6/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.FJCNctxFactoryForClient
```

3.2 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 the "Operating J2EE Applications" chapter, 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-J Edition

When Interstage client functions are installed, an environment setup is required. Refer to "[3.2.1 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.

Copy the client distribution items to the directory accessible to the EJB application that originated the call. If the EJB application is on another machine, copy the files to that machine.

Set the path for this directory in the EJB application originating call class path.

Windows32/64

```
C:\Interstage\J2EE\var\deployment\ijserver\[IJServer name]\distribute\[deployed ejb-jar file name]\
[client distribution data jar name] (*1)
```

Solaris32/64 **Linux32/64**

```
/opt/FJSVj2ee/var/deployment/ijserver/[IJServer name]/distribute/[deployed ejb-jar file name]/
[client distribution data jar name] (*1)
```

*1 [client distribution data jar 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".



Overwrite deployment may fail or files may remain as a result of a release of deployment if the client distribution items are not copied and a Java VM such as another IJServer directly referred these items.

3.2.1 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

When using the Client Package, define the Naming Service activated host name in the inithost file.

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 server package or Interstage Studio is being used, and runs when the statement shown below exceeds the initial value set in the config file, add the additional value shown below.

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.SingletonORB
-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 <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 the client package or Interstage Studio is being used and JBK has been installed, the text file is stored as follows:

```
<JBK-install-directory>\jre\lib (*1)
```

*1 For J2EE application client only



Example

[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.PortableRemoteObjectDelegatImpl
```

Setting Environment Variables

Windows32/64

| Environment variable | Setting value |
|----------------------|---|
| CLASSPATH | C:\Interstage\J2EE\lib\isj2ee.jar (*1) C:\Interstage\ODWin\etc\class\ODjava4.jar C:\Interstage\EJB\lib\fjcontainer94.jar (*2) |

Solaris32/64

| Environment variable | Setting value |
|----------------------|--|
| CLASSPATH | /opt/FJSVisj2ee/lib/isj2ee.jar (*1) /opt/FSUNod/etc/class/ODjava4.jar /opt/FJSVejb/lib/fjcontainer94.jar |
| LD_LIBRARY_PATH | /opt/FSUNod/lib |

Linux32/64

| Environment variable | Setting value |
|----------------------|--|
| CLASSPATH | /opt/FJSVisj2ee/lib/isj2ee.jar (*1) /opt/FJSVod/etc/class/ODjava4.jar /opt/FJSVejb/lib/fjcontainer94.jar |
| LD_LIBRARY_PATH | /opt/FJSVod/lib |

*1 Required for J2EE application client only

*2 In the case of the EJB service. In the case of an EJB client, specify the following:
C:\Interstage\EJBCL\lib\fjcontainer94.jar

3.3 Environment Setup when JDBC (Database) is Referenced

Configure the environment set up to use the JDBC data sources.

Before performing environment setup, install the JDBC driver following the instructions in the manual of the database being used.

The following environments are described in this section:

- Environment set up when Symfoware is used



Windows32/64 Solaris32/64

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

In addition to the use of the above databases, application operation during development can be confirmed using generically-defined data sources to verify connection to a database that is temporarily not supported.

For details, refer to "[3.3.5 Environment Setup when Generically-defined Data Sources are Used](#)".

Interstage Management Console Database Connection Test

The Interstage Management Console can be used to perform a connection test which verifies that the settings for connection to the database are correct.

To run the test, click [Resource] > [JDBC] > [Create New] or [Resource] > [JDBC] > [Environment Settings].

The following settings are required to run the database connection test:

- CLASSPATH, PATH, LIBRARYPATH

Refer to the environment settings for each database shown below, and set the required path. In the Interstage Management Console, click [System] > [Environment Settings] > [J2EE Properties]. Set the required class path in [J2EE Properties].

- environment variable **Solaris32/64** **Linux32/64**

If the database JDBC driver requires environment settings, refer to the environment settings for each database shown below, and set the required environment variable before starting the Interstage JMX service. The settings for the environment variable depend on the method that is used to start the Interstage JMX Service.

- Starting the Interstage JMX Service using the isjmxstart command

After the environment variable has been set, stop the Interstage JMX Service using the isjmxstop command, and then start it again using the isjmxstart command. This will activate the environment variable that was set.

- Starting the Interstage JMX Service using the 'S95isjmxstart' system initialization script

After the environment variable has been set, to ensure that it is activated when the OS starts up, add the environment variable to the 'S95isjmxstart' system initialization script. This will ensure that the environment variable is activated the next time the OS starts up. For details on subjects such as the S95isjmxstart storage directory, refer to "S95isjmxstart" in the Reference Manual (Command Edition).



Note

- The connection test checks that the JDBC data source definition contents are valid.

Not all of the IJServer environment settings are tested. Check the IJServer environment variables and access authority separately.

- When an Oracle database is used, the DB connection test does not check the validity of the connection or implicit connection cache properties. Refer to the Oracle manual to check that the connection and implicit connection cache properties are correct.
- To use the Oracle RAC function, obtain a connection by accessing the address list DB server. If at least one defined DB server can be connected correctly, no error will be generated so not all of the host names can be confirmed. To set the address list, check that the specified values are correct (in addition to performing the DB connection test function). To confirm that the host name has been set correctly, specify the host name and execute the ping command provided by the operating system and await a response from the correct IP address.
- Validity of "Data resource name" cannot be checked by executing the Symfoware DB connection test.

3.3.1 Environment Setup when Symfoware is Used

The environment setup method differs depending on the "data source type to be used", as follows:

- Using Interstage Connection Pooling
- Using Symfoware Connection Pooling

In Interstage Application Server 8.0 or earlier, only data sources that used Symfoware connection pooling could be used. In this version, data sources that use Interstage connection pooling can be used. With the use of Interstage connection pooling, activation of the JDBC naming service is no longer required but the connection pooling function provided by Interstage can be used. Use of Interstage connection pooling is therefore recommended.



If an incorrect data resource name is specified, the connection will be pooled, but an error will be generated when the application is executed.

Using Interstage Connection Pooling

The following environment set up is required when Symfoware is used.

- Setting Environment Variable
- Registering a Data Source with Interstage

When Symfoware on a non-Interstage server system is accessed, the operation below is also required.

- Symfoware Installed Server System's Environment Setup

Setting Environment Variable

The parameters of the environment variables to be set and the setup method for each operating environment are as follows:

- Setting Item
- Setting Method

Setting Item

Set the following item.

Windows32/64

| Setting item | Driver version | Setting value |
|--------------|-----------------------|--|
| PATH | - | <Symfoware Server client installation directory>\JDBC\fjjdbc\bin Windows(R) system directory\ESQL\BIN |
| CLASSPATH | SymfowareV10 or later | <Symfoware Server client installation directory>\JDBC\fjjdbc\lib\fjsymjdbc3.jar |

Solaris32/64

| Setting item | Driver version | Setting value |
|--------------|-----------------------|---|
| LIBRARY PATH | - | <FSUNrdb2b installation directory>/FSUNrdb2b/lib (*1) <JDBC driver installation directory>/FJSVsymjd/fjjdbc/bin (*1) |
| CLASSPATH | SymfowareV10 or later | <JDBC driver installation directory>/FJSVsymjd/fjjdbc/lib/ fjsymjdbc3.jar (*1) |

Linux32/64

| Setting item | Driver version | Setting value |
|--------------|-----------------------|---|
| LIBRARY PATH | - | <FJSVrdb2b installation directory>/FJSVrdb2b/lib (*1) <JDBC driver installation directory>/FJSVsymjd/fjjdbc/bin (*1) |
| CLASSPATH | SymfowareV10 or later | <JDBC driver installation directory>/FJSVsymjd/fjjdbc/lib/ fjsymjdbc3.jar (*1) |

*1 The default directory is '/opt'.

Setup Method when Using an IJServer

Implement the following settings when running a Web application or EJB application in an IJServer.

For isj2eeadmin command details, refer to "isj2eeadmin" in the Reference Manual (Command Edition).

| Setting item | Setting method |
|---|---|
| <p>Windows32/64</p> <p>PATH</p> | <p>Set one of the following paths:</p> <ul style="list-style-type: none"> - The path of the J2EE properties - The path of WorkUnit settings <p>Use one of the following methods to set the parameter:</p> <ul style="list-style-type: none"> - Interstage Management Console [System] > [Update System Settings] > [J2EE Settings] > [Path] [System] > [WorkUnit] > [IJServer Name] > [Settings] > [WorkUnit Settings] > [Path] - isj2eeadmin command J2EE system definition file IJServer definition file <p>Refer to "isj2eeadmin" in the Reference Manual (Command Edition) for details.</p> |
| <p>Solaris32/64</p> <p>Linux32/64</p> <p>LIBRARY PATH</p> | <p>Set the following library paths:</p> <ul style="list-style-type: none"> - The Library path of J2EE properties - The Library path of IJServer settings <p>Use one of the following methods to set the parameter:</p> <ul style="list-style-type: none"> - Interstage Management Console [System] > [Update System Settings] > [J2EE Settings] > [Library Path] [System] > [WorkUnit] > [IJServer Name] > [Settings] > [WorkUnit Settings] > [Library Path] - isj2eeadmin command J2EE system definition file IJServer definition file <p>Refer to "isj2eeadmin" in the Reference Manual (Command Edition) for details.</p> |
| <p>CLASSPATH</p> | <p>Set the following class paths:</p> <ul style="list-style-type: none"> - The Class Path of J2EE properties - The Class Path of WorkUnit settings <p>Use one of the following methods to set the parameter:</p> <ul style="list-style-type: none"> - Interstage Management Console [System] > [Update System Settings] > [J2EE Settings] > [Classpath] [System] > [WorkUnit] > [IJServer Name] > [Settings] > [WorkUnit Settings] > [Classpath] - isj2eeadmin command J2EE system definition file IJServer definition file <p>Refer to "isj2eeadmin" in the Reference Manual (Command Edition) for details.</p> <p>If the IJServer does not separate class loaders, this parameter will be enabled even if it is specified in a system environment variable. For details on setting the classpath used by an IJServer, refer to "Class Settings Used by IJServer" in "Class Loader" in the "Design of J2EE Application" chapter.</p> |

| Setting item | Setting method |
|--------------|--|
| | <p>The JDBC drivers set in the class path must not be stored in the "WEB-INF/lib" directory in the Web application directory structure. If the JDBC drivers are stored in this directory, it may not be possible to use the following functions:</p> <ul style="list-style-type: none"> - Pre-opened connection count - Reconnect on Failure |

Setup Method in a Client Environment

To operate a J2EE application client in a client environment, set the environment variable. An example is shown below.

[Example of setting environment variable in system environment variable in Windows(R)] [Windows32/64](#)

Select [Control Panel] > [System] > [Details] and click the environment variable button.



This explanation is for Windows Server(R) 2008. The operation method depends on the OS to be used.

[Example of setting environment variable in system environment variable by using command]

Windows32/64

When the Symfoware V10.0.0 and higher Symfoware Server client installation directory is 'C:\SFWCLNT', and the Windows(R) system directory is 'C:\WINNT'

```
set PATH=C:\WINNT\ESQL\BIN;%PATH%
set PATH=C:\SFWCLNT\JDBC\fjjdbc\bin;%PATH%
set CLASSPATH=C:\SFWCLNT\JDBC\fjjdbc\lib\fjsymjdbc3.jar;%CLASSPATH%
```

Solaris32/64

When the Symfoware V10.0.0 and higher JDBC driver installation directory and FSUNrdb2b installation directory is '/opt'

```
setenv LD_LIBRARY_PATH /opt/FSUNrdb2b/lib:${LD_LIBRARY_PATH}
setenv LD_LIBRARY_PATH /opt/FJSVsymjd/fjjdbc/bin:${LD_LIBRARY_PATH}
setenv CLASSPATH /opt/FJSVsymjd/fjjdbc/lib/fjsymjdbc3.jar:${CLASSPATH}
```

Linux32/64

When the Symfoware V10.0.0 and higher JDBC driver installation directory and FJSVrdb2b installation directory is '/opt'

```
setenv LD_LIBRARY_PATH /opt/FJSVrdb2b/lib:${LD_LIBRARY_PATH}
setenv LD_LIBRARY_PATH /opt/FJSVsymjd/fjjdbc/bin:${LD_LIBRARY_PATH}
setenv CLASSPATH /opt/FJSVsymjd/fjjdbc/lib/fjsymjdbc3.jar:${CLASSPATH}
```

Registering a Data Source with Interstage

The Interstage Management Console is used to define the data source. The data source can also be changed with the isj2eeadmin command. Refer to "isj2eeadmin" in the Reference Manual (Command Edition) 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 RDB2_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.

Windows32/64

```
Symfoware install drive:\SFWETC\RDB\ETC\UXPSQLENV
```

Solaris32/64

```
/opt/FSUNrdb2b/etc/fssqlenv
```

Linux32/64

```
/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 Application Server Messages manual for details of the output exception information.

RDB2_TCP Port Number Setting

Set the RDB2_TCP port number in the following file.

Windows32/64

In the case of Windows Server(R) 2008

```
Windows install directory\Windows\system32\drivers\etc\services
```

Solaris32/64 **Linux32/64**

```
/etc/services
```



Example

When 2,050 is allocated to port number

```
RDBII 2050/TCP
```

3.3.2 Using Symfoware Connection Pooling

In addition to the method used to enable Interstage connection pooling, perform the following tasks when the data source type being used requires the use of Symfoware connection pooling:

- Starting JDBC Naming Service
- Registering a Data Source with the JDBC Naming Service

Starting JDBC Naming Service

This task must be executed only when the data source type being used requires the use of Symfoware connection pooling.

If JDBC is used, the Symfoware Naming Service must be running before the JDBC data source can be used for registration/reference, and before J2EE applications can be executed.

Start the Naming Service using the following procedure.

1. Check the Java environment.

Check whether the Java environment is set up. For details on Java environment setup, refer to "Environment Setup of Java" in "Checking the Operating Environment" in the "Operating J2EE Applications" chapter.

2. Check environment variables.

Set the environment variables specified in "Setting environment variables".

3. Start the Naming Service

Start the Naming Service. The following example uses the Java command to start the Naming Service and change the port number to 10327. The default port number for the JDBC Naming Service is 10326.

Sample:

```
java com.fujitsu.symfoware.jdbc2.naming.SYMNameService 10327
```



Note

If the JDBC naming service has not started when using name conversion, 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 "Exceptions Output during J2EE Usage" in the Application Server Messages manual.

Registering a Data Source with the JDBC Naming Service

This operation must be executed only when the data source type being used requires the use of Symfoware connection pooling.

Register a data source with the JDBC Naming Service if JDBC is to be used.

Add the JDBC data sources using the following procedure.

1. Check the Java environment.

Check whether the Java environment is set up. For details on setting up the Java environment, refer to "Environment Setup of Java" in "Checking the Operating Environment" in the "Operating J2EE Applications" chapter.

2. Check the environment variables.

Set the environment variables specified in "Setting environment variables".

3. Start the JDBC data source registration tool.

Start the Symfoware JDBC data source registration tool as a user with administrator authority.

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 |

3.3.3 Environment Setup when Oracle is Used

The following environment set up is required when Oracle is used:

- Setting Environment Variable
- Registering a Data Source with Interstage

When using Oracle Real Application Clusters, refer to "Linkage with Oracle Real Application Clusters".

To use the File System Service Provider, in addition to the above operations, perform the following operations. The data source that does not use File System Service Provider is defined as the default.

- Environment set up when the File System Service Provider is used

Setting Environment Variable

The parameters to be set for the environment variables and the setup method for each operating environment are as follows:

- Setting Item
- Setting Method

Setting Item

Enter the following settings, which are required to operate the Oracle JDBC driver. The Oracle home directory is the directory selected to install Oracle software products. Specify the Oracle home directory in which Oracle has been installed.

Refer to the Oracle manual for further details.

Refer to the JDBC driver manual for details of the Java versions supported by each JDBC driver.

| Setting item | Oracle version | Setting value (*1) |
|--------------|--------------------|---|
| CLASSPATH | Oracle11g or later | <Oracle home directory>\jdbc\lib\ojdbc6.jar(*2) |
| | | <Oracle home directory>\jdbc\lib\orai18n.jar |

*1 For Solaris or Linux, replace "\" with "/".

*2 Use this within the scope of JDBC3.0 - functionalities added for JDBC4.0 cannot be used.

When the OCI driver is used, the following setting is required in addition to the above setting.

Windows32/64

| Setting item | Oracle version | Setting value (*1) |
|--------------|----------------|---------------------------|
| PATH | Common | Oracle home directory\bin |

Solaris32/64 Linux32/64

| Setting item | Oracle version | Setting value |
|--------------|-------------------------|--------------------------------|
| LIBRARY_PATH | Oracle11g or later (*1) | Oracle install directory/lib32 |
| | | Oracle home directory/lib |
| ORACLE_HOME | Common | Oracle home directory |

*1 Set the library path in the order described in the above table.

Setting Method

Setup Method when Using an IJServer

Implement the following settings when running a Web application or EJB application in an IJServer.

Refer to "isj2eeadmin" in the Reference Manual (Command Edition) for details on using the isj2eeadmin command.

| Setting item | Setting method |
|---|---|
| <p>Windows32/64</p> <p>PATH</p> | <p>Set the following paths:</p> <ul style="list-style-type: none"> - The path of J2EE properties - The path of WorkUnit settings <p>Use one of the following methods to set the parameter:</p> <ul style="list-style-type: none"> - Interstage Management Console <ul style="list-style-type: none"> [System] > [Update System Settings] > [J2EE Settings] > [Path] [System] > [WorkUnit] > [IJSERVER Name] > [Settings] > [WorkUnit Settings] > [Path] - isj2eeadmin command <ul style="list-style-type: none"> J2EE system definition file IJSERVER definition file <p>Refer to "isj2eeadmin" in the Reference Manual (Command Edition) for details.</p> |
| <p>Solaris32/64</p> <p>Linux32/64</p> <p>LIBRARY PATH</p> | <p>Set the following library paths:</p> <ul style="list-style-type: none"> - The library path of J2EE properties - The library path of WorkUnit settings <p>Use one of the following methods to set the parameter:</p> <ul style="list-style-type: none"> - Interstage Management Console <ul style="list-style-type: none"> [System] > [Update System Settings] > [J2EE Settings] > [Library Path] [System] > [WorkUnit] > [IJSERVER Name] > [Settings] > [WorkUnit Settings] > [Library Path] - isj2eeadmin command <ul style="list-style-type: none"> J2EE system definition file IJSERVER definition file <p>Refer to "isj2eeadmin" in the Reference Manual (Command Edition) for details.</p> |
| <p>CLASSPATH</p> | <p>Set the following class paths:</p> <ul style="list-style-type: none"> - The Class Path of J2EE properties - The Class Path of WorkUnit settings <p>Use one of the following methods to set the parameter:</p> <ul style="list-style-type: none"> - Interstage Management Console <ul style="list-style-type: none"> [System] > [Update System Settings] > [J2EE Settings] > [Classpath] [System] > [WorkUnit] > [IJSERVER Name] > [Settings] > [WorkUnit Settings] > [Classpath] - isj2eeadmin command <ul style="list-style-type: none"> J2EE system definition file IJSERVER definition file <p>Refer to "isj2eeadmin" in the Reference Manual (Command Edition) for details.</p> <p>If the IJSERVER does not separate class loaders, this parameter will be enabled even if it is specified in a system environment variable. For details on how to set the classpath used by an IJSERVER, refer to "Class Settings Used by IJSERVER" in "Class Loader" in the "Design of J2EE Application" chapter.</p> <p>The JDBC drivers set in the class path must not be stored in the "WEB-INF/lib" directory in the Web application directory structure. If the JDBC drivers are stored in this directory, it may not be possible to use the following functions:</p> |

| Setting item | Setting method |
|---|---|
| | <ul style="list-style-type: none"> - Pre-opened connection count - Reconnect on Failure |
| Solaris32/64 Linux32/64 ORACLE_HOME | Set ORACLE_HOME as follows: <ul style="list-style-type: none"> - The Environment settings of IJServer settings ORACLE_HOME=/opt/oracle Use one of the following to set the parameter: <ul style="list-style-type: none"> - Interstage Management Console - isj2eedadmin command |

Setup Method in a Client Environment

To operate a J2EE application client in a client environment, set the environment variable. An example is shown below.

[Example of setting environment variable in system environment variable in Windows(R)] **Windows32/64**

Select [Control Panel] > [System] > [Details] and click the environment variable button.

Note

This explanation is for Windows Server(R) 2008. The operation method depends on the OS to be used.

[Example of setting environment variable in system environment variable by using command]

ORACLE_HOME is the installation home directory for Oracle.

Windows32/64

```
set CLASSPATH=%CLASSPATH%;%ORACLE_HOME%\jdbc\lib\ojdbc6.jar
set CLASSPATH=%CLASSPATH%;%ORACLE_HOME%\jdbc\lib\orai18n.jar
```

Solaris32/64 **Linux32/64**

```
[When using the C shell]
setenv CLASSPATH ${CLASSPATH}:${ORACLE_HOME}/jdbc/lib/ojdbc6.jar
setenv CLASSPATH ${CLASSPATH}:${ORACLE_HOME}/jdbc/lib/orai18n.jar
```

Registering a Data Source with Interstage

The Interstage Management Console is used to define the data source. The data source can also be changed with the isj2eedadmin command. Refer to "isj2eedadmin" in the Reference Manual (Command Edition) for details.

To define the data sources, select from the following data source types according to the function to be used.

| Data source type | Use |
|---|---|
| To use Interstage connection pooling | The data source enables the use of the Interstage connection pooling function. This type is selected by default. |
| To use Oracle connection pooling | The data source enables the use of the Oracle implicit connection cache. Select this type in the following instances: <ul style="list-style-type: none"> - The high-speed connection failover function of the Oracle Real Application Clusters (Oracle RAC) is used. - The Statement cache function is used. - Tuning using the Oracle connection cache property is to be performed. |
| Windows32/64 Solaris32 | Select this type when the distributed transaction (global transaction) environment is used |

| Data source type | Use |
|---|-----|
| Linux32/64 To use distributed transactions | |

The check box used to define "To use the global transactions" in Interstage Application Server 8.0 or earlier versions has been removed. To use global transactions, select "To use distributed transactions" as the "data source type". When the data source defined in the "To use global transactions" check box of Interstage Application Server 8.0 or earlier versions is restored in the current version, the display indicates that "To use distributed transactions" has been selected.

Environment Setup when the File System Service Provider is Used

In addition to the above, the following must be set to use the File System Service Provider:

Setting Environment Variable

In addition to JDBC driver settings, the following must be set. The File System Service Provider is offered in Interstage and is already set in the classpath shown in the table below. For this reason, this task is not required if you are using IJServer.

| Setting item | Oracle version | JDK/JRE version | Setting value |
|--------------|----------------|-----------------|---|
| CLASSPATH | common | common | Windows32/64 C:\Interstage\J2EE\lib\providerutil.jar C:\Interstage\J2EE\lib\fscontext.jar Solaris32/64 Linux32/64 /opt/FJSVj2ee/lib/providerutil.jar /opt/FJSVj2ee/lib/fscontext.jar |

To use the latest jar file, download the File System Service Provider from the Oracle website and set the CLASSPATH environment variable.

If you are using a client environment, set the jar provided in Interstage (shown in the table above) or the classpath of the downloaded jar.

Registering Data Sources with the File System Service Provider

Register data sources with the File System Service Provider. If the Interstage Management Console or isj2eeadmin command is used to register a data source with Interstage, it is possible to specify that the data source be registered with the File System Service Provider when it is registered with Interstage.

In such cases, it is not necessary to perform this step. Refer to "isj2eeadmin" in the Reference Manual (Command Edition) for details on using the isj2eeadmin command.

To register a data source with the File System Service Provider, create a data source registration application by referring to the sample source code in the following location and then use it to register the data source.

Refer to the Oracle manual for details on data source registration applications.

Windows32/64

```
C:\Interstage\J2EE\sample\datasource\FJDSJNDILOCAL.java
```

When a data source is registered with the File system Service Provider, a file named ".bindings" will be created in the location specified by PROVIDER_URL.

3.3.4 Environment Setup when SQL Server is Used

Also, in this section, the environment setup for connecting with SQL Server using a Microsoft (R) JDBC driver is explained.

The following environment set up is required when SQL Server is used:

- Remote Connection Settings

- Setting Environment Variable
- Registering a Data Source with Interstage
- JDBC Driver Logging Function

To use the File System Service Provider, the following operations must be performed in addition to the above operations. A data source that does not use File System Service Provider is defined as the default.

- For details, refer to "[3.3.5 Environment Setup when Generically-defined Data Sources are Used](#)".

Downloading and Installing Microsoft(R) JDBC Driver

1) Download

The Microsoft(R) JDBC driver is not included in Microsoft(R) SQL Server(TM).

Download it from the Microsoft website.

Use a JDBC driver that is compatible with the version of SQL Server(TM) being used.

2) Installation

Refer to the installation method described on the Microsoft website for details of the installation.

Remote Connection Settings

If you use the free version of a product, or a product for development or test systems, you may not be able to connect remotely using the default settings.

Refer to the Microsoft(R) SQL Server(TM) manual and make the settings so that remote connection can be used.

Setting Environment Variable

The parameters of the environment variables to be set and the setup method for each operating environment are as follows:

- Setting Item
- Setting Method

Setting Item

Enter the following settings, which are required to operate the Microsoft(R) JDBC driver.

| Setting item | Setting value |
|--------------|---|
| CLASSPATH | JDBC Driver install directory\<version>\<location>\sqljdbc.jar (*1) |

*1 <version>: In the case of SQL Server JDBC Driver 3.0, "sqljdbc_3.0"

<location>: In the case of Japanese version, "jpn". In the case of English version, "enu".

Setting Method

Setup Method when Using an IJServer

Implement the following settings when running a Web application or EJB application in an IJServer.

Refer to "isj2eeadmin" in the Reference Manual (Command Edition) for details on using the isj2eeadmin command.

| Setting item | Setting method |
|--------------|--|
| CLASSPATH | Set the following class paths: <ul style="list-style-type: none"> - The Class Path of J2EE properties - The Class Path of WorkUnit settings Use one of the following methods to set the parameter: <ul style="list-style-type: none"> - Interstage Management Console [System] > [Update System Settings] > [J2EE Settings] > [Classpath] |

| Setting item | Setting method |
|--------------|---|
| | <p>[System] > [WorkUnit] > [IJServer Name] > [Settings] > [WorkUnit Settings] > [Classpath]</p> <ul style="list-style-type: none"> - isj2eeadmin command <p>J2EE system definition file</p> <p>IJServer definition file</p> <p>Refer to "isj2eeadmin" in the "Reference Manual (Command Edition)" for details.</p> <p>If the IJServer does not separate class loaders, this parameter will be enabled even if it is specified in a system environment variable. For details on setting the classpath used by an IJServer, refer to "Class Used by IJServer" in "IJServer File Configuration" in "Designing J2EE Applications".</p> <p>The JDBC drivers set in the class path must not be stored in the "WEB-INF/lib" directory in the Web application directory structure. If the JDBC drivers are stored in this directory, it may not be possible to use the following functions:</p> <ul style="list-style-type: none"> - Pre-opened connection count - Reconnect on Failure |

Setup Method in a Client Environment

To operate a J2EE application client in a client environment, set the environment variable. An example is shown below.

[Example of setting environment variable in system environment variable in Windows(R)]

Windows32/64

Select [Control Panel] > [System] > [Details] and click the environment variable button.

Note

This explanation is for Windows Server(R) 2008. The operation method depends on the OS to be used.

[Example of setting environment variable in system environment variable by using command]

Windows32/64

```
set CLASSPATH=%CLASSPATH%;C:\mssqlserver\sqljdbc_3.0\jpn\sqljdbc.jar
```

Solaris32/64 **Linux32/64**

```
[When using the C shell]
setenv CLASSPATH ${CLASSPATH}:/opt/mssqlserver/sqljdbc_3.0/jpn/sqljdbc.jar
```

Registering a Data Source with Interstage

The Interstage Management Console is used to define the data source. The data source can also be changed with the isj2eeadmin command. Refer to "isj2eeadmin" in the Reference Manual (Command Edition) for details.

JDBC Driver Logging Function

In Microsoft(R) JDBC Driver, the java.util.logging package logging function offered in JDK can be used for debugging. For details about how to output a JDBC Driver log, refer to the JDBC Driver documentation. The default standard output and standard error output are output in the container log.

Note

The java.util.logging package logging function offered in JDK is also used to output Interstage user snap information. For this reason, the JDBC Driver log is also output by default when the user snap information is output. To prevent the output of the JDBC Driver log, make the definitions shown in the table below.

| | |
|-----------------------------------|--|
| Definition file storage directory | <p>Windows32/64</p> <p>C:\Interstage\EJB\etc</p> <p>Solaris32/64 Linux32/64</p> <p>/opt/FJSVejb/etc</p> |
| Definition filename | FJlogging.properties |
| Additional definition | com.microsoft.sqlserver.jdbc.level = OFF |

Environment Setup when the File System Service Provider is Used

In addition to the above procedures, the following procedures must be implemented to use the File System Service Provider:

Setting Environment Variable

In addition to the JDBC driver settings described above, the following must be set. The File System Service Provider is provided in Interstage and is already set in the classpath shown in the table below. For this reason, this task is not required if you are using IJServer.

| Setting item | Setting value |
|--------------|---|
| CLASSPATH | <p>Windows32/64</p> <p>C:\Interstage\J2EE\lib\providerutil.jar</p> <p>C:\Interstage\J2EE\lib\fscontext.jar</p> <p>Solaris32/64 Linux32/64</p> <p>/opt/FJSVj2ee/lib/providerutil.jar</p> <p>/opt/FJSVj2ee/lib/fscontext.jar</p> |

To use the latest jar file, download File System Service Provider from the Oracle website and set the CLASSPATH environment variable. If you are using a client environment, set the jar provided in Interstage (shown in the table above) or the classpath of the downloaded jar.

Registering Data Sources with the File System Service Provider

Register data sources with the File System Service Provider. If the Interstage Management Console or isj2eadmin command is used to register a data source with Interstage, it is possible to specify that the data source be registered with the File System Service Provider when it is registered with Interstage.

In such cases, it is not necessary to perform this step. Refer to "isj2eadmin" in the Reference Manual (Command Edition) for details on using the isj2eadmin command.

To register a data source with the File System Service Provider, create a data source registration application by referring to the sample source code in the following location and then use it to register the data source.

The sample source code is intended for use with Oracle, so it must be modified for use with the Microsoft(R) JDBC driver. Refer to the Microsoft(R) JDBC driver manual for details on data source registration applications.

Windows32/64

```
C:\Interstage\J2EE\sample\datasource\FJDSJNDILOCAL.java
```

When a data source is registered with the File System Service Provider, a file named ".bindings" will be created in the location specified by PROVIDER_URL.

3.3.5 Environment Setup when Generically-defined Data Sources are Used

Data sources for a database or a JDBC driver can be created when the JDBC data source is created. This process is referred to as generically defining a data source. Application operation during development can be confirmed by using generically-defined data sources to verify connection to a database that is temporarily not supported.. Use of supported databases or JDBC drivers is recommended for the production

run (operation is not guaranteed when unsupported databases or JDBC drivers are used). Note the following when generically-defined data sources are used.

Connection Pooling Function

When the `java.sql.ConnectionPoolDataSource` interface is implemented by the generically-defined data source cluster, connections to data sources will be pooled on the Interstage side. Note that the automatic re-connection function cannot be used.

When a data source that does not implement the `java.sql.ConnectionPoolDataSource` interface is defined, connections will not be pooled on the Interstage side. Whether connections are pooled or not depends on the implementation on the JDBC driver side. For details, refer to the manual for the JDBC drivers (or database in use).

EJB CMP1.1 Entity Bean and CMP2.0 Entity Bean

The CMP 1.1 Entity Bean or CMP 2.0 Entity Bean to which the EJB container performs database access processing cannot use the generically-defined data sources. When a data source that cannot be used with CMP definition is specified, an error will be generated at the time of IJServer activation. When a module setting a data source that cannot be used with CMP definition is deployed using the J2EE HotDeploy function, the activation processing will fail and result in deactivation.

The CMP1.1 Entity Bean or CMP2.0 Entity Bean can use the data sources temporarily during development by specifying an option. Neither the CMP1.1 Entity Bean nor the CMP2.0 Entity Bean can use generically-defined data sources. To use them temporarily, specify the following option.

| | |
|---------------------------|--|
| Definition file directory | <div style="border: 1px solid black; padding: 2px; display: inline-block;">Windows32/64</div> C:\Interstage\EJB\etc <div style="border: 1px solid black; padding: 2px; display: inline-block; margin-right: 5px;">Solaris32/64</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Linux32/64</div> /opt/FJSVejb/etc |
| Definition file name | FJEJBconfig.properties |
| Key to specify | "cmp_generic_datasource_use" |
| Value to specify | Specify "yes". |

If a container error is generated after specifying this option, cancel the option specification immediately. The following restriction applies when non-supported databases are used:

- [Restriction]

When non-supported databases have been used and the record data being entered (using the create method of the CMP 1.1 Entity Bean) already exists, a "java.rmi.RemoteException" will be returned. If the databases are supported, a("javax.ejb.DuplicateKeyException" will be returned).

Essential API

To use generically-defined data sources using the JNDI lookup function of J2EE, the database or JDBC driver in use must support the following APIs.

| Interface name | Method name | Remarks |
|------------------------------------|--|---|
| javax.sql.ConnectionPoolDataSource | getPooledConnection() | Only when the data source implementing the <code>java.sql.ConnectionPoolDataSource</code> interface is defined. |
| | getPooledConnection(String user,String password) | |
| javax.sql.DataSource | getConnection() | |
| | getConnection(String user, String password) | |
| javax.sql.PooledConnection | addConnectionEventListener(ConnectionEventListener listener) | Only when the data source implementing the <code>java.sql.ConnectionPoolDataSource</code> interface is defined. |
| | close() | |
| | getConnection() | |

| Interface name | Method name | Remarks |
|---------------------|---|---|
| | removeConnectionEventListener(ConnectionEventListener listener) | |
| java.sql.Connection | close() | <p>To obtain connection after the J2EE transaction has started, settings such as AutoCommit should be implemented in the following sequence:</p> <ol style="list-style-type: none"> 1. setAutoCommit(false) 2. setTransactionIsolation(Set value) 3. setReadOnly(false) <p>The Setting shown in item 2 above is executed only when a non-default value is set for the isolation-level in the DB connection settings.</p> |
| | commit() | |
| | getAutoCommit() | |
| | getTransactionIsolation() | |
| | isClosed() | |
| | isReadOnly() | |
| | setAutoCommit(boolean autoCommit) | |
| | setReadOnly(boolean readOnly) | |
| | setTransactionIsolation(int level) | |

Setting up Environment

The following environment set up is required:

- Setting Environment Variable
- Registering a Data Source with Interstage
- Setting Information for other Data Source Properties

Setting Environment Variable




Refer to the JDBC driver manual or the manual for each database in use for details of the required environment settings.

Setting Method

Setup Method when Using an IJServer

Implement the following settings when running a Web application or EJB application in an IJServer.

Refer to "isj2eadmin" in the Reference Manual (Command Edition) for details on using the isj2eadmin command.

| Setting item | Setting method |
|--|--|
|  PATH | <p>Set the following paths:</p> <ul style="list-style-type: none"> - The Path of J2EE properties - The Path of WorkUnit settings <p>Use one of the following methods to set the parameter:</p> <ul style="list-style-type: none"> - Interstage Management Console <ul style="list-style-type: none"> [System] > [Update System Settings] > [J2EE Settings] > [Path] [System] > [WorkUnit] > [IJServer Name] > [Settings] > [WorkUnit Settings] > [Path] - isj2eadmin command <ul style="list-style-type: none"> J2EE system definition file IJServer definition file <p>Refer to "isj2eadmin" in the Reference Manual (Command Edition) for details.</p> |
|   LIBRARY PATH | <p>Set the following library paths:</p> <ul style="list-style-type: none"> - The Library Path of J2EE properties - The Library Path of WorkUnit settings |

| Setting item | Setting method |
|--------------|---|
| | <p>Use one of the following methods to set the parameter:</p> <ul style="list-style-type: none"> - Interstage Management Console [System] > [Update System Settings] > [J2EE Settings] > [Library Path] [System] > [WorkUnit] > [IIServer Name] > [Settings] > [WorkUnit Settings] > [Library Path] - isj2eeadmin command J2EE system definition file IIServer definition file <p>Refer to "isj2eeadmin" in the Reference Manual (Command Edition) for details.</p> |
| CLASSPATH | <p>Set the following class paths:</p> <ul style="list-style-type: none"> - The Class Path of J2EE properties - The Class Path of WorkUnit settings <p>Use one of the following methods to set the parameter:</p> <ul style="list-style-type: none"> - Interstage Management Console [System] > [Update System Settings] > [J2EE Settings] > [Classpath] [System] > [WorkUnit] > [IIServer Name] > [Settings] > [WorkUnit Settings] > [Classpath] - isj2eeadmin command J2EE system definition file IIServer definition file <p>Refer to "isj2eeadmin" in the Reference Manual (Command Edition) for details.</p> <p>If the IIServer does not separate class loaders, this parameter will be enabled even if it is specified in a system environment variable. For details on how to set the classpath used by an IIServer, refer to "Class Settings Used by IIServer" in "Class Loader" in the "Design of J2EE Application" chapter.</p> <p>The JDBC drivers set in the class path must not be stored in the "WEB-INF/lib" directory in the Web application directory structure. If the JDBC drivers are stored in this directory, it may not be possible to use the following functions:</p> <ul style="list-style-type: none"> - Pre-opened connection count - Reconnect on Failure |

Setup Method in a Client Environment

To operate a J2EE application client in a client environment, set the environment variable.

Registering a Data Source with Interstage

The Interstage Management Console is used to define the data source. The data source can also be changed using the isj2eeadmin command. Refer to "isj2eeadmin" in the Reference Manual (Command Edition) for details.

Setting Information for other Data Source Properties

Information set for other data source properties is set for data sources as follows at the time of the DB connection test or when the data source is used.

1. Checks whether or not a method with the name, "set + data-source-property-name", exists in the data sources. If the initial letter of the data source property name is a lower case letter, it is converted to a capital letter and joined to "set".

Example: If the data source property name is "property", executes a method called "setProperty"

2. If a method name "1." exists in the data sources and the method has one argument, converts the data source property value to the argument type data and executes the method for that data source

An error will be generated in the following situations. Check if the data source property has been set correctly using the DB connection test function.

- When the property name is incorrect.
- When the property data type is incorrect.
- When a value that cannot be converted to the property data type has been specified as the property value.

3.4 JDBC (Database) Connections

This section explains the following functions, which are used during JDBC (database) connections:

- Connection Pooling
- Automatic Reconnection Function

Refer to "Tuning of IJServer" in the Tuning Guide for a detailed description of connection pooling and the automatic reconnection function.

3.4.1 Connection Pooling

Connection pooling is a function for pooling (maintaining) and reusing connections to a database instead of creating a new connection each time one is required.

Connection pooling can reduce the number of times an application from the same user connects to a database, and reduce the load on applications that establish database connections.

The following aspects of connection pooling are explained in this section:

- Connection Pooling Methods that can be Used in Interstage
- The Units of Connection Pool Creation
- Connection Pools and Data Sources

Connection Pooling Methods that can be Used in Interstage

The following two connection pooling methods can be selected in Interstage Application Server:

- Connections are Pooled by Interstage
- Connections are Pooled by JDBC

Connections are Pooled by Interstage

Connection pools are created on each Java process on an IJServer and are managed by the IJServer.

Interstage manages the connection pools, and provides functions such as the fine-tuning of connection pools and an automatic reconnect function for JDBC connections. It also makes it possible to monitor how the connections are used.

Connections are Pooled by JDBC

Connection pools are created on a JDBC driver and are managed by the JDBC driver.

Because connection pool management is performed by a JDBC driver, connection pool tuning relies on the settings of the JDBC driver. This means that it is not possible to use Interstage functions, such as the automatic reconnect function for JDBC connections and connection usage monitoring.

The Units of Connection Pool Creation

Interstage creates connection pools according to the number of concurrent processes on each IJServer.

This is the same for both of the connection pooling methods used in Interstage. Connection pools are created according to the number of concurrent processes on each IJServer.

Connection Pools and Data Sources

Connection pools are managed and pooled by each data source.

Connection objects are pooled in a connection pool while they are not being used by applications. Pooled connections are discarded at the following times:

- When an idle timeout occurs
- When the connection is determined to have been disabled by the automatic reconnection function
- When the IJServer stops running
- When the ijstune command is executed (or if there is a transaction while the command is being executed, after the transaction closes)

This also means that Connection objects are cached and reused within the same transaction. Refer to "Transaction Control" in the "Design of J2EE Application" chapter, for details.

Refer to "Tuning of IJServer" in the Tuning Guide for information on the correspondence between databases and connection pooling.

3.4.2 Automatic Reconnection Function

The automatic reconnection function discards pooled connections that have become invalid because of a database server failure or communication error. It then reconnects to the database server after it has been restored to enable database access to continue.

This function makes it possible to automatically reconnect to a database without having to restart the IJServer.

Example of the Benefits of the Automatic Reconnection Function

The operation of this function is explained below using the following sample system:

- The database used in the example is Oracle Real Application Clusters (referred to as the Oracle RAC in this document).
- The Oracle RAC divides operation roles between an "active server" and a "standby server".
- During normal operation, a connection is only established with the active server.

An application server (referred to in this example as the IJServer) pools connections with a connected database and returns pooled connections when getConnection is subsequently called.

If a problem occurs in the active server that is normally connected, operations will automatically switch to the standby server. Connections that have been pooled in the IJServer become invalid when the active server fails. This means that once operations have switched to the standby server, any attempt by an application to access a database will generate an error and the application will fail to run.

To avoid this problem, the system needs to be changed so that connections in the IJServer connection pool that automatically become invalid when the active server fails are released.

Using the Automatic Reconnection Function

An IJServer performs the following processes to recover (reconnect) a database server after it has failed.

Using SQL (*1) to verify a connection

When an IJServer receives a getConnection call from an application, it issues an SQL (*1) statement to a connection obtained from the pool to verify the connection.

If the operation completes normally, the acquired object is judged to be valid and is returned.

If an exception occurs, the connection obtained from the pool is discarded and the getConnection call is executed in the database directly.

Recovery mechanism when a database connection error occurs

If the SQL (*1) statement generates an exception (or there is no pool when the getConnection is called from the application, so getConnection is called in the database directly and an exception returned), the database connection process is repeated a specified number of times over a specified interval until it completes normally.

Once the connection succeeds, the connection to the application is returned.

If the connection fails after the specified number of retries, SQLException is returned.

*1

The following SQL statement, which is recommended by Oracle, is issued:

```
select sysdate from dual
```

In this example, the SQL statement "select sysdate from dual" is issued because an Oracle database is used. If Symfoware or SQL Server is used, the following statement is issued. In Symfoware, the following SQL statement causes a syntax error but the error code issued at the time of error generation confirms whether or not the database connection was made normally.

```
select 1
```

Refer to "Tuning of IJServer" in the Tuning Guide for a detailed description of the automatic reconnection function.

Reconnection Timing

Reconnection to a DBMS following a DBMS error occurs when a connection is obtained from the connection pool, or when a connection is obtained directly from the DBMS if the pool contains no connections. (Transactions commence and SQL can be issued when connection to the DBMS is confirmed.)

If the DBMS fails during a transaction or after a connection is obtained, an exception will occur when a database or an Entity Bean (CMP) is accessed (SQL is issued) within a transaction. (In such cases, reconnection to the DBMS will not occur.)

If this problem occurs, close the JDBC connection, roll back the transaction, and try to continue the process. (If the transaction management type is Container, the container will perform rollback processing.)

The following table lists the exceptions that occur when an invalid connection is used by an Entity Bean (CMP).

| Method | Returned Exception |
|--|--|
| Create | java.rmi.RemoteException or javax.transaction.TransactionRolledbackException |
| findByPrimaryKey | |
| find<METHOD> | |
| Remove | |
| nextElement method corresponding to the "Enumeration" return value of the find<METHOD> | java.lang.RuntimeException |
| next method corresponding to the Iterator of the "Collection" return value of the find<METHOD> | |

The following table lists the exceptions that are returned when a transaction completes.

| Method | Returned Exception |
|----------|--|
| Commit | javax.transaction.HeuristicMixedException or javax.transaction.HeuristicRollbackException (When the transaction is marked as having been rolled back) |
| Rollback | - |

Refer to "Tuning of IJServer" in the Tuning Guide for a detailed description of the automatic reconnection function.

Refer to "[Appendix B Linkage with Oracle Real Application Clusters](#)" when linking with Oracle RAC.

3.4.3 Supported APIs

Only the JDBC standard APIs are supported. Use the standard API as a method specific to each JDBC driver (i.e. OracleConnection-specific method) cannot be used. The java.lang.ClassCastException error will be generated if an object is cast to the JDBC driver-specific class in order to execute a specific method.

3.5 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 the "Operating J2EE Applications" chapter 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 20 Developing a JMS Application](#)" for details.

Resource Access Definition

Make the resource access definition on the Interstage Management Console or using the isj2eadmin command. For isj2eadmin command details, refer to the Reference Manual (Command Edition).

When Interstage client functions are installed, use the JMS operation command. Refer to "J2EE Resource Access Definition" for details.

Refer to the "JMS Operation Commands" chapter in the Reference Manual (Command Edition) for details.

3.6 Environment Setup when JavaMail is Referenced

When JavaMail is referenced, make the resource access definition on the Interstage Management Console or using the isj2eadmin command. For isj2eadmin command details, refer to the Reference Manual (Command Edition).

3.7 Environment Setup when URL is Referenced

When a URL is referenced, be sure to use the name conversion function.

Refer to "[3.11 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>

```

3.8 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 or the `isj2eeadmin` command 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 "[1.1.4 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.

Windows32/64

```

J2EE common directory\deployed\jca\ra\[Resource name]
* The J2EE common directory default value is C:\Interstage\J2EE\var\deployment.

```

Solaris32/64 **Linux32/64**

```

J2EE common directory/deployment/deployed/jca/ra/[Resource name]
* The J2EE common directory default value is /opt/FJsvj2ee/var/deployment.

```

When Distributed Transaction is Used **Windows32/64** **Solaris32** **Linux32/64**

Select [Resource] > [connector] > [Deploy] on the Interstage Management Console and set "Use Global Transaction" to "Use".

The `isj2eeadmin` command can also be used to implement these settings. For details, refer to the Reference Manual (Command Edition).

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. Like the following examples, set up is performed via the Interstage Management Console or using the isj2eeadmin command.

- Select [WorkUnit] > "WorkUnit Name" > [WorkUnit Setting] and set the following in [Path] on the Interstage Management Console.

Windows32/64

```
C:\Interstage\J2EE\var\deployment\deployed\jca\ra\RA01
```

Solaris32/64 Linux32/64

```
/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.

Windows32/64

```
C:\Interstage\J2EE\var\deployment\deployed\jca\ra\RA01\RA01.jar
```

Solaris32/64 Linux32/64

```
/opt/FJSVj2ee/var/deployment/deployed/jca/ra/RA01/RA01.jar
```

When Distributed Transaction is Used Windows32/64 Solaris32 Linux32/64

Windows32/64

Set PATH and CLASSPATH in the system environment variable. Note that after the setting, the OS must be rebooted.

Solaris32 Linux32/64

Set PATH and CLASSPATH in the system environment variable. Note that the setting must be made before Interstage is started up.

3.9 Description in Deployment Descriptor File

Write reference object information to the deployment descriptor file.

The following explains the tags for object reference.

For details on the deployment descriptor file, refer to:

- For J2EE application clients, refer to "[3.13 J2EE Application Client Deployment Descriptor File Detailed Setup](#)".
- For Web applications, refer to "[6.3 Web Application Environment Definition File \(Deployment Descriptor\)](#)".
- For EJB applications, refer to the "Interstage Studio User's Guide" (note that in this case you should use the EJB deployment descriptor editor of Interstage Studio).

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 |
| service-ref | Web service |
| resource-ref | JDBC datasource |
| | JMS connection factory |
| | JavaMail mail session |
| | URL(Uniform Resource Locator) |
| | connector connection factory |
| resource-env-ref | JMS Destination |
| message-destination-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. | | | | | | | | | | | | |
| <table border="1"> <tr> <td>description</td> <td>Specifies optional information to be indicated to the user. It can be omitted.</td> </tr> <tr> <td>ejb-ref-name</td> <td>Specifies Enterprise Bean's reference name with the following prefix added. (xxxxx: optional character string) - ejb/xxxxx</td> </tr> <tr> <td>ejb-ref-type</td> <td>Specifies Enterprise Bean's application type in one of the following. - Entity - Session</td> </tr> <tr> <td>Home</td> <td>Specifies a home interface name. As the home interface name, specify a restricted name (a package internal interface name).</td> </tr> <tr> <td>Remote</td> <td>Specifies a remote interface name. As the remote interface name, specify a restricted name (a package internal interface name).</td> </tr> <tr> <td>ejb-link</td> <td>Specifies an Enterprise Bean name. It can be omitted.</td> </tr> </table> | 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 an Enterprise Bean name. It can be omitted. | |
| | 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 an 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. | | | | | | | | | | | | |
| <table border="1"> <tr> <td>description</td> <td>Specifies optional information to be indicated to the user. It can be omitted.</td> </tr> <tr> <td>ejb-ref-name</td> <td>Specifies Enterprise Bean's reference name with the following prefix added. (xxxxx: optional character string) - ejb/xxxxx</td> </tr> <tr> <td>ejb-ref-type</td> <td>Specifies Enterprise Bean's application type in one of the following. - Entity - Session</td> </tr> </table> | 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 | | | | | | | |
| | 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 | | | | | | | | | | | | |

| Tag | | Explanation |
|-------------------------|-----------------------|---|
| | 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 an Enterprise Bean name. It can be omitted. |
| service-ref | | Defines information related to a Web service reference. It can be specified more than once. For details, refer to "Service Reference Description" in the "Developing Web Services" chapter. Note: This can be set in deployment descriptor files of J2EE1.4 and higher. |
| 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) <ul style="list-style-type: none"> - 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. <ul style="list-style-type: none"> - 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. <ul style="list-style-type: none"> - 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) <ul style="list-style-type: none"> - 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. <ul style="list-style-type: none"> - In the case of JMS : javax.jms.Topic or javax.jms.Queue |
| message-destination-ref | | Defines information related to a Destination reference. It can be specified more than once. Note: This can be set in deployment descriptor files of J2EE1.4 and higher. In J2EE1.3 and lower, specify the JMS Destination setting in resource-env-ref. |
| | description | Specifies optional information to be provided to the user. It can be omitted. |

| Tag | | Explanation |
|---------------------|------------------------------|---|
| | message-destination-ref-name | Specify a name that has the following Prefix for the Destination reference ('xxxxx' can be any char string). - jms/xxxxx |
| | message-destination-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. - javax.jms.Topic - javax.jms.Queue |
| | message-destination-usage | Specify one of the following as the method to use resources in the application for the user. - For receiving messages: Consumes - For sending messages: Produces - For sending and receiving messages: ConsumesProduces |
| | message-destination-link | Specify the JMS Destination name that was specified for the message-destination message-destination-name. It can be omitted. |
| message-destination | | Defines the Destination reference correspondence relationship. This definition is mandatory when message-destination-link is specified. It can be specified more than once. Note: This can be set in deployment descriptor files of J2EE1.4 and higher. In J2EE1.3 and lower, specify the JMS Destination setting in resource-env-ref. |
| | description | Specifies optional information to be indicated to the user. It can be omitted. |
| | message-destination-name | Specify the JMS Destination name. |
| 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. In any of the following cases if the env-entry-value tag is omitted from the Web application, if a Web application is not used javax.naming.NamingException or its sub class is thrown on lookup. - When env-entry-type is java.lang.Boolean: Boolean.FALSE - When env-entry-type is one of the following: All objects whose value is 0 |

| Tag | Explanation |
|-----|---|
| | <ul style="list-style-type: none"> - java.lang.Byte - java.lang.Character - java.lang.Short - java.lang.Integer - java.lang.Long - java.lang.Float - java.lang.Double - When env-entry-type is java.lang.String: javax.naming.NamingException or its sub class is thrown on lookup. |

Note

Do not specify tags that have namespace prefixes.

If this type of tag is specified, the deployment may fail, and it may not be possible to use the name conversion functionality.

Example

```
<prefix:ejb-ref>
```

message-destination-ref and message-destination

- In the message-destination-ref-name tag, specify the name in the argument of the JNDI lookup method in the application. In the following example, "jms/MyTopic" is specified in the message-destination-ref-name tag for lookup in the application.

```
ctx.lookup("java:comp/env/jms/MyTopic")
```

- In the message-destination-link tag, for the message-destination tag specify the value in the message-destination-name tag. This is not mandatory.
- The JMS Destination name can be specified in the message-destination message-destination-name tag. If the Destination contained in the same ejb-jar application is the target, when the lookup name is different, the defining of message-destination to link from message-destination-ref makes it possible to change the lookup name or to localize when the Destination name changes.

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"]

J2EE 1.4 and higher

```
<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>
<message-destination-ref>
  <description>JMS Information2</description>
  <message-destination-ref-name>jms/JMS2</message-destination-ref-name>
  <message-destination-type>javax.jms.Topic</message-destination-type>
  <message-destination-usage>Consumes</message-destination-usage>
  <message-destination-link>Topic001</message-destination-link>
</message-destination-ref>
...
<message-destination>
  <description>JMS Destination</description>
  <message-destination-name>Topic001</message-destination-name>
</message-destination>
```

J2EE 1.3 and lower

```
<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>
```

Point

.....

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>
```



Note

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.
- If the EJB application uses both the Home interface and the Local Home interface to refer to the same EJB application, the Local Home interface has been returned.

For the user response to this, refer to restriction 15, 'When the lookup via Remote interface is done in the EJB application that implements both Local and Remote interfaces from the same EJB application' under Restrictions on the EJB Service in 'Operating J2EE Applications', Restrictions of the Product Notes manual.

- The Local Home interface is returned in the following cases:
 - When EJB applications are referenced
 - When the referenced EJB applications are implemented in the Home interface and the Local Home interface

To perform lookup processing using the Local Home interface and the Home interface, the work shown below must be completed to develop and run the EJB applications.

Developing EJB applications

1. Edit the deployment descriptor

Edit the 'EnterpriseBean reference name' configured in "ejb-ref-name" of the [EJB Reference Tag] or the [Local EJB Reference Tag] so that it is unique.

2. Develop the EJB application

Ensure that the EJB application names specified in the argument for lookup are different for the Local Home interface and the Home interface. Enter an EJB application name for the EnterpriseBean reference name configured in the EJB Reference or Local EJB Reference set in step 1.

Running EJB applications

- 1. Edit the name conversion file

Use the name conversion file for the EnterpriseBean reference name changed in step 1 and the EJB application name specified in the argument for lookup in the EJB application.

3.10 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) { }

```

3.11 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. For details on this file, refer to "[3.11.2 interstage.xml File](#)". If there is a Web application anywhere on the server, however, the interstage.xml file cannot be edited directly. After the application is deployed, use the Interstage Management Console to set the 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.



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.)

Solaris32/64 Linux32/64

J2EE common directory/ijserver/[IJSERVER name]/apps/[module name]/META-INF/interstage.xml

(The default J2EE common directory location is /opt/FJSVj2ee/var/deployment.)

For J2EE application client

Locate the name conversion file in the following directory and specify the file name in an environment property. For details, refer to "3.11.1 Name Conversion File".

Windows32/64

C:\Interstage\J2EE\etc

Solaris32/64

/etc/opt/FJSVj2ee/etc

Linux32/64

/etc/opt/FJSVj2ee/etc



Note

When reference object information is not written to the deployment descriptor file, the name conversion function cannot be used.

3.11.1 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>
    <message-destination-entry>
      <message-destination-name>
        message-destination-name
      </message-destination-name>
      <jndi-name>jndi-name</jndi-name>
    </message-destination-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 |
|--------------------------------|--------------------------------------|
| <code><</code> | <code>&lt;</code> |
| <code>></code> | <code>&gt;</code> |
| <code>&</code> | <code>&amp;</code> |

- 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 |
|--|--|
| <code><client></code> | Specified for a J2EE application client. It can be specified more than once. |
| <code><app-name></code> | Specifies a name-converted application name. |
| <code><ejb-ref-entry></code> | 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. |
| <code><ejb-ref-name></code> | Specifies a deployment descriptor reference name. |
| <code><jndi-name></code> | Specifies an EJB application name (operation environment real name) corresponding to <code><ejb-ref-name></code> . |
| <code><res-entry></code> | 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. |
| <code><res-ref-name></code> | Specifies a deployment descriptor reference name. |
| <code><datasource-name></code> | Specifies a resource access definition name (operation environment real name) corresponding to <code><res-ref-name></code> . |
| <code><res-env-entry></code> | 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. |
| <code><res-env-ref-name></code> | Specifies a deployment descriptor reference name. |
| <code><environment-name></code> | Specifies a resource access definition name (operation environment real name) corresponding to <code><res-env-ref-name></code> . |
| <code><message-destination-entry></code> | Defines the JMS Destination name conversion. It can be specified more than once. Each time this is defined, the <code><message-destination-name></code> and <code><jndi-name></code> tags must both be specified. |
| <code><message-destination-name></code> | Specifies a deployment descriptor reference name. |
| <code><jndi-name></code> | Specifies a JMS Destination definition name (the real name of the operating environment) for <code><message-destination-name></code> . |

Note: If the reference name is duplicated, the name that was defined last is used.

Description Example (for J2EE Application Client)

A description example of an 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>
```

3.11.2 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>
    <app-name>app-name</app-name> or <jndi-name>jndi-name</jndi-name> or
    <module-name>module-name</module-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>
```

```

<message-destination-entry>
  <message-destination-name>
    message-destination-name
  </message-destination-name>
  <jndi-name>jndi-name</jndi-name>
</message-destination-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, <jndi-name> or <module-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 an EJB application. It can be specified more than once. |
| <app-name> | Specifies a name-converted application name in the case of a Web application. |
| <jndi-name> | Specifies a converted EJB application name in the case where <ejb-ref-entry>, <res-entry>, and <res-env-entry> are specified in the EJB application. |
| <module-name> | Specify the EJB module that performs name conversion when <message-destination-entry> is specified in the 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 an 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. |

| Tag | Explanation |
|-----------------------------|--|
| <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>. |
| <message-destination-entry> | Defines the JMS Destination name conversion. It can be specified more than once. Each time this is defined, the <message-destination-name> and <jndi-name> tags must both be specified. |
| <message-destination-name> | Specifies a deployment descriptor reference name. |
| <jndi-name> | Specifies a JMS Destination definition name (the real name of the operating environment) for <message-destination-name>. |

Note: If the reference name is duplicated, the name that was defined last is used.

Description Example (for Web Application)

A description example of an 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 an 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]

An example description in the interstage.xml file to perform name conversion in the EJB application is shown below. The deployment module name is "CatalogEJB.jar", and the EJB application names are "OperationBean" and "EmployeeBean".

```

<?xml version="1.0"?>
<!DOCTYPE fujitsu-ebe-definition SYSTEM 'fujitsu-ebe-definition.dtd'>
<fujitsu-ebe-definition >
  <ejb>
    <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>
    <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>
  <ejb>
    <module-name>CatalogEJB.jar</module-name>
    <message-destination-entry>
      <message-destination-name>Topic001</message-destination-name>
      <jndi-name>CatalogTopic</jndi-name>
    </message-destination-entry>
  </ejb>
</fujitsu-ebe-definition>

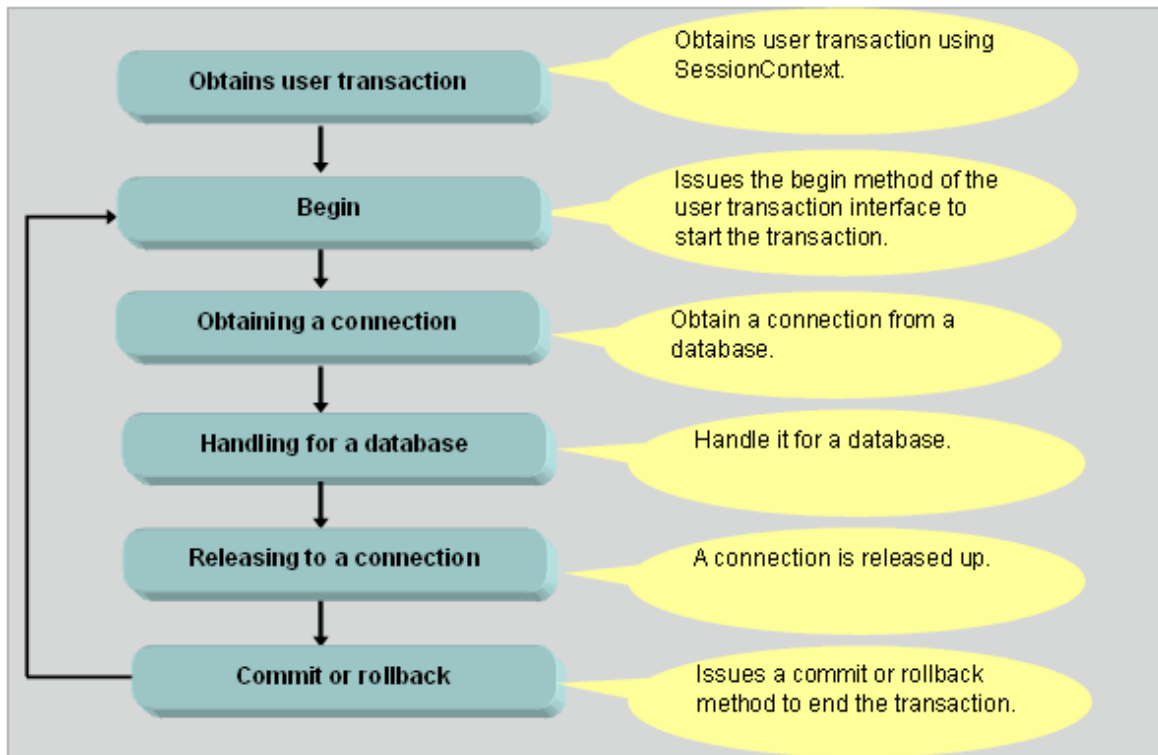
```

3.12 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

The flow of the transaction function processing when the UserTransaction interface is used is shown below.



Methods of the UserTransaction Interface

The UserTransaction interface uses the following methods.

The methods that can be used is shown below.

| 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. |
| Windows32/64 Solaris32 Linux32/64 | Changes the timeout associated with a transaction. Started by the begin method in the current thread. |
| setTransactionTimeout | |

Note

The `setTransactionTimeout` method of the `UserTransaction` interface is only valid if a distributed transaction is used. If this method is used in a local transaction, the value that is specified is invalid.

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(javax.transaction.HeuristicMixedException e ) {
    /* If HeuristicMixedException occurs, execute commit or rollback.*/
```

```

        userTransaction.rollback();
        throw e;
    } catch( java.lang.IllegalStateException e ) {
        userTransaction.rollback();
        throw e;
    } catch( java.lang.SecurityException e ) {
        userTransaction.rollback();
        throw e;
    } catch( javax.transaction.SystemException e ) {
        userTransaction.rollback();
        throw e;
    } catch( Throwable e ) {
        throw e;
    }
    ...

```

Note

If `javax.transaction.HeuristicMixedException` occurs in `UserTransaction` interface commit processing, perform a commit or rollback. If a commit or rollback is not performed, the transaction is considered to still have a "starting" status.

If the exceptions shown below occur in `UserTransaction` interface commit processing, perform a rollback. If a rollback is not performed, the transaction is considered to still have a "starting" status.

If the `UserTransaction` object is re-used, `javax.transaction.NotSupportedException` may occur if the `begin()` method is issued while the transaction still has "starting" status.

- `java.lang.IllegalStateException`
- `java.lang.SecurityException`
- `javax.transaction.SystemException`

3.13 J2EE Application Client Deployment Descriptor File Detailed Setup

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.

J2EE 1.4 deployment descriptor

```

<?xml version="1.0" encoding="UTF-8"?>
<application-client version="1.4"
xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/application-client_1_4.xsd">
    <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>

```

J2EE 1.3 deployment descriptor

```

<?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>

```

- In the J2EE1.4 deployment descriptor, the <?xml...> tag must be at the beginning of the deployment descriptor file for the description of the XML declaration.

In the J2EE1.3 deployment descriptor, 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.

- Do not specify tags that have namespace prefixes.

If this type of tag is specified, it may not be possible to use the name conversion functionality.

Example: <px:ejb-ref>

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. |

For details on the tags related to object reference (listed below), refer to "[3.9 Description in Deployment Descriptor File](#)".

- env-entry
- ejb-ref
- resource-ref
- resource-env-ref
- service-ref
- message-destination-ref
- message-destination

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.

4.1 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

4.1.1 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 Interstage Directory Service (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 Directory Service is not included in the Windows(64 bit) and Linux(64 bit) Interstage Application Server Standard-J Edition. It must be set up separately.

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.

4.1.2 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.

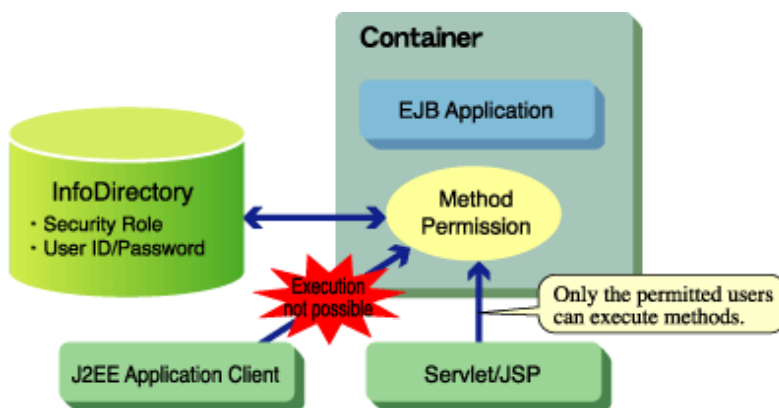
INTEGRAL and CONFIDENTIAL allow access with SSL used. For example "allow access with SSL enabled" is possible.

4.1.3 Method Permissions

Method permissions restrict access to EJB application methods.

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.



Method permissions use information about users, therefore the J2EE application client or a Web application must authenticate the user.

4.1.4 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.

4.1.5 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>
```

4.1.6 Run-as Security Function

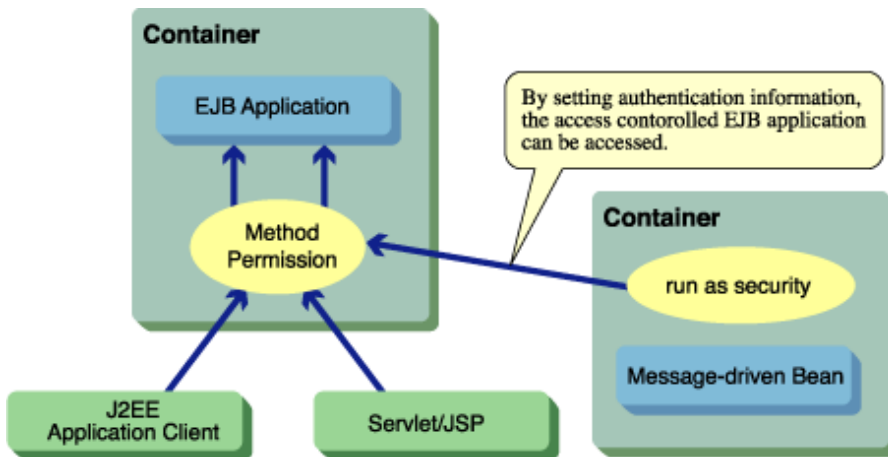
The run-as security function is a function that can specify the authentication information for the EJB application.

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.



The following settings are needed to use the run-as security function:

- Deployment descriptor setting
- User ID and password setting
- Directory service setting

Deployment Descriptor Setting

The run-as security settings (configuration details) can be specified in the deployment descriptor. The security role name needs to be specified in the run-as tag of the 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 Interstage Studio or the Interstage Management Console.



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 "[1. Directory Service Setting](#)".

Note

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

4.2 Embedding the Security Function

This section describes how to install the security functions.

1. Directory Service setting
2. Setting for each application

1. Directory Service Setting

The setting procedure of Directory Service is explained as follows.

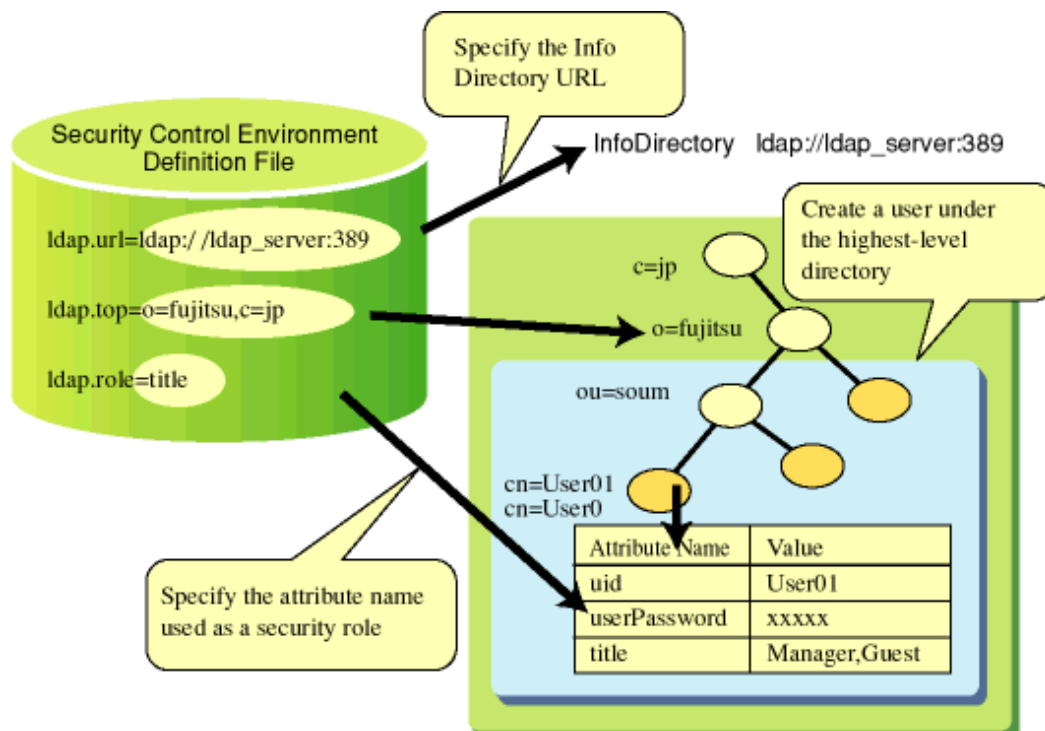
1. Setting Up the Security Management Environment Definition Files
2. User and Security Role Settings
3. Directory Service Work Procedure

4.2.1 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

The following figure shows the relationship between the security management environment definition file and Directory Service.



The Filename of Security Management Environment Definition Files

Installation of the J2EE package sets the security management environment definition files with the following filenames:

Windows32/64

```
C:\Interstage\J2EE\etc\security.properties
```

Solaris32/64 **Linux32/64**

```
/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 the table below.

| 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.

```
ldap.url=ldap://ldap_server:389  
ldap.top=o=fujitsu,c=jp  
ldap.role=title
```

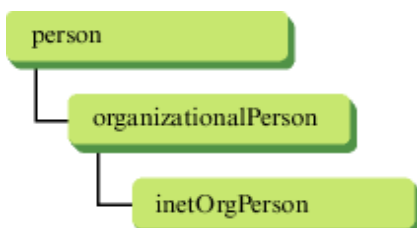
When any definition item in the security management environment definition file is changed, restart each container.

4.2.2 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 the following table to the user ID, password, and security role.

| 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.

4.2.3 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 Interstage Directory Service work procedures are shown below.

Interstage Directory Service Work Procedure

Refer to the Interstage Directory Service Operator's Guide for details of the Interstage Directory Service function.

[Operation with Interstage Management Console]

1. Create a repository.
2. Start the repository.

[Operation with Entry Administration Tool]

1. Set the connection repository.
2. Log in to the repository.
3. 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 a New Repository] 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.

Windows32/64

Execute the C:\Interstage\bin\irepeditent command to display the Entry Administration Tool.

Solaris32/64 Linux32/64

In the X window operating environment, execute 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 ldap.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:

| Item | Description |
|----------------|---|
| 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 |

| Item | Description |
|--------------|---|
| 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.

4.2.4 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)



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 "[4.1.5 Resource-connectable User Control Function](#)".

4.2.5 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



Example

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 (*7). 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.



Example

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 "[4.1.5 Resource-connectable User Control Function](#)".

4.2.6 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 Interstage Studio User's Guide.

An Interstage Management Console can also define a method permission.

Setting up the Resource-connectable User Control Function

Refer to the "[4.1.5 Resource-connectable User Control Function](#)".

4.3 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.



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 Java VM 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 the following table.

| Specification item | Description |
|--------------------|--|
| Log filename | <p>Specify the filename of the log.</p> <p>If it is specified with a relative path, it is interpreted as a relative path from the current directory on which Java VM is running.</p> <p>If the log filename is not specified or if a directory name is specified as a log filename, logs are not collected.</p> <p>Note</p> |

| Specification item | Description |
|--------------------|--|
| | Specify a different log filename for each Java VM to be started. If the same file is specified, logs output from two or more Java VMs may be mixed or partial loss may occur. |
| 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.

| 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"
```

4.4 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 J2SE



See

When dealing with errors, refer to the following information as required:

- For details of security management environment definition files, refer to "[4.2.1 Setting Up the Security Management Environment Definition Files](#)".
- For details of Interstage Directory Service errors, refer to Messages Beginning with 'irep', or Messages Output by Interstage Directory Service in the Interstage Messages manual.
- For details of trace function security function, refer to "[4.3 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 2 Servlet/JSP Edition

| | |
|---|-----|
| Chapter 5 Functions of the Servlet Service..... | 181 |
| Chapter 6 Web Application Development..... | 182 |
| Chapter 7 How to Call Web Applications..... | 214 |
| Chapter 8 Session Recovery..... | 219 |

Chapter 5 Functions of the Servlet Service

This chapter describes the functions of the Servlet Service.

- Session Recovery Function

5.1 Session Recovery Function

If the Servlet container or server crashes, session recovery can be used by a running Servlet container to inherit the session information and allow the application to continue.

If the Servlet container or server is balanced in environments on more than one server, session recovery can be used to inherit and use session information if an error occurs.

The session information is managed in Web applications in the Servlet container.

If a Servlet container crashes, the session information stored in the Servlet container is destroyed.

If session recovery is enabled, the session information created in each Servlet container can be backed up in the Session Registry Server, and session information can be recovered and inherited by a running Servlet container so that processing can continue.



Note

- Session recovery can be used in the following products. It cannot be used in Web Package.

- Session Registry Server

Interstage Application Server Enterprise Edition

- IJServer/Session Registry Client

Interstage Application Server Enterprise Edition

Interstage Application Server Standard-J Edition

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:

- Configuring the Web Application Directory
- Notes on the Development of Web Applications
- Web Application Environment Definition File (Deployment Descriptor)

Refer to "Debugging Applications" in the "Operating J2EE Applications" chapter for details of debugging.

6.1 Configuring the Web Application Directory

The section explains the configuration of the Web application directory. The Web application files are created under the root directory (the "Web application root directory") according to the directory configuration shown in the table shown.

| Directory name | Explanation |
|--|---|
| Directory under the root directory or directories except WEB-INF | Stores HTML files, image files, and JSP files. The directory can be created freely. |
| WEB-INF | Stores Web application environment definition files (deployment descriptor)(web.xml). |
| WEB-INF/classes | Stores Java class files such as servlets. |
| WEB-INF/lib | Stores JAR files that are Java class files compressed in JAR format. |



Note

If there are Java class files with the same name as JAR files under "WEB-INF/classes" and "WEB-INF/lib", Java class files under "WEB-INF/classes" have priority.

6.2 Notes on the Development of Web Applications

This chapter describes the points to be noted when developing Web applications.

6.2.1 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.

6.2.2 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.

6.2.3 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.

6.2.4 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 "[6.3 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).

6.3 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(XML schema base) and is shown in the following example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4">
  <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>
    <dispatcher>value</dispatcher>
  </filter-mapping>
```

```

<filter-mapping>
  <filter-name>value</filter-name>
  <url-pattern>pattern</url-pattern>
  <dispatcher>value</dispatcher>
</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>
<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-type>entry-type</env-entry-type>
  <env-entry-value>entry-value</env-entry-value>
</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>
<message-destination-ref>
  <description>description</description>
  <message-destination-ref-name>name</message-destination-ref-name>
  <message-destination-type>type</message-destination-type>
  <message-destination-usage>usage</message-destination-usage>
  <message-destination-link>link</message-destination-link>
</message-destination-ref>
<message-destination>
  <description>description</description>
  <message-destination-name>name</message-destination-name>
</message-destination>
<service-ref>
  <service-ref-name>name</service-ref-name>
  <service-interface>interface</service-interface>
  <wsdl-file>file</wsdl-file>
  <jaxrpc-mapping-file>file</jaxrpc-mapping-file>
</service-ref>
<locale-encoding-mapping-list>
  <locale-encoding-mapping>
    <locale>locale</locale>
    <encoding>encoding</encoding>
  </locale-encoding-mapping>
</locale-encoding-mapping-list>
<jsp-config>
  <taglib>
    <taglib-uri>uri</taglib-uri>
    <taglib-location>location</taglib-location>
  </taglib>
  <jsp-property-group>
    <url-pattern>pattern</url-pattern>
    <el-ignored>true or false</el-ignored>
    <page-encoding>value</page-encoding>

```

```
<scripting-invalid>true or false</scripting-invalid>
<is-xml>true or false</is-xml>
<include-prelude>uri</include-prelude>
<include-coda>uri</include-coda>
</jsp-property-group>
</jsp-config>
</web-app>
```

Note

Notes on Coding

- No definitions other than those described in this manual are supported. .
- `<?xml...>` and `< web-app xmlns...>` appearing at the beginning provide an XML declaration and schemaLocation, and must be stated at the beginning of the deployment descriptor file.
- If multi-byte characters are to be used in the deployment descriptor (including comments), specify UTF-8 for the encoding format ("encoding=") of `<?xml...>`.
- Provide individual tags in the order described above.
- Tags with a default value of "x" cannot be omitted. When omitted, the tag definition becomes invalid or causes an error.
- When it overlaps and a tag which cannot perform multiple specification is used, the tag specified at the end becomes invalid or causes an error.
- 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.
- The entry format for the deployment descriptor of Servlet 2.2 or 2.3 can also be used.
- When the entry format of the deployment descriptor of the application that has already been deployed is changed from that for Servlet 2.2 or 2.3 to Servlet 2.4, the application must be redeployed.
- Do not specify tags that have namespace prefixes. If this type of tag is specified, the deployment may fail, and it may not be possible to use the name conversion functionality.
- Example: `<pfx:servlet>`

Windows32/64

- The following characters can be used in pathnames:
Alphanumeric characters, '+', '-', '_', '.', '\$', '%', '/', and '~'
- Pathnames can be up to 255 bytes long.

Solaris32/64 Linux32/64

- 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

| 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 |
| 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 | Possible |
| error-page | Defines resources corresponding to error codes and Java exception types. | 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 |
| locale-encoding-mapping-list | Defines the mapping of the locales and the character code | Optional | Possible |
| jsp-config | Defines the values shared by JSPs in the Web application | Optional | Impossible |



See

For details about the following tags, refer to "Description in deployment descriptor file" in "JNDI" of the "J2EE User's Guide".

- resource-env-ref
- resource-ref
- env-entry
- ejb-ref
- ejb-local-ref
- message-destination-ref
- message-destination
- service-ref



Point

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.

In the case of a Windows(R) system, please read a path suitably.

6.3.1 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 xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
  http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
  version="2.4">
  ...
</web-app>
```

Entry Example

```
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
  http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
  version="2.4">
  <context-param>
  ...
  </context-param>
</web-app>
```

6.3.2 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

```
<display-name>Example Security Constraint</display-name>
```

6.3.3 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.

Duplicate initialization parameter names cannot be defined.

Entry Format

```
<context-param>
  <param-name>name</param-name>
  <param-value>value</param-value>
</context-param>
```

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

```
<context-param>
  <param-name>E-mail</param-name>
  <param-value>taro@fujitsu.co.jp</param-value>
</context-param>
```

Web application coding example

```
public doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    String mail = getServletContext().getInitParameter("E-mail");
}
```

6.3.4 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

| Tag | Description | Tag requirement | Multiple specification |
|--------------|---|-----------------|------------------------|
| filter-name | Alias of the filter class. Specify a unique name. This alias is used to find matches of the servlet name or URI pattern defined by filter-mapping. | Required | Impossible |
| filter-class | Specify the fully qualified name of a mapped filter class. | 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 "[6.3.5 Filter Class Application Target](#)".

6.3.5 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>
  <dispatcher>value</dispatcher>
</filter-mapping>
<filter-mapping>
  <filter-name>value</filter-name>
  <url-pattern>pattern</url-pattern>
  <dispatcher>value</dispatcher>
</filter-mapping>
```

Tag Details

| Tag | Description | Tag requirement | Multiple specification |
|-------------|-------------------------------|-----------------|------------------------|
| filter-name | Specify a unique filter name. | Required | Impossible |

| Tag | Description | Tag requirement | Multiple specification |
|--------------|--|-----------------|--|
| | This name must match the value of a filter-name tag in a filter tag. | | |
| url-pattern | <p>Specify the URI pattern to be mapped with the filter.</p> <p>This tag cannot be used together with the servlet-name tag.</p> <p>The URL is entered as follows:</p> <ul style="list-style-type: none"> - 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 | <p>Specify the name of servlet to be mapped with the filter.</p> <p>This tag cannot be used together with the url-pattern tag.</p> <p>Enter the name specified in the servlet-name tag of the servlet tag as the servlet name.</p> <p>If an undefined servlet name is entered or the value is omitted, the definition becomes invalid or causes an error.</p> | Required | Impossible |
| Dispatcher | <p>The timing for calling the filter. Specify the following values:</p> <ul style="list-style-type: none"> - REQUEST (default value) The file cluster becomes valid in response to the client request. - FORWARD The filter class becomes valid when the forward() method of RequestDispatcher or JSP "forward" is activated. - INCLUDE The file cluster becomes valid during the dynamic "include" such as when the forward() method of RequestDispatcher or JSP "forward" is activated. - ERROR The file cluster becomes valid during an error page transition. | Optional | <p>Possible</p> <p>Four different values can be specified.</p> |

Entry Example

A filter definition for a specific Servlet is shown as follows:

```
<filter>
  <filter-name>helloWorld</filter-name>          ...alias of a filter class
```

```

<filter-class>MyHelloWorldFilter</filter-class> ...filter class name
  <init-param>
    <param-name>company</param-name>
    <param-value>Fujitsu Ltd.</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>helloWorld</filter-name>          ...alias of a filter class
  <servlet-name>MyHelloWorld</servlet-name>      ...servlet name
</filter-mapping>

```

A filter definition for a specific URL (the URL of the JSP file in the following example) is shown as follows:

```

<filter>
  <filter-name>helloWorld</filter-name>          ...alias of a filter class
  <filter-class>MyHelloWorldFilter</filter-class> ...filter class name
  <init-param>
    <param-name>company</param-name>
    <param-value>Fujitsu Ltd.</param-value>
  </init-param>
</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>

```

A filter definition for a URI pattern is shown as follows. This example uses a wildcard to specify that all resources under "/jsp/" be subjected to the filter function.

```

<filter>
  <filter-name>helloWorld</filter-name>          ...alias of a filter class
  <filter-class>MyHelloWorldFilter</filter-class> ...filter class name
  <init-param>
    <param-name>company</param-name>
    <param-value>Fujitsu Ltd.</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>helloWorld</filter-name>          ...alias of a filter class
  <url-pattern>/jsp/*</url-pattern>             ...filter and the URI pattern to map
</filter-mapping>

```

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 {
        HttpServletResponse 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);
        }
    }
}

```

6.3.6 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

| Tag | Description | Tag requirement | Multiple specification |
|----------------|---|-----------------|------------------------|
| listener-class | Specifies the complete class name for the following events: <ul style="list-style-type: none"> - 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) | Required | Impossible |

| Tag | Description | Tag requirement | Multiple specification |
|-----|---|-----------------|------------------------|
| | <ul style="list-style-type: none"> - Start/end a request (javax.servlet.ServletRequestListener interface implementation class) - Add/replace/delete attributes to a request (javax.servlet.ServletRequestAttributeListener interface implementation class) <p>If a nonexistent class name is specified, the Web application fails to start.</p> | | |

Entry Example

```
<listener>
  <listener-class>listeners.ContextListener</listener-class>
</listener>
```

6.3.7 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.

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

| Tag | Description | Tag requirement | Multiple specification |
|-------------------|---|---|------------------------|
| servlet-name | Defines the name of the servlet or JSP. This value is also used when the servlet name is specified using the servlet-mapping tag or the filter-mapping tag (*1). | 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.  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 |
| load-on-startup | Defines the startup when Servlet Container is started. Order which loads Servlet and JSP. It defines by -2147483648 to 2147483647. Non-numeric values cannot be used. Loading proceeds in order from the smallest number to the largest. If 0 is specified, the relevant servlet or JSP is loaded first. 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 first. 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 If the same value is defined more than once, then the order of loading within the same value cannot be guaranteed. | 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 |

| Tag | Description | Tag requirement | Multiple specification |
|-----------|---|---|------------------------|
| 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 |

*1 Provided [Servlet runs without mapping] is enabled, access from the Web browser is possible using the servlet name. In this case, alphanumeric characters, '+', '-', '.', '_', '\$' can be specified.

Entry Example

When Defining a Servlet

```
<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>
<security-role>
  <role-name>Manager</role-name>
</security-role>
```

When Defining a JSP File

```
<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>
<security-role>
  <role-name>Manager</role-name>
</security-role>
```

6.3.8 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

| Tag | Description | Tag requirement | Multiple specification |
|--------------|---|-----------------|------------------------|
| servlet-name | <p>Defines the servlet or JSP name to which a request is to be mapped.</p> <p>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.</p> | Required | Impossible |
| url-pattern | <p>Defines the URL mapped to the servlet or JSP application.</p> <p>The URL is entered as follows:</p> <ul style="list-style-type: none"> - 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:

```
<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>
```

A mapping definition for a request where the URL path information has prefix "director" is shown as follows:

```
<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>
```

A mapping definition for a request where the URL ends with ".do" is shown as follows:

```
<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>
```

6.3.9 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

| 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

```
<session-config>
  <session-timeout>30</session-timeout>
</session-config>
```

6.3.10 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

| 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

```
<mime-mapping>
  <extension>jpg</extension>
  <mime-type>image/jpeg</mime-type>
</mime-mapping>
```

Default Mime Type

| Extension | Mime type |
|-----------|----------------------------|
| abs | audio/x-mpeg |
| ai | application/postscript |
| aif | audio/x-aiff |
| aifc | |
| aiff | |
| aim | application/x-aim |
| art | image/x-jg |
| asf | video/x-ms-asf |
| asx | |
| 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 |

| Extension | Mime type |
|--------------------|----------------------------------|
| css | text/css |
| dib | image/bmp |
| doc | application/msword |
| dtd | application/xml-dtd |
| 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 |
| ico | image/x-icon |
| 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/midi |
| latex | application/x-latex |
| m3u | audio/x-mpegurl |
| mac | image/x-macpaint |
| man | application/x-troff-man |
| mathml | application/mathml+xml |
| me | application/x-troff-me |
| mid | audio/midi |
| midi | audio/midi |
| mif | application/vnd.mif |

| Extension | Mime type |
|-----------------------------|--|
| mov | video/quicktime |
| movie | video/x-sgi-movie |
| mp1 mpa | audio/x-mpeg |
| mp2 mp3 | audio/mpeg |
| mpe mpeg mpega mpg | video/mpeg |
| mpv2 | video/mpeg2 |
| ms | application/x-troff-ms |
| nc | application/x-netcdf |
| oda | application/oda |
| odb | application/vnd.oasis.opendocument.database |
| odc | application/vnd.oasis.opendocument.chart |
| odf | application/vnd.oasis.opendocument.formula |
| odg | application/vnd.oasis.opendocument.graphics |
| odi | application/vnd.oasis.opendocument.image |
| odm | application/vnd.oasis.opendocument.text-master |
| odp | application/vnd.oasis.opendocument.presentation |
| ods | application/vnd.oasis.opendocument.spreadsheet |
| odt | application/vnd.oasis.opendocument.text |
| ogg | application/ogg |
| otg | application/vnd.oasis.opendocument.graphics-template |
| otg | application/vnd.oasis.opendocument.text-web |
| otp | application/vnd.oasis.opendocument.presentation-template |
| ots | application/vnd.oasis.opendocument.spreadsheet-template |
| ott | application/vnd.oasis.opendocument.text-template |
| 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 |

| Extension | Mime type |
|------------------|-------------------------------|
| ppm | image/x-portable-pixmap |
| pps | application/vnd.ms-powerpoint |
| ppt | application/vnd.ms-powerpoint |
| ps | application/postscript |
| psd | image/x-photoshop |
| qt | video/quicktime |
| qti | image/x-quicktime |
| qtif | |
| ras | image/x-cmu-raster |
| rdf | application/rdf+xml |
| rgb | image/x-rgb |
| rm | application/vnd.rn-realmedia |
| roff | application/x-troff |
| rtf | text/rtf |
| rtx | text/richtext |
| sh | application/x-sh |
| shar | application/x-shar |
| smf | audio/x-midi |
| sit | application/x-stuffit |
| snd | audio/basic |
| src | application/x-wais-source |
| sv4cpio | application/x-sv4cpio |
| sv4crc | application/x-sv4crc |
| svg | image/svg+xml |
| svgz | image/svg |
| swf | application/x-shockwave-flash |
| t | application/x-troff |
| tar | application/x-tar |
| tcl | application/x-tcl |
| tex | application/x-tex |
| texi | application/x-texinfo |
| texinfo | |
| tif | image/tiff |
| tiff | |
| tr | application/x-troff |
| tsv | text/tab-separated-values |
| txt | text/plain |
| ulw | audio/basic |
| ustar | application/x-ustar |

| Extension | Mime type |
|------------|---------------------------------|
| vrml | model/vrml |
| vsd | application/x-visio |
| vxml | application/voicexml+xml |
| 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 | model/vrml |
| xbm | image/x-xbitmap |
| xht | application/xhtml+xml |
| xhtml | application/xhtml+xml |
| xls | application/vnd.ms-excel |
| xml | application/xml |
| xpm | image/x-xpixmap |
| xsl | application/xml |
| xslt | application/xslt+xml |
| xul | application/vnd.mozilla.xul+xml |
| xwd | image/x-xwindowdump |
| Z | application/x-compress |
| z | |
| zip | application/zip |

6.3.11 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 whether the value specified in [WorkUnit] > [WorkUnit name] > [Settings] > [Servlet Container Settings] > [Display Directory Listing if Index File not Present] is On or Off 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

| Tag | Description | Tag requirement | Multiple specification |
|--------------|-----------------------|-----------------|------------------------|
| welcome-file | Defines welcome file. | Required | Possible |

Entry Example

```
<welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
  <welcome-file>index.htm</welcome-file>
</welcome-file-list>
```

6.3.12 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

| 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 |
| 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 a definition error. | Required | Impossible |

| Tag | Description | Tag requirement | Multiple specification |
|-----|---|-----------------|------------------------|
| | <p>Windows32/64</p> <p>When entering directories in the partial pathname, separate each directory name with a slash (/), not a backslash (\).</p> <p>Note) An error page with a built-in web browser may be displayed if a web browser is set.</p> | | |

Either an error-code tag or an exception-type tag is defined.

When both are not specified, this error-page tag becomes a definition error

Entry Example

For HTTP Error

```
<error-page>
  <error-code>500</error-code>
  <location>/error/http/code500.html</location>
</error-page>
```

For Java Exception

```
<error-page>
  <exception-type>java.lang.IllegalStateException</exception-type>
  <location>/error/exception/IllegalState.html</location>
</error-page>
```

6.3.13 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

| 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. | Required | Possible |

| Tag | Description | Tag requirement | Multiple specification |
|----------------------|---|-----------------|------------------------|
| | Define it using the target path from root directory of Web application. Add "/" to the top. | | |
| 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. When the role-name tag or value is omitted, no user can access. | Optional | Possible |
| 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

```

<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>

```

6.3.14 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

| Tag | Description | Tag requirement | Multiple specification |
|-------------------|---|-----------------|------------------------|
| auth-method | <p>Defines the authentication method.</p> <p>Specify one of the following:</p> <ul style="list-style-type: none"> - BASIC HTTP BASIC authentication - FORM Form-based authentication <p>If the auth-method tag is omitted, "BASIC" becomes the optional value. The value cannot be omitted.</p> | Optional | Impossible |
| realm-name | <p>Defines the area name used for HTTP Basic authentication. The area name is displayed on the screen for user authentication (dialog box).</p> <p>When the HTTP Basic authentication is not used, the specification is ignored.</p> <p>The specification must be made when the HTTP Basic authentication is used.</p> | Optional | Impossible |
| form-login-config | <p>Defines the start and end of the form-based authentication definitions.</p> <p>Specify the login page and error page used for form-based authentication.</p> <p>When form-based authentication is not used, the specification is ignored.</p> <p>The specification must be made when the form-based authentication is used.</p> | Optional | Impossible |
| form-login-page | <p>Defines the login page used for form-based authentication.</p> <p>It is required when the form-login-config tag is defined.</p> <p>The specified login page is displayed in the Web browser at form-based authentication.</p> <p>Note</p> <p>The login page must use the following interface to pass the user name and password to the Servlet Container:</p> <ul style="list-style-type: none"> - Application name: j_security_check - Parameter name -> user name: j_username - Parameter name -> password: j_password <p>Example</p> | Required | Impossible |

| Tag | Description | Tag requirement | Multiple specification |
|-----------------|---|---|------------------------|
| | : <FROM ACTION="j_security_check" METHOD="POST"> UserName: <INPUT TYPE="text" NAME="j_username"> Password: <INPUT TYPE="password NAME="j_password"> <input type="submit" value="send"></FORM> : When form-based authentication is specified with an auth-method tag, the log-in page must be specified using this tag. | | |
| 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

```
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Welcome Page</realm-name>
</login-config>
```

For Form-based authentication

```
<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>
```

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 "[6.3.9 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.

6.3.15 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

| 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

```
<security-role>
  <role-name>Manager</role-name>
</security-role>
```

6.3.16 Mapping of the Locales and the Character Code

Mapping of the locales and the character encoding must be defined using the locale-encoding-mapping-list tag.

Entry Format

```
<locale-encoding-mapping-list>
  <locale-encoding-mapping>
    <locale>locale</locale>
    <encoding>encoding</encoding>
  </locale-encoding-mapping>
</locale-encoding-mapping-list>
```

Tag Details

| Tag | Description | Tag requirement | Multiple specification |
|-------------------------|---|-----------------|------------------------|
| locale-encoding-mapping | Defines the mapping of the locales and character encoding | Required | Possible |
| locale | Specifies the locale defined in ISO-639-1 and ISO-3166 | Required | Impossible |
| encoding | Specifies the character code that is to be mapped for the locale. | Required | Impossible |

Entry Example

```
<locale-encoding-mapping-list>
  <locale-encoding-mapping>
    <locale>ja</locale>
    <encoding>Shift_JIS</encoding>
  </locale-encoding-mapping>
</locale-encoding-mapping-list>
```


6.3.17 Definitions Shared by JSPs in Web Application

The definition shared by JSPs in the Web application must be made using the jsp-config tag.

Entry Format

```
<jsp-config>
  <taglib>
    <taglib-uri>uri</taglib-uri>
    <taglib-location>location</taglib-location>
  </taglib>
  <jsp-property-group>
    <url-pattern>pattern</url-pattern>
    <el-ignored>true | false</el-ignored>
    <page-encoding>value</page-encoding>
    <scripting-invalid>true | false</scripting-invalid>
    <is-xml>true | false</is-xml>
    <include-prelude>uri</include-prelude>
    <include-coda>uri</include-coda>
  </jsp-property-group>
</jsp-config>
```

Tag Details

| Tag | Description | Tag requirement | Multiple specification |
|-----------------|--|-----------------|------------------------|
| taglib | Defines the tag library when embedded in a JSP | Optional | Possible |
| taglib-uri | Defines the URI of the JSP tag library used by the Web application. Specifies the URI to be defined in <taglib> of the JSP file. | Required | Impossible |
| taglib-location | <p>Defines the TLD (Tag Library Description) file name of the tag library</p> <p>Specified using the relative path from the Web application root directory. Adds "/" to the beginning of the path.</p> <p>When a path that does not exist is entered, the Web application fails to activate.</p>  | Required | Impossible |

| Tag | Description | Tag requirement | Multiple specification |
|--------------------|---|-----------------|------------------------|
| | To enter a directory in the relative path, '/' is used, instead of '\', between directory names. | | |
| jsp-property-group | Specifies the definitions and limitations for JSPs that meet the URL pattern set using the <url-pattern> tag | Optional | Possible |
| url-pattern | Specifies the URL pattern of JSPs to which <jsp-property-group> is applied | Required | Possible |
| el-ignored | Specifies whether or not to ignore the Expression Language in a JSP <ul style="list-style-type: none"> - true: Ignores the Expression Language (not an error) - false: Evaluates the Expression Language The default value is "false". | Optional | Impossible |
| page-encoding | Specifies the JSP file character code JSP compilation fails if the specified value is not the same as that specified in page Encoding of the "page" directive in the JSP file. | Optional | Impossible |
| scripting-invalid | Specifies whether or not to disallow the JSP script entry <ul style="list-style-type: none"> - true : Disallow the script entry JSP compilation fails if a script is entered in JSPs. - false: Allows the script entry When scripting-invalid is omitted, the script entry will be allowed. | Optional | Impossible |
| is-xml | Specifies whether or not to process the JSP file as JSP documents (XML syntax) <ul style="list-style-type: none"> - true: Processes the JSP file as JSP documents (XML syntax) JSP compilation fails if a JSP is not entered as the JSP document (XML syntax). - false: Processes a JSP as a regular JSP page (non-XML syntax) The default behavior is to process as JSP documents (XML syntax) if the file extension is ".jspx" | Optional | Impossible |
| include-pragma | Specifies the URI of the file to be included at the beginning of a JSP. Results in the same operation as when the "include" directive is specified at the beginning of a JSP. | Optional | Possible |
| include-coda | Specifies URI of the file to be included at the end of a JSP. Results in the same operation as when the "include" directive is specified at the end of a JSP. | Optional | Possible |

Example

```

<jsp-config>
  <taglib>
    <taglib-uri>http://www.fujitsu.com/interstage/jservlet/example-taglib</taglib-uri>
    <taglib-location>/WEB-INF/tld/example-taglib.tld</taglib-location>
  </taglib>
</jsp-config>

```

```
<url-pattern>/jsp/*</url-pattern>
<el-ignored>true</el-ignored>
<page-encoding>UTF-8</page-encoding>
<scripting-invalid>>false</scripting-invalid>
<is-xml>>false</is-xml>
<include-prelude>/WEB-INF/jsp/include_pre.jspf</include-prelude>
<include-coda>/WEB-INF/jsp/include_coda.jspf</include-coda>
</jsp-property-group>
</jsp-config>
```

Example of JSP file entry

```
<html>
<body>
<%@ taglib uri="http://www.fujitsu.com/interstage/jsp/jsp/example-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 the <tag-class> tag of the tag library descriptor file is changed, the JSP files that use the tag library corresponding to the changed tag library descriptor must be recompiled.

JSPs are recompiled when the java source file and the corresponding class file are not located in a work directory directly under the IJServer directory.

Therefore, when the java source file and the class file that correspond to the JSP file are deleted, the JSPs are recompiled.

For example, if the JSP file path from the Web application root directory is "/jsp/HelloJSP.jsp", the following is generated:

- Source file name: jsp\HelloJSP_jsp.java
- Class file name: jsp\HelloJSP_jsp.class

Chapter 7 How to Call Web Applications

This chapter describes how to call Web applications.

The definition examples are for the Solaris.

If you are using a Windows(R) system, change the path to one that is suitable for reading in a Windows(R) system.

7.1 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] > [Settings] > [Servlet Container Settings [Show]] and change the setting of [Run Servlet even if mapping is not present?].

7.1.1 Call that Requires Mapping

Servlet URL

The Servlet URL corresponds to the URL pattern defined in "6.3.8 Servlet Mapping" (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
```

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 & ...
```



Note

- 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.

Windows32/64

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)."

7.1.2 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
```

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 "6.3.7 Servlet Attributes" in "Web Application Development" for an explanation of alias settings. Below is an example using the servlet name and an example using an alias.



Example

Example using the servlet name

```
http://hostname/webap11/servlet/HelloWorldServlet
```

Example using an alias ("HelloWorldServlet" defined by the alias "Hello".)

```
http://hostname/webap11/servlet/Hello
```



Note

- 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.

Windows32/64

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)."

7.2 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.



Example

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
```

The port number can be omitted. In this case, 80 is used as the port number.



Example

```
http://hostname/webapl1/jsp/Hello/HelloJSP.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.

7.3 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.



Example

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
```

The port number can be omitted. In this case, 80 is used as the port number.



Example

```
http://hostname/webappl1/ap1/Hello/index.htm
```

Calling from within an HTML Document

```
<A HREF="/Web-application-name/Partial-pathname-file">Click Here</A>
```

Chapter 8 Session Recovery

This chapter describes session recovery in the Servlet service.

Session recovery can be used in the following products:

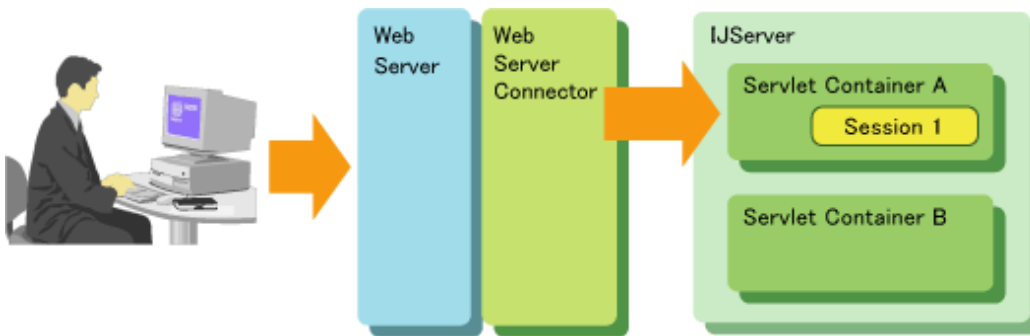
- Session Recovery (Session Registry Server)
 - Interstage Application Server Enterprise Edition
- Session Recovery (IJSERVER/Session Registry Client)
 - Interstage Application Server Enterprise Edition
 - Interstage Application Server Standard-J Edition

8.1 About Session Recovery

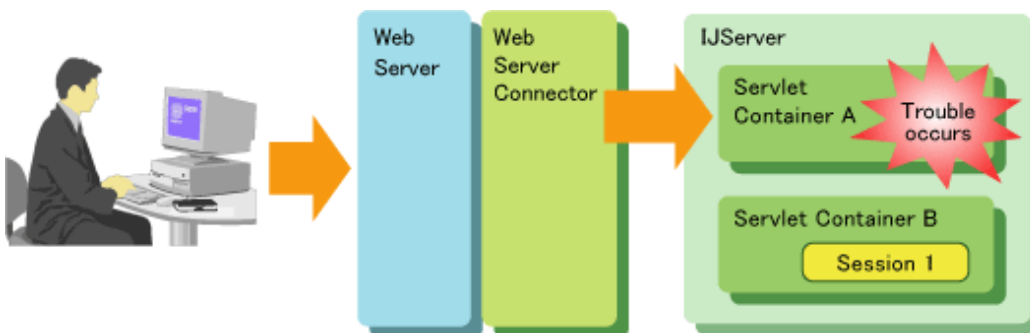
In the event that a Servlet container process or server crashes, Session recovery enables another Servlet container to inherit Servlet session information so that the Web application can continue.

If the IJSERVER is restarted, Servlet sessions that were running before the restart can continue without being destroyed.

Before an Error Occurs



After an Error Occurs



Session Recovery Configuration Elements

Session recovery is configured as shown below:

Session Registry Server

Session Registry Server is server software used to save Servlet session information used in the IJSERVER.

When an error occurs, information about the session before the error is used to recover the session so that the Web application can continue.

IJSERVER(Servlet container)

This is the executable environment for the Web application.

To allow the Web application to continue after an error occurs, process concurrency must be used, or the Web application must be set up in more than one server environment.

Session Registry Client

When session recovery is enabled, Session Registry Client runs as the existing session management module as an add-in to the IJServer (Servlet container). This is used for communication between session management and Session Registry Server.

Web server connector

The Web server connector forwards requests received by the Web server to the Servlet container.

Normally, if the Servlet session has requests from the client, the Web server connector distributes the requests to the Servlet container used to create the session. If, for some reason, the requests cannot be distributed to one Servlet container, they can be distributed to another so that the Web application can continue.

8.1.1 Session Backup

Sessions are backed up in Session Registry Server and the Servlet container. When an error occurs, the session backed up in Session Registry Server can be recovered and inherited in another Servlet container.

Session Backup Mode

The following types can be selected as the mode for backing up the session:

- [Request End Time]
- [Fixed Interval] (default value)

The backup mode is set using the following Interstage Management Console options in Session Registry Client. The default is [Fixed Interval]. The default backup interval is [5] seconds.

- [WorkUnit] > "WorkUnit Name" > [Environment Settings] > [Session Recovery Settings] > [Backup Mode]

For details about Session Registry Client settings, refer to "[8.4 Session Registry Client Settings](#)".

The backup modes have the following settings:

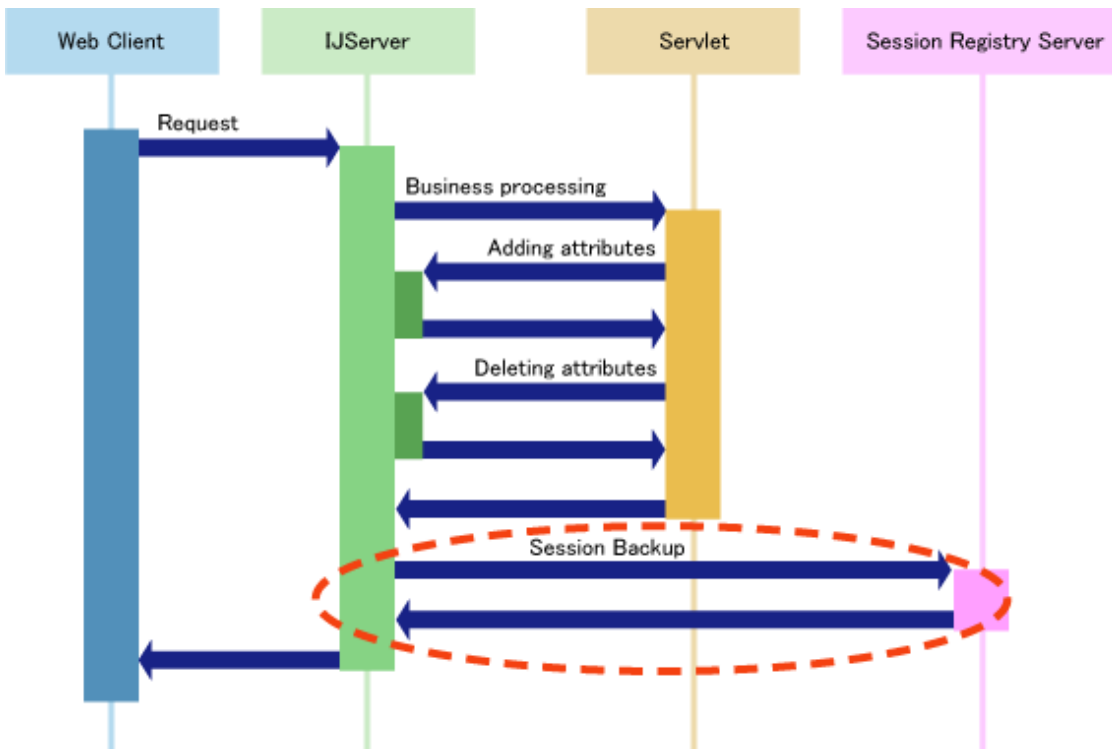
- Update of the session at the time of recovery (guarantee that a session of a newer status is backed up)
[Request End Time] > [Fixed Interval]
- Performance
[Fixed Interval] > [Request End Time]

The isj2eadmin command can be used to configure these settings.

For details about the isj2eadmin command, refer to "isj2eadmin" in the Reference Manual (Command Edition).

Request End Time

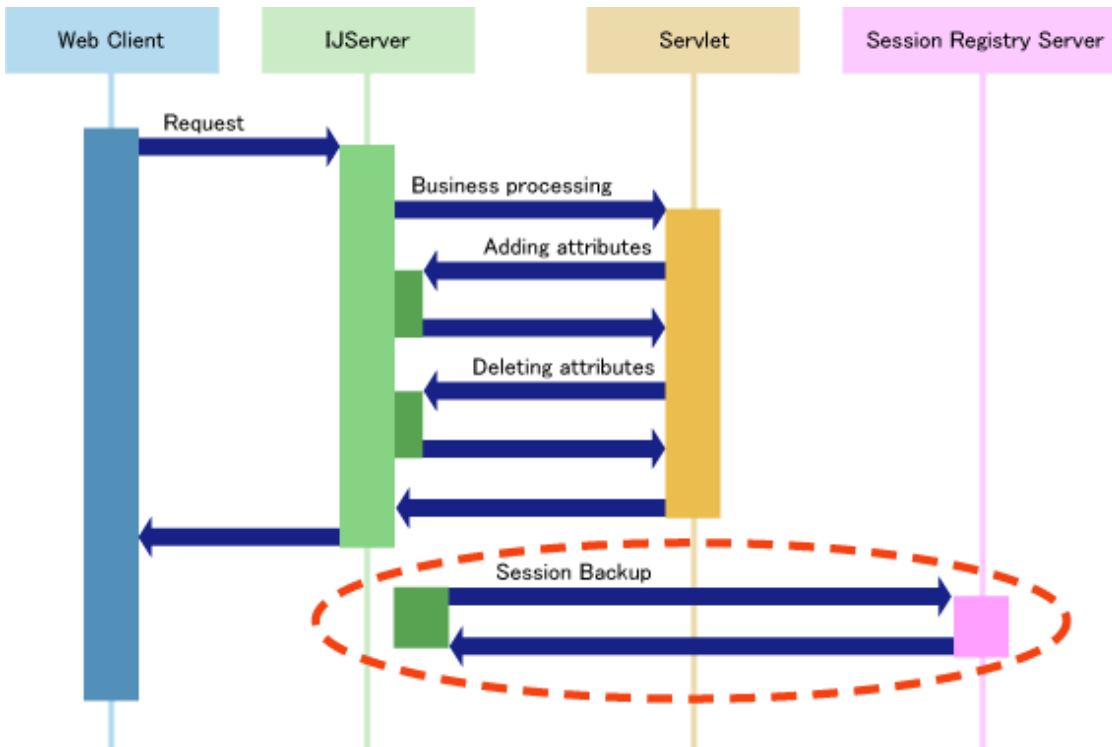
The session is backed up after requests are completed. The session waits until the completion of requests and the backup in Session Registry Server before responding to the client.



- The session backup time is also included in the processing time for requests from the client. The processing time can therefore be backed up and compared at fixed intervals.

Fixed Interval

The session is backed up regularly by the Servlet container after the completion of request processing.



- "Request processing" is backed up separately. Performance is not directly affected until the request is returned.
- If the Servlet container crashes before the backup in Session Registry Server, the session cannot be recovered.

If Session Backup Fails

No error is sent to the client or Web application if session backup fails. This is because notifying the client or Web application and using session recovery reduce the application usage ratio.

If session backup fails, Session Registry Server is marked as unusable, and there is no communication (backup or recovery) until it can be used again.

For details about mechanisms that can be used in Session Registry Server, refer to "8.1.4 Monitoring Session Recovery".

Processing Sessions for more than one Request

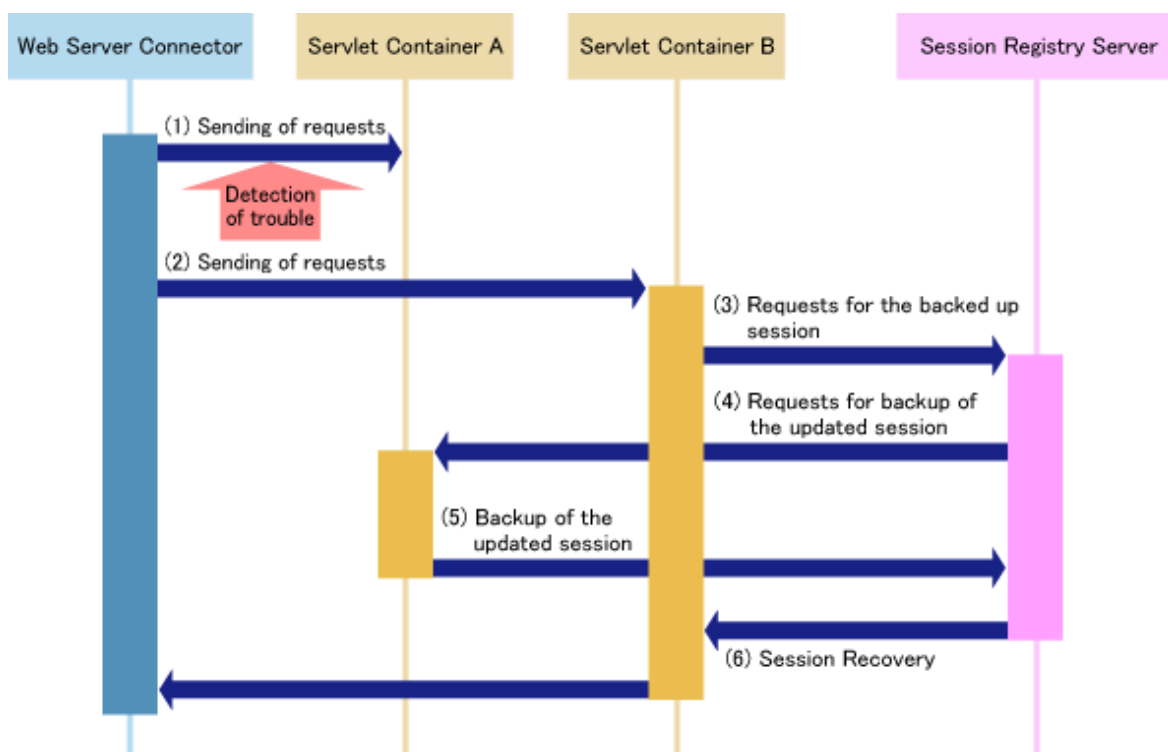
All requests handled in the same single session must be processed in the same JavaVM at the same time. Accordingly, if a session is recovered in another Servlet container the original Servlet container session must be disabled.

If a session is recovered in another Servlet container while Session Registry Server is enabled, the attempt to destroy the original Servlet container session may be unsuccessful because of a network problem. In this case, the original Servlet container session is destroyed when communication between the Servlet container and Session Registry Server is restored.

8.1.2 Session Recovery

If the Servlet container does not maintain the session corresponding to the session ID provided by the client (if, for example, the Servlet container that handled the original session crashed), a session backed up in another container is obtained from Session Registry Server and the requests are processed.

If there is a request from the Servlet container for the session, Session Registry Server simply returns the updated session, not the backed up session, and destroys the original Servlet container session. This is shown in the figure below.



1. An attempt was made by the Web server connector to send the request to Servlet Container A, but the attempt failed because an error occurred in Servlet Container A.
2. The Web server connector distributes the requests to Servlet Container B.
3. Servlet Container B does not own the session, so it issues a request for a backed up session to Session Registry Server.
4. Session Registry Server issues a request for the updated session to Servlet Container A, the original owner of the session.

5. Servlet Container A transfers the session to Session Registry Server.

The session on Servlet Container A is destroyed by the transfer.

At this time, `HttpSessionListener.sessionDestroyed` and `HttpSessionBindingListener.valueUnBound` are not called.

6. Session Registry Server recovers the updated session that is received by the Servlet Container B.

If there is an error in communication with Servlet Container A and the updated session cannot be obtained, Session Registry Server recovers a session that has already been backed up to Servlet Container B.

In (4), when communication with Servlet Container A (original owner of the session) cannot be established, or when processing demand on the updated session of Servlet Container A is high concurrency, Session Registry Server returns the backup it has of the session, and does not request the latest session from Servlet Container A. So even when the latest session cannot be recovered, of the latest backup of the session is always available.

If the session in Servlet Container A in (5) is in use, the session is not backed up until the session is free. This period depends on the application processing, but a maximum wait time can be specified.

If the session is in use when the wait time is exceeded, the session is not recovered in Servlet Container B.

At this time, a new session is returned to the application for `getSession()` and `getSession(true)`, and null is returned for `getSession(false)`.

HttpSessionActivationListener

Sessions containing united objects can receive notifications from container events such as session activation and activation. Containers that contain sessions and which are used to move sessions between VMs receive requests to notify all the attributes that implement `HttpSessionActivationListener`.

| Event | Method | |
|----------------------------------|---|--|
| | <code>sessionWillPassivate</code> | <code>sessionDidActivate</code> |
| Recovery | Executed using the original Servlet container | Executed using the Servlet container in the recovery destination |
| The Servlet container is stopped | Executed before the session is backed up | - |
| The Servlet container is started | - | Executed after the session is recovered |

If the Session Cannot be Recovered

In the cases shown below, session recovery cannot be used because the session does not exist in Session Registry Server.

- [Maximum number of sessions that can be contained in Session Registry Server] is exceeded. Backup is performed, and the (old) session destroyed by the backup is recovered.
- The Servlet container crashes before the backup when the session backup mode is [Fixed Interval].
- Session Registry Server is restarted when IJServer and serialization of the session is disabled.
- Session Registry Server crashes and the Servlet container is restarted before serialization is performed at the specified interval, even if serialization of the session is enabled.

For details about the guarantee range for recovery, refer to "[8.2 Session Recovery Scope](#)".

8.1.3 Enabling and Disabling Session Recovery in the URI

The extension for the request URI from the Web browser can be specified to disable session recovery for the static contents.

The extension for disabling session recovery is set using the following Interstage Management Console options in Session Registry Client:

- [WorkUnit] > "WorkUnit Name" > [Environment Settings] > [Session Recovery Settings] > [End of the non-session URL]

For details about Session Registry Client settings, refer to "[8.4 Session Registry Client Settings](#)".

The `isj2eadmin` command can be used to configure these settings.

For details about the `isj2eadmin` command, refer to "`isj2eadmin`" in the Reference Manual (Command Edition).

Scenarios for using this function are described below.

When there are requests from the Web browser for dynamic contents (such as Servlet and JSP) or static contents (such as html and image files embedded in the html) that use the session, the static contents are also handled as part of the Web application. For this reason, when the session is created, the lastAccessedTime of the session (the time at which the client last sent a request associated with the session) is updated (if Cookies are set to be used),

For this reason, session backup occurs even if there is a request for static contents. If the server crashes, session recovery is also performed for the static contents.

If updating the lastAccessedTime according to access of the static contents is not particularly important for the application, however, the extension can be specified to disable session backup and recovery to prevent the performance being affected and unnecessary recovery processing.



- If a resource that executes this definition is accessed, backup in Session Registry Server is not performed.

Accordingly, the Session Registry Server session can only be timed out and deleted from the backup when a resource that executes this definition is accessed.

- When form base authentication is used, the authentication information is stored in the Servlet session.

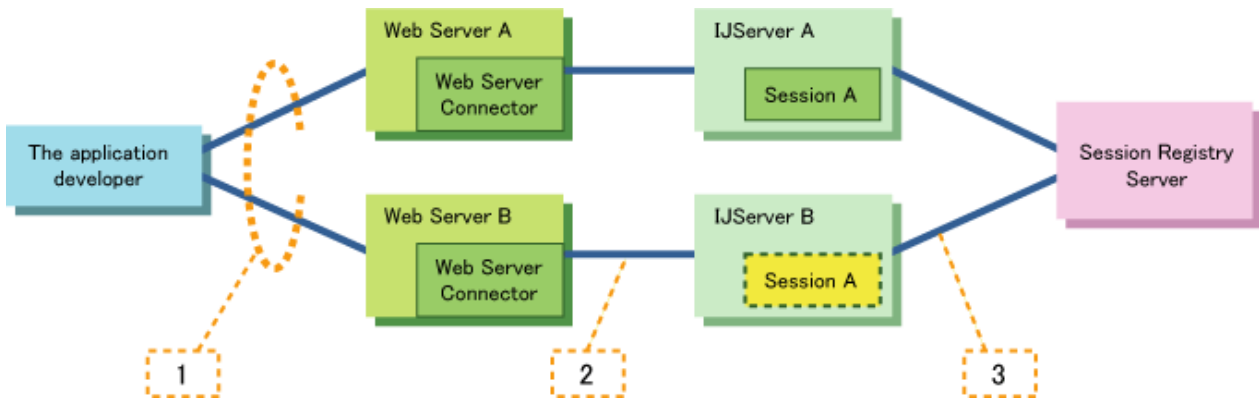
For this reason, the session is considered to be new, and re-authentication may be necessary if all of the following are true:

- Session recovery is enabled
- The extension is specified for [End of the non-session URL] of [Session Recovery Settings] in Session Registry Client (Servlet container)
- A request was made for the specified extension contents from the client
- The request was distributed to a different Servlet container due to the following reasons:
 - The IJServer restart operation was performed
 - IJServer stop or abnormal exit
- The request that performed the IJServer restart or stop was distributed to a different Servlet container because of an abnormal exit, and
- The corresponding contents are an authentication target.

Notes about Load Balancing in Web Application Independent Headers (EmbeddedStrings for Independent URLs)

If load balancing is performed for Web application independent headers (or embedded strings for independent URLs), as shown in the configuration in the figures below, these independent values are not added to the request for the static contents. The load balancing mechanism may distribute the requests to an IJServer other than the IJServer used to create the Servlet session. In this case, the session may be recovered on another IJServer, regardless of whether the IJServer is running normally (when session management is using Cookies).

Frequent occurrence of session recovery may result in a decreased performance in response.



1. If a request is distributed using application independent headers, requests for static content may be distributed to Web Server B, even if the request is for Session A.
2. The Session A request is not an IJServer B request, so there is no distribution destination for the Web server connector. In this case, it is distributed to any IJServer (in this case, IJServer B).
(Depending on the system configuration, the internal definition may be the same, and distributed to IJServer B.)
3. Session A does not exist on IJServer B. Session A is obtained from Session Registry Server. Session A of Servlet Container A is destroyed.

In the above type of load balancing, session recovery must be disabled for the corresponding contents.

8.1.4 Monitoring Session Recovery

If session recovery is used, it monitors whether the IJServer and Session Registry Server can be used on a reciprocal basis.

If an error is detected, the IJServer and Session Registry Server are marked as unusable, and are removed as a processing target until they can be used.

The IJServer sends a monitoring message to the Session Registry Server port at a fixed interval (every [10] seconds), and checks that a response is returned normally.

If a response is not returned inside the response wait time ([20] seconds), Session Registry Server sessions marked as unusable are removed as backup targets. Then, if a response is returned normally from a Session Registry Server marked as unusable, that Session Registry Server can be used and the session backup is performed.

Session Registry Server checks that a monitoring message is sent from the IJServer at a fixed interval (every [30] seconds).

If a monitoring message is not sent from the IJServer, the IJServer is marked as unusable, and removed as a processing target at the time of session recovery.

8.1.5 Web Server Connector Fault Monitoring

The Web server connector detects Servlet container problems when requests from the client are distributed to the Servlet container. (This takes anywhere from a few seconds to a few minutes)

The Web server connector cannot detect whether the Servlet container has been recovered. For this reason, requests are re-distributed to the Servlet container [1] minute after the problem is detected. At this time, the re-connection fails if the Servlet container has not been recovered.

Web server connector fault monitoring can be used to avoid this problem.

If the IJServer and Web server are each distributed on separate servers for the load balancing application, Web server connector fault monitoring can be used to monitor the operation status of the distribution destination IJServer, automatically exclude as a distribution target the IJServer on which the fault occurred, and automatically return the IJServer recovered from fault status to distribution target status.

If session recovery is used, it is recommended that it is used together with Web server connector fault monitoring.

For details about Web server connector fault monitoring, refer to "Monitoring Web Server Connector Faults" in the "Operating J2EE Applications" chapter.

8.1.6 Maximum Number of Sessions Maintained in the Session Registry Server

To avoid a system crash due to insufficient resources (such as memory) it is possible to restrict the number of sessions maintained in Session Registry Server.

The number of sessions maintained in Session Registry Server is specified in the "backup.limit" definition in the Session Registry Server environment definition file. For details, refer to "[Session Registry Server Environment Definition File Settings](#)".

If the specified number of sessions maintained in Session Registry Server is exceeded, the backup of the session not used for the longest period of time is destroyed, and sessions for which there are backup requests are maintained in Session Registry Server.

Sessions for which the backup is destroyed have no backup in Session Registry Server until there is a backup request. For this reason, if there is a recovery request for the session (when the Servlet container crashes, for example), the session is not recovered.

If the backup is destroyed, it does not mean that the session itself is destroyed. As long as no errors occur in session recovery, the session can continue. At this time, when this session is used it is backed up again and the session not used for the longest period is destroyed.

8.1.7 Session Serialization

Session information backed up in the Session Registry Server can be serialized. To enable/disable serialization, use the "session.store" definition in the Session Registry Server environment definition file. For details, refer to "[Session Registry Server Environment Definition File Settings](#)".

If session serialization is enabled, sessions backed up in Session Registry Server are written out in the serialized file.

If Session Registry Server is restarted, the session is read from the serialized file and recovered.

If session serialization is enabled, sessions are also written out in the serialized file when Session Registry Server is not running.

Session serialization timing

The session serialization timing can be specified.

The session is serialized at the interval specified in Session Registry Server.

Serialized file storage directory

The directory for storing the serialized file of the session can be specified in any Session Registry Server directory.

If the relative path is specified, the directory is relative to the directory used to deploy Session Registry Server.

The "session recovery" directory is created under the specified directory and the serialized file is output in this directory. If the corresponding directory already exists, that directory is used.

Reading the serialized file

If session serialization is enabled, Session Registry Server reads these serialized files once there is communication from the IJServer after it starts up.



- To use Session Registry Server in the Cluster Service, enable session serialization and specify the shared disk path that can be used from each node session as the serialized file storage directory.
- If Session Registry Server is restarted before session serialization is performed, the session is recovered from the serialized file for the session that was backed up last.
- The time required for serialization depends on the following factors:
 - Size of the objects stored in the session attributes
 - The number of created/updated sessions
 - The file system performance
- Before using serialization, in the following cases, delete the serialized file using the serialized session clear command:
 - The application operation is closed (the application is undeployed, or the WorkUnit is deleted)

- The serialized file storage directory is changed
- Session Registry Server is deleted

However, if the serialized file storage directory is under the directory used to deploy the Session Registry Server (including the default), it is deleted automatically.

- Interstage Application Server is uninstalled
- **Windows32/64**

The network drive or the UNC path cannot be specified for the directory where the serialized files are stored.

8.1.8 Clearing All Sessions

To clear all of the sessions on the business system to free up resources for business initialization and memory, stop and then restart all IJServers and Session Registry Servers.

Backup and recovery of backed up sessions only occurs if the session recovery functions are stopped and restarted. For this reason, not all the sessions can be cleared.

To enable session serialization, execute the serialized session clear command before the restart.

For details about the command, refer to "isj2eeadmin" in the Reference Manual (Command Edition).

8.1.9 Session Recovery Log

The session recovery output log is output in the following logs:

- The log output in the IJServer container log
- The Session Registry Server log

The output directory for each log is set using the following Interstage Management Console options:

- [WorkUnit] > "WorkUnit Name" > [Environment Settings] > [WorkUnit Settings] > [Log File Directory]

Rollover of the log file and the generations for storing the log are set using the following Interstage Management Console options:

- [WorkUnit] > "WorkUnit Name" > [Log Settings]

The isj2eeadmin command can be used to configure these settings.

For details about the isj2eeadmin command, refer to "isj2eeadmin" in the Reference Manual (Command Edition).

Logs Output in the IJServer Container Log

The logs output in the IJServer container log are output as error messages, WARNING messages, information, and access logs.

Configure the settings to select whether the access log is output or not.

Output of the access log is set using the following Interstage Management Console options in the Session Registry Client.

- [WorkUnit] > "WorkUnit Name" > [Environment Settings] > [Session Recovery Settings] > [Outputs the access log]

For details about Session Registry Client settings, refer to "[8.4 Session Registry Client Settings](#)".

Session Registry Server Logs

Session Registry Server logs are output as error messages, warning messages, information, and access logs.

Configure the settings to select whether the access log is output or not.

The access log of Session Registry Server is specified in the "access_log" definition in the Session Registry Server environment definition file. For details, refer to "[Session Registry Server Environment Definition File Settings](#)".

8.1.10 Destroying Expired (timed out) Sessions Maintained in the Session Registry Server

Normally, Session Registry Server destroys the maintained session when notification is received from the IJServer that the session was disabled.

If the IJServer crashes, however, this notification is not received. Unnecessary sessions remain in Session Registry Server and the Session Registry Server resources are compressed.

To resolve this, Session Registry Server monitors the maintained sessions at fixed intervals so that expired (timed out) sessions can be destroyed to prevent resource compression. The default monitoring interval is [60] seconds.

The monitoring interval is specified in the "clean.interval" definition in the Session Registry Server environment definition file. For details, refer to "[Session Registry Server Environment Definition File Settings](#)".

In this monitoring functionality, expired sessions are merely deleted. For this reason, if the IJServer crashes and the session exists only on the Session Registry Server, the following session destruction-related processing types will not be executed:

- javax.servlet.http.HttpSessionListener#sessionDestroyed(javax.servlet.http.HttpSessionEvent)
- javax.servlet.http.HttpSessionAttributeListener#attributeRemoved(javax.servlet.http.HttpSessionBindingEvent)
- javax.servlet.http.HttpSessionBindingListener#valueUnbound(javax.servlet.http.HttpSessionBindingEvent)

If it is necessary to execute the processing types above, by specifying the parameters shown below in the Session Registry Server environment definition file - the expired sessions that exist only on the Session Registry Server will be sent to the IJServer that is still alive, and it will be possible to execute session destruction-related processing on the IJServer:

- invalid.session.send
- invalid.session.waiting.time

For details, refer to "[Session Registry Server Environment Definition File Settings](#)".

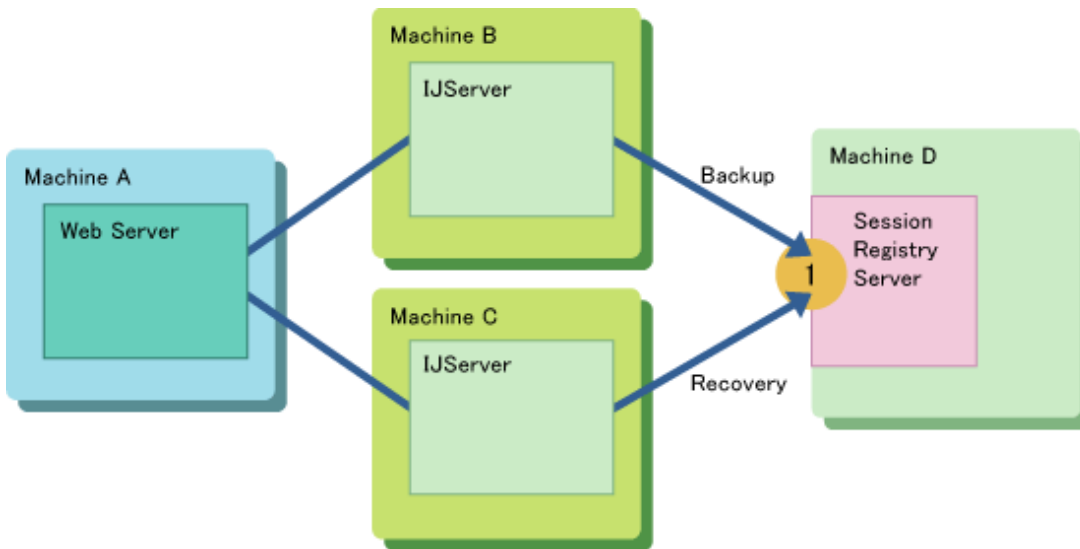
8.1.11 Session IDs

The session ID obtained from the ServletAPI shown below is different for the Cookie value added for session management between the Servlet container and Web browser, and the value set in the URL for URL encoding. To reference the session ID in the Web application, use the session ID obtained from the following ServletAPI. Do not use the value encoded in the Cookie or URL.

- javax.servlet.http.HttpSession#getId()
- javax.servlet.http.HttpServletRequest#getRequestId()

8.1.12 Session Registry Server Access Restrictions

The IP address (the Server D IP address in the figure below) and port number used to receive communication for backup and recovery requests from the Servlet container are specified in the Session Registry Server. At this time, the IP address of the partner that allows communication (the Servlet container, Server B and Server C IP address in the figure below) can be specified to restrict IP address access so that communication from an IP address other than the one specified is not received.



The IP address for communication is set using the following Interstage Management Console options in Session Registry Server.

- [WorkUnit] > "WorkUnit Name" > [Environment Settings] > [Web Server Connector Settings] > [Web Server IP Address]

For details about Session Registry Server Settings, refer to "8.3 Session Registry Server Settings".

The isj2eadmin command can be used to configure these settings.

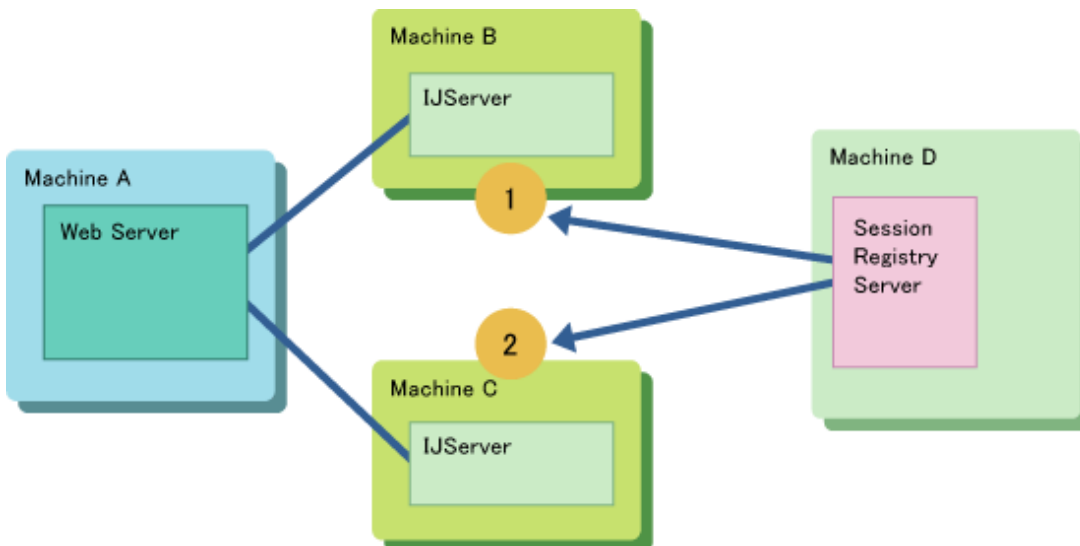
For details about the isj2eadmin command, refer to "isj2eadmin" in the Reference Manual (Command Edition).

8.1.13 Specifying the Servlet Container Control Port

If session recovery is used, the control port (1 and 2 in the figure below) must be set in the Servlet container.

The control port is used to process requests from the Session Registry Server.

If the control port is specified, the IP address of the partner that allows communication (for this function, this is Session Registry Server, Server D IP address in the figure below) can be specified to restrict IP address access so that communication from an IP address other than the one specified is not received.



The control port and the IP address for access are set using the following Interstage Management Console options in Session Registry Client.

- [WorkUnit] > "WorkUnit Name" > [Environment Settings] > [Servlet Container Settings] > [Control Port]
- [WorkUnit] > "WorkUnit Name" > [Environment Settings] > [Servlet Container Settings] > [Access permission IP address]

For details about Session Registry Client settings, refer to "[8.4 Session Registry Client Settings](#)".

The `isj2eeadmin` command can be used to configure these settings.

For details about the `isj2eeadmin` command, refer to "`isj2eeadmin`" in the Reference Manual (Command Edition).

8.2 Session Recovery Scope

The session information guarantee provides the following recovery when problems occur:

Session Recovery Guarantee Range when there is an Abnormal Exit in the Servlet Container

If a session in Session Registry Server is backed up, the session can be recovered in another running Servlet container even if there is an abnormal exit in the Servlet container.

Accordingly, an important point for the recovery guarantee range is whether the session in Session Registry Server is backed up.

The backup mode is set using the following Interstage Management Console options in Session Registry Client.

- [WorkUnit] > "WorkUnit Name" > [Environment Settings] > [Session Recovery Settings] > [Backup Mode]

For details about Session Registry Client settings, refer to "[8.4 Session Registry Client Settings](#)".

An example is shown below:

- [Request End Time] is specified

Backup is performed each time a response is returned to the client.

Using this setting, recovery is possible immediately after the response is returned to the client even when an error occurs.

- [Fixed Interval] is specified, and [5] is specified for interval value

The session information is backed up every [5] seconds. If there is an abnormal exit in the IJServer, the session information that was backed up last can be recovered.

The `isj2eeadmin` command can be used to configure these settings.

For details about the `isj2eeadmin` command, refer to "`isj2eeadmin`" in the Reference Manual (Command Edition).

Session Recovery Guarantee Range when there is an Abnormal Exit in Session Registry Server

If data stored in the session is serialized in a file, the serialized file can be recovered from the session when Session Registry Server is restarted even if there is an abnormal exit in the Servlet container.

Accordingly, an important point for the recovery guarantee range is whether the session in Session Registry Server is serialized.

The serialization interval is specified in the "`serialize.interval`" definition in the Session Registry Server environment definition file. For details, refer to "[Session Registry Server Environment Definition File Settings](#)".



Example

An example is shown below:

- [60] is set for `serialize.interval`

Session information is written out (serialized) in the file every [60] seconds.

The session cannot be guaranteed for [60] seconds following an abnormal exit. This depends on the serialization timing.

However, sessions backed up in Session Registry Server can be guaranteed.

Serialization

If serialization is used, it causes an improvement in reliability but deterioration in response times.

To enable/disable the serialization, use the "`session.store`" definition in the Session Registry Server environment definition file. For details, refer to "[Session Registry Server Environment Definition File Settings](#)".

Differences in operating behavior for each of the settings are described below:

- "on" is set for session.store

Session Registry Server serializes the session information regularly.

If Session Registry Server crashes because of unexpected errors, the session status can be recovered at the point before the crash when information was last serialized. Session Registry Server serializes the session information according to the interval specified for "serialize.interval" in the "serialize.file.path" directory in the Session Registry Server environment definition file.

- "off" is set for session.store

Session Registry Server does not serialize the session information.

The session status cannot be recovered if Session Registry Server crashes.

8.3 Session Registry Server Settings

To use session recovery, configure the following:

- Session Registry Server Settings
- Session Registry Client Settings

Session Registry Server Settings are described in the following sections:

For details, refer to "[8.4 Session Registry Client Settings](#)".

Overview

Session Registry Server is required to use session recovery.

Specify Session Registry Server in the IJServer used to back up the session.



- If the multiserver function is used, a server in the server group cannot be used to run Session Registry Server. Use an Independent Server or Stand-alone Server outside the Site.
- If the IJServer and Session Registry Server for running Web applications run on separate servers, "No" must be set for [Run Web server and WorkUnit on the same machine?] in the environment used to run Session Registry Server.

Settings Method

Create Session Registry Server.

Session Registry Server runs on the IJServer. Create the Session Registry Server IJServer and then deploy Session Registry Server.



The IJServer used to run user applications and the IJServer used to run Session Registry Server must be created separately.

To create Session Registry Server, perform the following steps:

1. Configure the system environment settings
2. Create the Registry Server WorkUnits
3. Deploy Session Registry Server
4. Configure Session Registry Server Environment Definition File Settings

1. System Environment Settings

To run Session Registry Server on a separate machine to the IJServer used to run Web applications, the following system environment settings must be set in the Interstage Management Console:

- [System] > [Environment Settings] > [Servlet Service Settings] > [Run Web server and WorkUnit on the same machine?] > [No]

2. Creating the Session Registry Server WorkUnit

Create the Session Registry Server WorkUnit using either of the following methods:

- Using the Interstage Management Console

For details, refer to "[8.3.1 Creating Session Registry Server WorkUnits \(Using the Interstage Management Console\)](#)".

- Using the `isj2eeadmin` command

For details, refer to "[8.3.2 Creating Session Registry Server WorkUnits \(Using the isj2eeadmin Command\)](#)".

3. Deploying Session Registry Server

Deploy Session Registry Server using either of the following methods:

- Using the Interstage Management Console

For details, refer to "[8.3.3 Deploying Session Registry Server \(Using the Interstage Management Console\)](#)".

- Using the `ijsdeployment` command

For details, refer to "[8.3.4 Deploying Session Registry Server \(Using the ijsdeployment Command\)](#)".



Note

Do not deploy other Web applications in the Session Registry Server IJServer.

4. Session Registry Server Environment Definition File Settings

Configure the Session Registry Server Environment Definition File Settings.

The Session Registry Server environment definition file is stored in the following directory:

Windows32/64

```
[IJServer directory]/apps/srs.ear/srs.war/WEB-INF/web.xml
```

Solaris32/64 **Linux32/64**

```
[IJServer directory]/apps/srs.ear/srs.war/WEB-INF/web.xml
```

For details about the settings, refer to "[8.3.5 Session Registry Server Environment Definition File Settings](#)".

8.3.1 Creating Session Registry Server WorkUnits (Using the Interstage Management Console)

This section describes the method to create Session Registry Server WorkUnits using the Interstage Management Console.

1. Open the [WorkUnit] > [New] windows in the Interstage Management Console.

Options not described below do not need to be set.

2. Set the following [General Settings] options.

| Option | Settings Contents |
|---------------|---|
| WorkUnit Name | Specify the name for identifying Session Registry Server. |
| WorkUnit Type | Select "IJServer". |

3. Set the following [Detailed Settings] > [IJSERVER Settings] options.

| Option | Settings Contents |
|---|---------------------------------|
| IJSERVER Type | Select "Web Applications Only". |
| Creating IJSERVER in the V8.0 compatible mode | Select "No" |

4. Set the following [Detailed Settings] > [WorkUnit Settings] options.

| Option | Settings Contents |
|--|---|
| JavaVM Option | Specify the Session Registry Server Heap size. For details about the values to set, refer to "Memory Requirements" in the Tuning Guide. Default:-Xms16m -Xmx256m |
| Auto Start | Specify whether Session Registry Server starts up at the same time as Interstage. Solaris32/64 Linux32/64 If "On" is selected, specify the start user. Session Registry Server is started up using the specified user authority. For details about changing the start user, refer to " 8.6.2 Changing the Session Registry Server Start User ". |
| Maximum application processing time | Specify the monitoring time for the Session Registry Server maximum processing time. For details, refer to " 8.5.1 Settings for each Timeout ". |
| Forcefully end application on timeout? | Specify the operation behavior when processing is not complete even though the Session Registry Server maximum processing time is exceeded. |
| WorkUnit maximum startup time | Specify the monitoring time until the Session Registry Server startup is complete. The default value should be sufficient. |
| Retry Count | Specify the number of restarts when there is an abnormal exit in Session Registry Server. [1] or less is recommended. Note: For details, refer to " 8.6.5 Restarting Session Registry Server ". |
| Retry Count Reset Interval | Specify the reset interval for abnormal exits in Session Registry Server. Note: For details, refer to " 8.6.5 Restarting Session Registry Server ". |
| Log File Directory | Specify the Session Registry Server log output directory. |

5. Set the following [Detailed Settings] > [Web Server Connector Settings] options.

| Option | Settings Contents |
|-----------------------|--|
| Web Server IP Address | This is the IP address used for access. The Servlet container IP address for Session Registry Server is specified using the "xxx.xxx.xxx.xxx" format. Separate multiple addresses with line feeds. For details, refer to " 8.1.12 Session Registry Server Access Restrictions ". If Session Registry Server and the IJSERVER are set up on the same server, the option ([Run the Web server and IJSERVER on the same server] is set) is not displayed. |

6. Set the following [Detailed Settings] > [Servlet Container Settings] options.

| Option | Settings Contents |
|------------------------------|---|
| Servlet Container IP Address | The Session Registry Server IP address is specified using the "xxx.xxx.xxx.xxx" format. If Session Registry Server and the IJSERVER are set up on the same server the option ([Run the Web server and IJSERVER on the same server] is set) is not displayed. |
| Timeout | Specify the time at which communication is cut when the connection between Session Registry Server and the Servlet container is broken. |

| Option | Settings Contents |
|--|---|
| | For details, refer to "8.5.1 Settings for each Timeout". |
| Port Number | Specify the port number used by Session Registry Server for connection to the Servlet container. |
| Maximum number of requests from client per maximum container value | Specify the maximum number of simultaneous Session Registry Server processes. For details, refer to "8.5.2 Concurrency (Number of Simultaneous Processes) Settings". |

7. Click [Create].

8.3.2 Creating Session Registry Server WorkUnits (Using the isj2eadmin Command)

This section describes the method to create Session Registry Server WorkUnits using the isj2eadmin command.

For details about the isj2eadmin command, refer to "isj2eadmin" in the Reference Manual (Command Edition).

1. Copy the IJServer definition file samples to any directory.

Samples storage directory:

Windows32/64

```
C:\Interstage\F3FMjssrs\sample\srs
```

Solaris32/64 **Linux32/64**

```
/opt/FJSVjssrs/sample/srs/
```

| File | Character Code |
|----------------------|----------------|
| sample_utf-8.xml | UTF-8 |
| sample_euc-jp.xml | EUC-JP |
| sample_shift_jis.xml | Shift_jis |

Note

The IJServer definition file samples all have the same IJServer definition contents, but the files are separated according to the character code that is entered. Edit the file so that the character code is appropriate.

Example

Copy the Shift_jis samples file as "srs.xml".

Windows32/64

```
copy C:\Interstage\F3FMjssrs\sample\srs\sample_shift_jis.xml srs.xml
```

Solaris32/64 **Linux32/64**

```
cp /opt/FJSVjssrs/sample/srs/sample_shift_jis.xml srs.xml
```

2. Edit the copied IJServer definition file according to the Session Registry Server operating environment. For details, refer to "IJServer Definition File".
3. Use the isj2eadmin command to create the IJServer.



Example

```
Isj2eeadmin ijserver -a -f srs.xml
```

IJServer Definition File

This section describes the IJServer definition file.

Description Format

The IJServer definition file is described in the following format.

Note

Parts shown in bold must be configured according to the environment.

```
<?xml version=" 1.0" encoding="Shift_JIS" standalone="yes" ?>
<Isj2eeIjServer Definition>
  <IJServer>
    <Name>[ijserver name]</Name>
    <Type>WEB</Type>
    <AutomaticStart>
      <Mode>YES</Mode>
      <User>root</User>
    </AutomaticStart>
    <ApplicationRetry>
      <AbnormalTerminationCounts>1</AbnormalTerminationCounts>
      <RetryCountResetTime>600</Retry CountResetTime>
    </ApplicationRetry>
    <CurrentDirectory>
      <NumberOfRevisionDirectories>1</NumberOfRevisionDirectories>
    </CurrentDirectory>
    <Common>
      <JavaCommandOptions>-Xms 16m -Xmx256m</JavaCommandOptions>
      <ProcessingTime>
        <MaximumProcessingTime>400</MaximumProcessingTime>
        <TerminateProcessModeForTimeout>NO</TerminateProcessModeForTimeout>
      </ProcessingTime>
    </Common>
    <Web>
      <IPAddress/>
      <Timeout>60</Timeout>
      <Ports>
        <Number>[port number]</Number>
      </Ports>
      <ThreadConcurrency>
        <MaxThreads>64</MaxThreads>
      </ThreadConcurrency>
      <Www>
        <AcceptedHosts>
          <Address/>
        </AcceptedHosts>
      </Www>
    </Web>
    <Log>
      <Directory/>
      <Mode>SIZE</Mode>
      <Size>1</Size>
      <StartTime/>
      <Interval/>
      <HistorySize>1</HistorySize>
    </Log>
```



```
</IJServer>  
</Isj2eeIjServer Definition>
```

Definition Contents

For details about each tag, refer to "isj2eeadmin" in the Reference Manual (Command Edition).

Name: WorkUnit settings

Specify the name for identifying Session Registry Server.

This must be configured according to the environment.

Type: WorkUnit settings

Do not change "WEB".

Mode: WorkUnit settings

Specify whether Session Registry Server starts up at the same time as Interstage.

This can be changed if necessary.

User: WorkUnit settings

Solaris32/64 Linux32/64

Specify the start user if YES is specified in the <Mode> tag. Session Registry Server is started up using the specified user authority.

For details about changing the start user, refer to "[8.6.2 Changing the Session Registry Server Start User](#)".

This can be changed if necessary.

AbnormalTerminationCounts: WorkUnit settings

Specify the number of restarts when there is an abnormal exit in Session Registry Server. [1] or less is recommended.

For details, refer to "[8.6.5 Restarting Session Registry Server](#)".

This can be changed if necessary.

RetryCountResetTime: WorkUnit settings

Specify the reset interval for abnormal exits in Session Registry Server.

For details, refer to "[8.6.5 Restarting Session Registry Server](#)".

This can be changed if necessary.

NumberOfRevisionDirectories: WorkUnit settings

Specify the generations for backing up the current directory.

This can be changed if necessary.

JavaCommandOptions: WorkUnit settings

Specify the Session Registry Server Heap size.

This can be changed if necessary.

MaximumProcessingTime: WorkUnit settings

Specify the monitoring time for the Session Registry Server maximum processing time.

For details, refer to "[8.5.1 Settings for each Timeout](#)".

This can be changed if necessary.

TerminateProcessModeForTimeout: WorkUnit settings

Specify the operating behavior when processing is not complete even though the Session Registry Server maximum processing time is exceeded.

This can be changed if necessary.

IPAddress: Servlet container settings

The Session Registry Server IP address is specified using the "xxx.xxx.xxx.xxx" format.

This option is only enabled when the Web server and WorkUnit are not run in the same server.

This can be changed if necessary.

Timeout: Servlet container settings

Specify the time at which communication is cut when the connection between Session Registry Server and the Servlet container is broken.

For details, refer to "8.5.1 Settings for each Timeout".

This can be changed if necessary.

Number: Servlet container settings

Specify the port number used by Session Registry Server for connection to the Servlet container.

This must be configured according to the environment.

MaxThreads: Servlet container settings

Specify the maximum number of simultaneous Session Registry Server processes.

For details, refer to "8.5.2 Concurrency (Number of Simultaneous Processes) Settings".

This can be changed if necessary.

Address: Web Server Connector settings

This is the IP address used for access. The Servlet container IP address for Session Registry Server is specified using the "xxx.xxx.xxx.xxx" format. If specifying multiple addresses, specify one for each tag.

For details, refer to "8.1.12 Session Registry Server Access Restrictions".

If Session Registry Server and the IJServer are set up on the same server the option ([Run the Web server and IJServer on the same server] is set) is not displayed.

This can be changed if necessary.

Directory: Log settings

Specify the Session Registry Server log output directory.

This can be changed if necessary.

Mode: Log settings

Specify Log size or Time interval for the rollover.

This can be changed if necessary.

Size: Log settings

Specify the rollover Log size.

This can be changed if necessary.

StartTime: Log settings

Specify the rollover start time.

This can be changed if necessary.

Interval: Log settings

Specify the rollover interval.

This can be changed if necessary.

HistorySize: Log settings

Specify the generations for storing the rolled over log file.

This can be changed if necessary.

Note

- Tags not mentioned in Definition Contents must not be defined in the Session Registry Server IJServer definition file.
- When the IJServer of the Session Registry Client is in V8.0 compatible mode, the Session Registry Server must also be created in V8.0 compatible mode. To create the Session Registry Server in this mode, the Version tag must be used.

8.3.3 Deploying Session Registry Server (Using the Interstage Management Console)

This section describes the method to deploy Session Registry Server using the Interstage Management Console.

1. Open the [WorkUnit] > "Session Registry Server Name" > [Deploy] windows in the Interstage Management Console.
2. Configure the settings shown in the table below. Do not set any other options.

| [Deploy] window options | Settings Contents |
|---|--|
| [Deployment File] | Select "Deploy a file stored on the server", and then specify the following file: Windows32/64 C:\Interstage\F3FMjssrs\ear\srs.ear Solaris32/64 Linux32/64 /opt/FJSVjssrs/ear/srs.ear |
| [WorkUnit Restart] | "Restart WorkUnit after deployment" is not selected. |
| [Detailed Settings] > [Web Application Settings] > [Web Application Name] | Do not change the following: - ROOT |

3. Execute [Deploy].

8.3.4 Deploying Session Registry Server (Using the ijsdeployment Command)

This section describes the method to deploy Session Registry Server using the *ijsdeployment* command.

For details about the *ijsdeployment* command, refer to "*ijsdeployment*" in the Reference Manual (Command Edition).

Use the *ijsdeployment* command to deploy Session Registry Server.

Example

Session Registry Server Name (WorkUnit Name) is "SRSV01"

Windows32/64

```
ijsdeployment -n SRSV01 -f C:\Interstage\F3FMjssrs\ear\srs.ear
```

Solaris32/64 **Linux32/64**

```
ijsdeployment -n SRSV01 -f /opt/FJSVjssrs/ear/srs.ear
```

8.3.5 Session Registry Server Environment Definition File Settings

This section describes the Session Registry Server environment definition file settings.

Description Format

The Session Registry Server environment definition file is described in the following format:

```
<web-app>
  <context-param>
    <param-name>backup.limit</param-name>
    <param-value>0</param-value>
  </context-param>
  <context-param>
    <param-name>clean.interval</param-name>
    <param-value>60</param-value>
  </context-param>
  <context-param>
    <param-name>session.store</param-name>
    <param-value>off</param-value>
  </context-param>
  <context-param>
    <param-name>serialize.file.path</param-name>
    <param-value>serializedata</param-value>
  </context-param>
  <context-param>
    <param-name>serialize.interval</param-name>
    <param-value>60</param-value>
  </context-param>
  <context-param>
    <param-name>pop.timeout</param-name>
    <param-value>360</param-value>
  </context-param>
  <context-param>
    <param-name>access_log</param-name>
    <param-value>off</param-value>
  </context-param>
  <context-param>
    <param-name>invalid.session.send</param-name>
    <param-value>off</param-value>
  </context-param>
  <context-param>
    <param-name>invalid.session.waiting.time</param-name>
    <param-value>2000</param-value>
  </context-param>
</web-app>
```

Definition Contents

backup.limit: Restricts the number of maintained sessions

Specify the maximum number of sessions maintained in Session Registry Server. Specify a number from [0] to [2147483647]. The default is [0]. If [0] is specified, there is no maximum value.

clean.interval: Expired session monitoring interval

Specify the monitoring interval (in seconds) for timed out sessions. Specify a number from [30] to [2147483647]. The default is [60].

session.store: Session serialization

Select whether or not Session Registry Server serializes the session (stored in a file). Upper and lower case are treated as the same.

If the Cluster Service is used, "on" must be selected.

Select either of the following:

- on: Enabled
- off: Disabled (This is the default)

serialize.file.path: Directory for storing the serialized file of the session

This is essential if "on" is specified for session.store.

Specify in the following range the directory where the serialized files of the session data are to be stored.

Windows32/64

[1] to [90] characters

Solaris32/64 **Linux32/64**

[1] to [860] characters

If a relative path is specified, it is relative to the directory in which the Session Registry Server was deployed. In this case, ensure that the absolute path length does not exceed the maximum allowed.

The following applies when the initial value of this item is "serializedata" (relative path), the Interstage install directory and the IJServer directory use the default values, and the Session Registry Server name is "SRS001":

Windows32/64

```
C:\Interstage\J2EE\var\deployment\ijserver\SRS001\apps\srs.ear\srs.war\serializedata
```

To specify this item using the same conditions, up to 19 characters may be used.

Solaris32/64 **Linux32/64**

```
/opt/FJJSVj2ee/var/deployment/ijserver/SRS001/apps/srs.ear/srs.war/serializedata
```

To specify this item using the same conditions, up to 798 characters may be used.

The "session recovery" subdirectory under the directory specified in this definition is created and used.

For details about specified values, refer to the Notes following this section.

- If a directory that does not exist is specified, there is an error at startup, and application deployment is not possible.
- When the Cluster Service is used, specify the path of the shared disk that can be used from each node.

- **Windows32/64**

The network drive or the UNC path cannot be specified.

- **Solaris32/64** **Linux32/64**

If the Session Registry Server is operated by a user that does not have root authority for Session Registry Server, the user must have complete authority for the specified directory. Change the owner or authority if necessary.

serialize.interval: The interval for serializing the session

Specify the interval (in seconds) for serializing the session. Specify a number from [1] to [2147483647]. The default is [0].

pop.timeout: The wait time for obtaining the updated session

Specify the wait time (in seconds) for obtaining the updated session. Specify a number from [1] to [86400]. The default is [360].

At the time of session recovery, the updated session is obtained from the container that maintains the session. This is the maximum wait time for obtaining the updated session.

This wait time for the session is used during request processing. If the session is in use when the wait time overruns, the corresponding session is not recovered.

For details about the timeout, refer to "8.5.1 Settings for each Timeout".

access_log: Output Access Log

Select whether the access log output is enabled or disabled. Upper and lower case are treated as the same. A log of the backup, recovery, and monitoring communication details is output in the access log.

Select either of the following:

- **on:** Output
- **off:** Not output (This is the default)

invalid.session.send: Whether to send expired sessions to the IJServer

Set whether to send to the Session Registry Client that is still alive the sessions that reached the timeout specified in the Session Registry Server and have been destroyed but not notified by the IJServer. By enabling this definition, it will be possible to execute session destruction-related processing for sessions that exist only on the Session Registry Server.

- **on** : Enabled
- **off**: Disabled (This is the default)

invalid.session.waiting.time: Waiting time until issuing notification that the session is invalid

When "invalid.session.send" is "on", if sessions that reached the timeout specified in the Session Registry Server have not been notified by the Session Registry Client as being invalid, even though the waiting time (in seconds) specified in this definition was reached, the expired sessions are sent to the Session Registry Client that is still alive. Specify an integer value between 60 and 2147483647 for the waiting time (in seconds). The default value is 2000.



- To enter 2-byte code (such as comments or `serialize.file.path`) in the Session Registry Server environment definition file, use UTF-8.
- The Session Registry Server environment definition file contains tags not mentioned in Definition Contents. These must not be updated or deleted.

8.4 Session Registry Client Settings

To use session recovery, configure the following:

- Session Registry Server Settings
- Session Registry Client Settings

Session Registry Client Settings are detailed in the following sections.

For details, refer to "[8.3 Session Registry Server Settings](#)".

Configuration

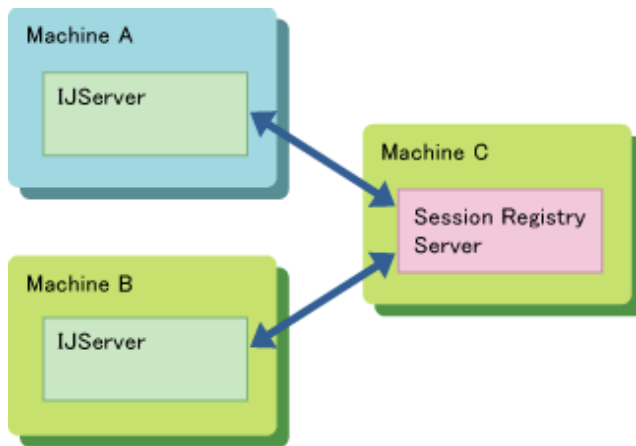
An example of the Session Registry Server and Session Registry Client configuration is shown below.

One Session Registry Server is used on more than one Server

In this configuration, the IJServer and Session Registry Server run on separate servers. The Web application on the IJServer can continue even if Server A or Server B crashes. However, Session recovery cannot be used if Server C crashes on Session Registry Server. (Session backup/recovery cannot be used, but business can continue normally.)

This configuration has the following disadvantages with regard to cost:

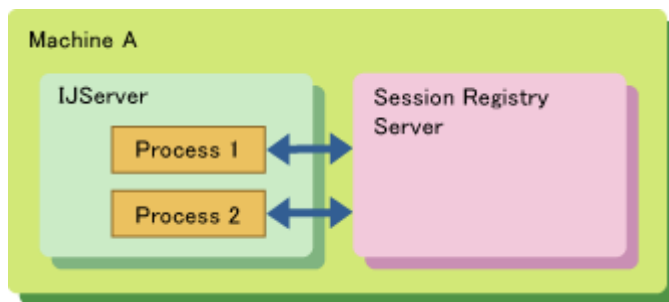
- A server must be reserved for Session Registry Server.
- The server used for Session Registry Server must be high-reliability.



Running IJServer using Process Concurrency

In this configuration, when an IJServer process crashes, the processes that did not crash are recovered in the session. The Web application can continue even if the process crashes.

In this configuration, the IJServer runs on one server. For this reason, the Web application cannot continue if the server crashes.



Settings Method

Specify the Session Registry Server used from the Servlet container in Session Registry Client Settings.

Session Registry Client (Servlet container) settings are described below.



Note

- Session Registry Client (Servlet container) settings can only be made in an environment in which the Session Registry Client package is installed.
- If the IJServer type is EJB Only, [Session Recovery Settings] and [Servlet Container Settings] are not displayed.
- **Windows32/64**
When the Session Registry Server is Windows, the Web application name is not case sensitive. To back up sessions of the applications with names that are the same but use different case, specify them for different Session Registry Servers.

To use Session Registry Client, the following settings are required in addition to the normal IJServer settings.

Configure the following settings using the Interstage Management Console:

- [WorkUnit] > [New] > [Servlet Container Settings]
- or
- [WorkUnit] > "WorkUnit Name" > [Environment Settings] > [Servlet Container Settings]
- Control Port
This is essential for using session recovery.

- Access permission IP address

If more than one IP address is set in the server used to run Session Registry Server, describe all the IP addresses.

- [WorkUnit] > [New] > [Session Recovery Settings]

or

[WorkUnit] > "WorkUnit Name" > [Environment Settings] > [Session Recovery Settings]

- Session Recovery

Select whether to use Session Registry Server in the IJServer.

- Session backup destination Session Registry Server address:Port

Specify the session backup directory Session Registry Server.

- Backup Mode

Select the mode for backing up the session.

- Response waiting time from the Repository server

Set the response wait time from Session Registry Server. For details about the values to set, refer to "[8.5.1 Settings for each Timeout](#)".

- End of the non-session URL

Add the non-session contents to the extension at the end of the URL.

- Outputs the access log

Specify whether to output an access log.

Use the isj2eeadmin command to configure these settings.

For details about the isj2eeadmin command, refer to "isj2eeadmin" in the Reference Manual (Command Edition).

8.5 Session Recovery Settings

This section describes the Session Recovery Settings.

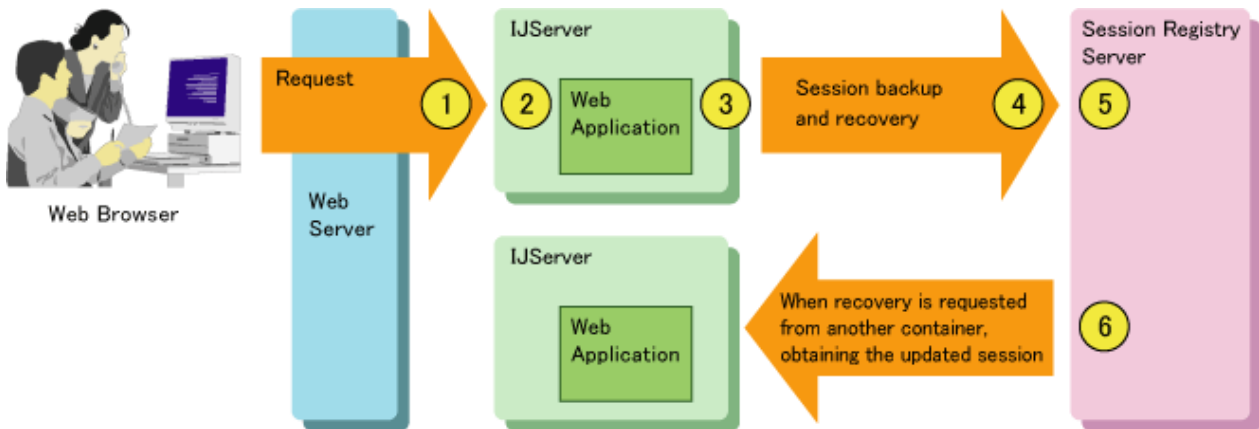
8.5.1 Settings for each Timeout

This section describes the communication timeout settings used in session recovery.

The timeouts shown in the table are set according to session recovery.

| Settings Location [Option] | | Value/Default Example in Session Registry Server | Contents |
|-------------------------------|--|---|--|
| (1) | IJServer:Servlet Container Settings [Timeout] | from [1] to [2147483] / 60 60 | Specify the time until the communication is disconnected if communication with the client is cut off. |
| (2) | IJServer:WorkUnit Settings [Application maximum processing time] | from [0] to [86400] / 480 480 | Specify the monitoring time for the maximum application processing time. |
| (3) | IJServer:Session Recovery Settings [Response waiting time from the Repository server] | from [1] to [86400] / 440 440 | Set the response wait time from Session Registry Server. |
| (4) | Session Registry Server:Servlet Container Settings | from [1] to [2147483] / 60 | Specify the time until the communication is disconnected if communication between Session Registry Server and the target Servlet container is cut off. |

| Settings Location [Option] | | Value/Default Example in Session Registry Server | Contents |
|-------------------------------|--|---|---|
| | [Timeout] | 60 | |
| (5) | Session Registry Server:WorkUnit Settings [Application maximum processing time] | from [0] to [86400] / 480 400 | Specify the monitoring time for the maximum application processing time for Session Registry Server. |
| (6) | Session Registry Server:Session Registry Server Environment Definition File [Wait time for obtaining the updated session (pop.timeout)] | from [1] to [86400] / 360 360 | When session recovery is used, the updated session is obtained from the container that maintains the original session. This is the maximum wait time for obtaining the updated session. This is the wait time for the session that is used during request processing. If the session is in use when the wait time overruns, the corresponding session is not recovered. |



Make the settings for each value so that the following relationship is satisfied.

| |
|-----------------------|
| (2) > (3) > (5) > (6) |
| (1) <= (2) |
| (4) <= (5) |

Set as follows when the IJServer or the Session Registry Server is in the V8.0 compatible mode:

| Settings [Option] | | Value/Default Example in Session Registry Server | Description |
|----------------------|--|---|---|
| (1) | IJServer:Servlet Container Settings [Timeout] | from [1] to [2147483] / 480 480 | Specify the time until the communication is disconnected if communication with the client is cut off. |
| (2) | IJServer:WorkUnit Settings [Application maximum processing time] | from [0] to [86400] / 480 480 | Specify the monitoring time for the maximum application processing time. |
| (3) | IJServer:Session Recovery Settings [Response waiting time from the Repository server] | from [1] to [86400] / 440 440 | Set the response wait time from Session Registry Server. |

| Settings [Option] | | Value/Default | Description |
|----------------------|--|------------------------------------|---|
| | | Example in Session Registry Server | |
| (4) | Session Registry Server:Servlet Container Settings [Timeout] | from [1] to [2147483] / 480 | Specify the time until the communication is disconnected if communication between the Session Registry Server and the target Servlet container is severed. |
| | | 400 | |
| (5) | Session Registry Server:WorkUnit Settings [Application maximum processing time] | from [0] to [86400] / 480 | Specify the monitoring time for the maximum application processing time for Session Registry Server. |
| | | 400 | |
| (6) | Session Registry Server:Session Registry Server Environment Definition File [Wait time for obtaining the updated session (pop.timeout)] | from [1] to [86400] / 360 | When session recovery is used, the updated session is obtained from the container that maintains the original session. This is the maximum wait time for obtaining the updated session. This is the wait time for the session that is used during request processing. If the session is in use when the wait time overruns, the corresponding session is not recovered. |
| | | 360 | |

8.5.2 Concurrency (Number of Simultaneous Processes) Settings

Specify a maximum of [2048] simultaneous Session Registry Server processes.

The total for the WorkUnit (IJServer) that serves as the backup directory for the Session Registry Server session is calculated as follows:

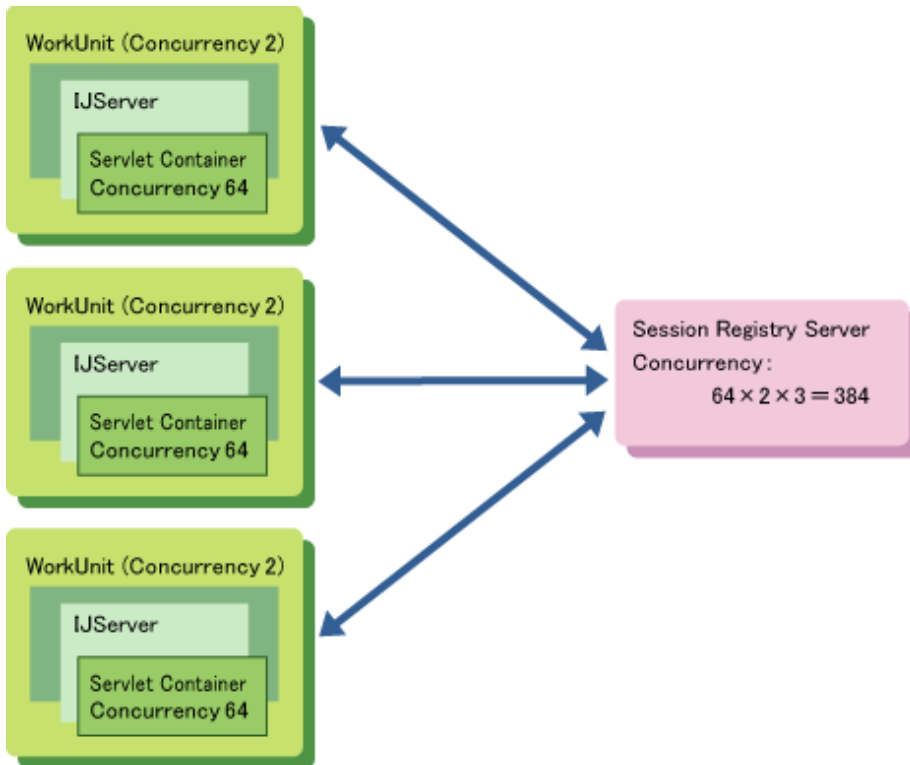
| |
|--|
| Number of simultaneous Servlet container processes x WorkUnit (IJServer) process concurrency |
|--|



Example

In the following example, Servlet container concurrency is [64], concurrency is [2], and WorkUnit is [3]. These are multiplied to calculate the total for the Session Registry Server backup directory:

| |
|--------------------------|
| [64] x [2] x [3] = [384] |
|--------------------------|



8.5.3 Example of Setting of IP Address and Port Number

An example of setting the IP address and port number for each of the following machine configurations is explained below.

- To run Session Registry Server on a separate machine to the IJServer used to run Web applications
- To run Session Registry Server on the same machine as the IJServer used to run Web applications, and to run Web server on the same machine as the IJServer
- To run Session Registry Server on the same machine as the IJServer used to run Web applications, and to run Web server on a separate machine to the IJServer



Note

To run Session Registry Server on a separate machine to the IJServer used to run Web applications, the following system environment settings must be set in the Interstage Management Console.

- [System] > [Environment Settings] > [Servlet Service Settings] > [Run Web server and WorkUnit on the same machine?] > [No]

To run Session Registry Server on a Separate Machine to the IJServer used to run Web Applications

Setting of Session Registry Server (192.0.2.111)

| | Item Name | | Setting Value Example | Notes |
|-----|-------------------------------|------------------------------|----------------------------|---|
| (1) | Servlet Container Settings | Servlet Container IP Address | 192.0.2.111 | Set the same value as the IP address and port number of (3) and (5). |
| | | Port Number | 5678 | |
| (2) | Web Server Connector Settings | Web Server IP Address | 192.0.2.222 192.0.2.223 | Set the IP address of machine A and machine B. If more than one IP address is set in the server used to run IJServer for the web application, describe all of the IP addresses. |

Setting of IJServer (MachineA: 192.0.2.222)

| | Item Name | | Setting Value Example | Notes | |
|-----|----------------------------|---|------------------------------|--|--|
| (3) | Session Recovery Settings | Session backup destination Session Registry Server address:Port | 192.0.2.111:5678 | Set the same value as the IP address and port number of (1). | |
| (4) | Servlet Container Settings | Control Port | Port Number | 15000 | - |
| | | | Access permission IP address | 192.0.2.111 | If using the access restrictions, set the same value as the IP address of (1). If more than one IP address is set in the server used to run Session Registry Server, describe all the IP addresses. |

Setting of IJServer (MachineB: 192.0.2.223)

| | Item Name | | Setting Value Example | Notes | |
|-----|----------------------------|---|------------------------------|--|---|
| (5) | Session Recovery Settings | Session backup destination Session Registry Server address:Port | 192.0.2.111:5678 | Set the same value as the IP address and port number of (1). | |
| (6) | Servlet Container Settings | Control Port | Port Number | 15000 | - |
| | | | Access permission IP address | 192.0.2.111 | If using the access restrictions, set the same value as the IP address of (1). If more than one IP address is set in the server used to run Session Registry Server, describe all of the IP addresses. |

To run Session Registry Server on the same machine as the IJServer used to run Web Applications, and to run Web server on the same Machine as the IJServer

Setting of Session Registry Server

| | Item Name | | Setting Value Example | Notes |
|-----|----------------------------|-------------|-----------------------|---|
| (1) | Servlet Container Settings | Port Number | 5678 | Set the same value as the port number of (2). |

Setting of IJServer

| | Item Name | | Setting Value Example | Notes | |
|-----|----------------------------|---|------------------------------|--|--|
| (2) | Session Recovery Settings | Session backup destination Session Registry Server address:Port | 127.0.0.1:5678 | Set "127.0.0.1" as the IP address. Set the same value as the port number of (1). | |
| (3) | Servlet Container Settings | Control Port | Port Number | 15000 15001 | When proces concurrency of IJServer (Workunit) used to run Web applications is [2] |
| | | | Access permission IP address | 127.0.0.1 | If using the access restrictions, set "127.0.0.1". |

To run Session Registry Server on the same machine as the IJServer used to run Web applications, and to run Web server on a separate machine to the IJServer

Table Setting of Session Registry Server

| | Item Name | | Setting Value Example | Notes |
|-----|-------------------------------|------------------------------|-----------------------|--|
| (1) | Servlet Container Settings | Servlet Container IP Address | 127.0.0.1 | Set "127.0.0.1"(*1) as the IP address. Set the same value as the port number of (3). |
| | | Port Number | 5678 | |
| (2) | Web Server Connector Settings | Web Server IP Address | 127.0.0.1 | Set "127.0.0.1"(*1). |

Setting of IJServer

| | Item Name | | Setting Value Example | Notes | |
|-----|----------------------------|---|------------------------------|--|---|
| (3) | Session Recovery Settings | Session backup destination Session Registry Server address:Port | 127.0.0.1:5678 | Set "127.0.0.1"(*1) as the IP address. Set the same value as the port number of (1). | |
| (4) | Servlet Container Settings | Control Port | Port Number | 15000 15001 | When process concurrency of IJServer (Workunit) used to run Web applications is [2] If using the access restrictions, set "127.0.0.1"(*1). |
| | | | Access permission IP address | 127.0.0.1 | |

(*1) Each IP address can also specify the IP addresses (192.0.2.111 etc.) of an actual machine instead of 127.0.0.1. However, both addresses cannot be specified.

8.6 Session Recovery Application Method

This section describes the method to run session recovery.

8.6.1 Session Registry Server Operations and Information Reference

Session Registry Server operations and information reference is performed by using the WorkUnit from the Interstage Management Console.

Session Registry Server Status Display and Operations

[WorkUnit] > "Session Registry Server Name" > [Operate]

Session Registry Server Environment Settings

[WorkUnit] > "Session Registry Server Name" > [Environment Settings]

[WorkUnit] > "Session Registry Server Name" > [Log Settings]

Use the isj2eadmin command to configure these settings.

For details about the isj2eadmin command, refer to "isj2eadmin" in the Reference Manual (Command Edition).

Session Registry Server List Display

[WorkUnit] > [Status]

Session Registry Server Monitor Display

[WorkUnit] > "Session Registry Server Name" > [Monitor]

Log Reference

[WorkUnit] > "Session Registry Server Name" > [View Log]

The log is output in the same file as the IJServer (Servlet container) log. For this reason, the container log/start information is referenced.

8.6.2 Changing the Session Registry Server Start User

Solaris32/64 Linux32/64

The WorkUnit used to run Session Registry Server can be started by any user. If session serialization is used, however, and the start user is changed after the first time the WorkUnit is started, inheritance of the serialized session cannot be guaranteed.

If the user that runs the application using serialization is different to the previous start user, first perform either of the following steps. Ensure that the specified directory exists in "serialize.file.path" or that the new user has complete authority:

- Clear the serialized file using the serialized file clear command.
- Change the directory specified in "serialize.file.path" of the Session Registry Server environment definition file to a different directory.

If the WorkUnit is started by a different user and the above steps are not taken, serialization may fail. If this happens, perform either of the following steps:

- Start the WorkUnit as the original user.



.....
This action is not possible if the WorkUnit is started even once with root authority. In this case, perform either of the following:
.....

- Clear the serialized file using the serialized file clear command.
- Change the directory specified in "serialize.file.path" to a directory for which the user has complete authority.

8.6.3 Isolating the Server

If the server used to run Session Registry Server is shut off for maintenance, for example, the following procedure is necessary:

1. Stop Session Registry Server.
2. Use the container log or system log to check the following in the IJServer used to perform the backup in the stopped Session Registry Server.
 - Session Registry Server is marked as unusable from "[8.1.4 Monitoring Session Recovery](#)" (message JSSR32001 is output).
3. Stop the server.

If the server is stopped without performing the above procedure, the response of Web applications that are running may deteriorate temporarily or requests from the client may be timed out.

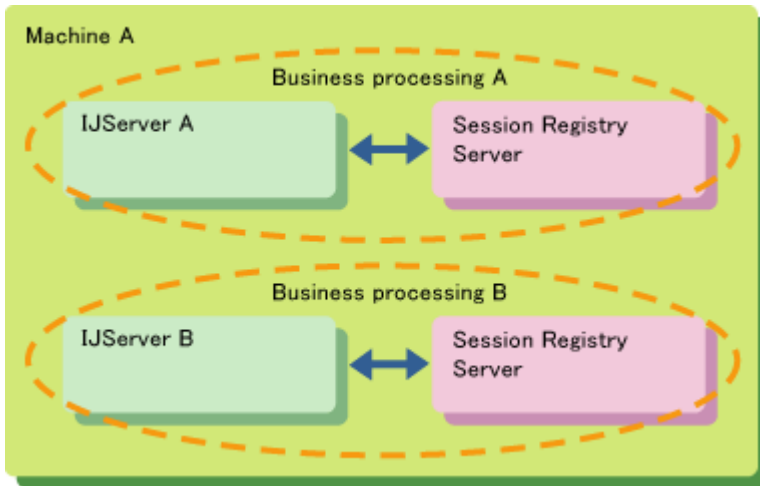
If the server is isolated, session backup/recovery cannot be used, but business can continue normally.

8.6.4 Running More than One Session Registry Server Application

More than one Session Registry Server can run in the same server.

Session Registry Server can be categorized for use in a single specific business application so that there is no effect on one business from another (such as performance, or following the occurrence of trouble).

Example of categorizing Session Registry Server for a single specific business application:



8.6.5 Restarting Session Registry Server

When there is an abnormal exit in Session Registry Server, there is a possibility that, depending on the Session Registry Server (WorkUnit) definition [Retry Count] and [Retry Count Reset Interval] settings, the problem will reoccur even if Session Registry Server is restarted. This possibility also exists if the cause is not established and fixed.

For example, if the estimate for the resources used was insufficient, the problem may reoccur even if Session Registry Server is restarted, (the estimate for the resources is exceeded and the session is backed up).

If Session Registry Server is restarted after the abnormal exit, and Session Registry Server and Servlet container communication is recovered, information consistency for sessions maintained for both is checked. This means that requests from the Web browser cannot be processed temporarily. If the restart is repeated without the cause being fixed, the response time may be affected.

For this reason, it is recommended that [1] or a low value is set for [Retry Count]. The Servlet container cannot be used to perform session backup or recovery if Session Registry Server is not running, but business processing can continue normally. After the cause of the problem is resolved, restart Session Registry Server.

The processing time required to check consistency depends on the network/server performance, the enabled number of sessions, and size.

8.6.6 Backing Up/Restoring Session Registry Server Resources

Session Registry Server is created and run as the IJServer WorkUnit.

Accordingly, the procedure for backing up/restoring IJServer resources is also used to backup/restore Session Registry Server resources.

For details about backing up/restoring the IJServer, refer to the "Maintenance (Resource Backup)" chapter in the Operator's Guide.

Note

The serialized file output when session serialization is enabled is a temporary resource on the application and is unnecessary for backup/restore operations. Depending on the directory specified to store the serialized file of the session (refer to the Note below) it may be restored together with other resources.

After the restoration, use the `clearsession` subcommand of the `jsrsadmin` command to clear the serialized session information before starting Session Registry Server.

If an initial value or relative path is specified.

For details about the `jsrsadmin` command, refer to "`jsrsadmin`" in the Reference Manual (Command Edition).

8.7 Application Creation Method

This section describes the Web application conditions for using session recovery.

- Objects stored in the session attribute must be able to implement the java.io.Serializable interface (that is, they must be able to be serialized).

If this condition is not met, IllegalArgumentException occurs when setAttribute is executed.

- The transient modifier must be declared in the instance variable maintained by the object stored in the session attribute that cannot be serialized (this modifier is not a backup or recovery target).
- If an object representing OS and process (Java VM) specific resources is stored in the session, the operating behavior cannot be guaranteed. For example, Threadclass and the input/output class (the class that inherits InputStreamclass or Writerclass), and the CORBA and EJB client (such as stub, org.omg.CORBA.ORB, and javax.naming.Context, and classes that maintain references to these kindred objects) are specific resources in the Java VM. To develop user definition objects, do not allow this type of class to be inherited or references to it to be maintained.
- All the information required to recover application requests must be stored in the session attribute.

In session recovery processing, only information stored in the session attribute is recovered. Information stored in objects and classes that are separate from the session are not recovered. If application request processing depends on non-session attribute dynamic information, the application may not be able to process requests in the recovery destination Servlet container correctly, even if session recovery processing is used.

- If the object class stored in the session attribute is changed and use of the pre-change backed up session is continued, the object class must be created or corrected according to the supported methods shown in "Method to Change Object Classes Stored in the Session Attribute".

If this condition is not met, java.io.InvalidClassException is output in the log when you attempt to recover in the post-correction class the object saved in the pre-correction class, and the application request is processed as a new session.



Note

The larger the object stored in the session attribute, the slower the execution time.

Method to Change Object Classes Stored in the Session Attribute

The methods to change object classes stored in the session attribute are as follows:

- Future Correction Method when the Object Class is Created
- Method to Change when the Object Class is Created but there is no Provision for Correction

Future Correction Method when the Object Class is Created

When the object class is created, if it is anticipated that attributes will be changed in the future and that the information stored pre-correction will also be used post-correction, the following action must be taken:

a) Creation

serialVersionUID is declared as the class variable when the object class is created and any value (the long integer) is set.

[Example of the pre-correction object class]

```
package com.fujitsu.sample;
import java.io.*;

public class Company implements Serializable {

    private String longName;
    private String shortName;
    private String address;
    static final long serialVersionUID = 10203040506070809L; .....( 1)

    public Company(String longName,
                   String shortName,
                   String address) {

        :
    }
}
```



```

        :
    }
}

( 1): Any (unique) long integer is set.

```

b) Correction

The object class is changed (the attribute is added).

[Example of the post-correction object class]

```

package com.fujitsu.sample;
import java.io.*;

public class Company implements Serializable {

    private String longName;
    private String shortName;
    private String address;
    private String telNumber; .....(2)
    static final long serialVersionUID = 10203040506070809L;

    public Company(String longName,
                    String shortName,
                    String address,
                    String telNumber ) {

        :
        :
    }
}

(2): The telNumber attribute is added.

```

Method to Change when the Object Class is Created but there is no Provision for Correction

If you want to use pre-correction information as post-correction, but there is no provision for correction when the object class is created, the following action must be taken.

a) Creation

[Example of the pre-correction object class]

serialVersionUID is not supported in the pre-correction object class.

```

package com.fujitsu.sample;
import java.io.*;

public class Company implements Serializable {

    private String longName;
    private String shortName;
    private String address;

    public Company(String longName, String shortName, String address) {

        :
        :
    }
}

```

b) Correction

1. The *serialver* command is executed and serialVersionUID obtained for the pre-correction class.

[Method of obtaining serialVersionUID]

```
> serialver com.fujitsu.sample.Company <-class name
com.fujitsu.sample.Company:    static final long
serialVersionUID = 7242562793746307240L; <-serialVersionUID is notified.
>
```

2. serialVersionUID obtained using the serialVersionUID post-correction class is set as the class variable.

[Example of the post-correction object class]

```
package com.fujitsu.sample;
import java.io.*;

public class Company implements Serializable {

    private String  longName;
    private String  shortName;
    private String  address;
    private String  telNumber;                ....(3)
    static final long serialVersionUID = 7242562793746307240L; ....(4)

    public Company(String longName, String shortName, String address, String telNumber ) {
        :
        :
    }
}
```

(3): The telNumber attribute is added.

(4): serialVersionUID obtained using the serialver command is set as the class variable.

Notes

- It is possible to add a unique serialVersionUID to the pre- and post-correction object class for compatibility.
- If the session information backed up in the pre-correction object class is recovered in the post-correction object class, the default ("String" type: [null], "Int" type: [0]) is notified to the added attribute. An error is not notified.
- For details about object serialization, refer to the following Java specification documentation.
<http://download.oracle.com/javase/1.5.0/docs/guide/serialization/index.html>

Part 3 EJB Edition

| | |
|--|-----|
| Chapter 9 Basic Functions of the EJB Service..... | 255 |
| Chapter 10 EJB Application Development..... | 273 |
| Chapter 11 How to Create Entity Beans..... | 275 |
| Chapter 12 How to Call EJB Applications..... | 281 |
| Chapter 13 Customize by EJB Service Operation Command..... | 301 |

Chapter 9 Basic Functions of the EJB Service

This chapter explains the following topics:

- Session Bean
- Entity Bean Optimization
- Message-driven Bean
- Time Monitoring Functions Supported by EJB Service
- Notes in EJB Service

9.1 Session Bean

This section has information on the following:

- STATELESS Session Bean Web Service

9.1.1 STATELESS Session Bean Web Service

STATELESS Session Bean can be published as a Web service endpoint.

Service endpoint interfaces in which methods published as Web services have been defined can be called in SOAP by defining the interface in the STATELESS Session Bean deployment descriptor file.

To develop Web service applications, refer to the EJB application development method and the "Developing Web Services" chapter. For details about the Web service operation, refer to the "Interstage Web Service Operation" chapter.

9.2 Entity Bean

This section has information on the following:

- Managing Entity Bean Instances
- Entity Bean Optimization
- Correspondence of Entity Bean and a Database
- EJB QL

9.2.1 Managing Entity Bean Instances



Point

.....
The number of instances, instance management mode and instance creation mode can be set for each Entity Bean.
.....

Setting the Number of Instances

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.

If data for the number of instances that was set (or greater than this number) is operated on, the instance is reused and the database accessed, therefore processing performance may be slightly affected. For details about using instances, refer to "[12.3 Relationship between Enterprise Bean Instance, EJB Object, and EJB Home](#)".

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.

| 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] > [JServer name] > [Application Status] tab > [Application List] > [EJB application name] > [Deployment Descriptor] > [Detailed Settings [Show]].

The table below lists the standards of instance creation mode selection for the Entity Bean.

| 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. |

9.2.2 Entity Bean Optimization

When creating an Entity Bean of BMP using Interstage Studio, 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.



Windows32/64 Solaris32 Linux32/64

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).

9.2.3 Increasing the Speed of CMP2.0 Multi-Item Searches

The CMP2.0 multi-item search acceleration function is an option that reduces the number of times that SQL statements have to be issued by executing a single SQL statement that performs a batch search for all column data and not just the primary key when the following methods are executed:

- Multi-item finder method of a CMP2.0 Entity Bean
- get accessor method for one-to-many relationship CMR
- get accessor method for many-to-many relationship CMR

When the multi-item finder method of a CMP2.0 Entity Bean was executed in V7.0 and earlier versions, only the primary key was loaded from the database, and EJB objects corresponding to the primary key were returned to the client.

Data other than the primary keys (CMF) was loaded from the database when each EJB object first accessed the CMF. This method is efficient when accessing only the data of a specific primary key because it does not load any surplus data. However, when all the data is loaded from a database, the number of database access operations becomes larger.

When the multi-item finder method of a CMP2.0 Entity Bean is executed in V8.0, the CMP2.0 multi-item search acceleration function loads all the data of a record at once. This enables data to be loaded from a database quickly, even when all the data is loaded.

This function is also an effective way to load the data of multiple Entity Beans when a CMR get accessor method is executed on Entity Beans that are in one-to-many or many-to-many relationships.

Settings can be performed using the Interstage Management Console or the command.

System Settings for Multi-item Search Acceleration

- Interstage Management Console

To speed up multi-item searches of all CMP2.0 Entity Beans and relationships, select [System] > [Settings] tab, and set [CMP2.0 multi-item search acceleration] to 'Apply to all Beans and all relationships'.

The default setting for this parameter is 'Set for each Bean and relationship'. When this setting is enabled, each CMP2.0 Entity Bean and relationship can be set individually.

- Command

Use the *isj2eedmin* command. Refer to "*isj2eedmin*" in the Reference Manual (Commands) for details.

Multi-item Search Acceleration Settings for Individual CMP2.0 Entity Beans

- Interstage Management Console

To determine whether to implement multi-item search acceleration for individual CMP2.0 Entity Beans, select [System] > [WorkUnit] > [WorkUnit Name] > [Application Status] > [Module Name] > [Bean Name] > [Deployment Descriptor] > [Interstage Additional Settings].

- Command

Use *ejbdefimport*.

Refer to the Interstage Management Console Help for information on the Interstage Management Console.

'relationship' CMF Access Speed up Settings

- Interstage Management Console

To make the settings for speeding up the search for multiple CMP2.0 Entity Beans, click [System] > WorkUnit > "WorkUnit Name" > 'Module Name' > "Bean Name" > [Define Application Environment]> [CMR Mapping].

- Command

Use *ejbdefimport*.

Reducing the number of times an SQL statement is issued will reduce the number of communications with a database and so improve search performance.

- Multi-item finder method of a CMP2.0 Entity Bean
- get accessor method for one-to-many relationship CMR
- get accessor method for many-to-many relationship CMR

The difference in SQL statements that are issued with and without the multi-item search acceleration option is shown below.

When the Multi-item Search Finder Method is Executed

The explanation that follows uses the following example:

- Multi-item finder method: findAll
- CMF: id and name
- Column corresponding to CMF: id and name
- DBMS table name: EMPLOYEE

When the Multi-item Search Acceleration Option is not Used

The following table shows the timing of SQL statements.

| Example with multi-item finder method (findAll) | Timing of SQL statements |
|---|---|
| Collection employees = employeeHome.findAll(); | SELECT id FROM EMPLOYEE |
| Iterator iter = employees.iterator(); | - |
| while (iter.hasNext()) { | |
| EmployeeLocal e = (EmployeeLocal)iter.next(); | |
| System.out.println("Id: " + e.getId()); | SELECT name,age,...FROM EMPLOYEE WHERE id = ? |
| System.out.println("Name: " + e.getName()); | - |
| } | |

The Primary Key field (id) is loaded when the multi-item finder method (findAll) is executed.

All CMF fields are loaded the first time that a Bean instance accesses the CMF (when the getId method is executed).

As the following diagram shows, the total number of SQL statements issued is $N + 1$ ("+1" because search (1) is necessary).

When the Multi-item Search Acceleration Option is Used

The following table shows the timing of SQL statements.

| Example with multi-item finder | Timing of SQL statements |
|---|---|
| Collection employees = employeeHome. findAll (); | SELECT id,name,age,... FROM EMPLOYEE |
| Iterator iter = employees.iterator(); | - |
| while (iter.hasNext()) { | |
| EmployeeLocal e = (EmployeeLocal)iter.next(); | |
| System.out.println("Id: " + e. getId ()); | |
| System.out.println("Name: " + e.getName()); | |
| } | |

When the multi-item finder method (findAll) is executed, an SQL statement that obtains all column data is issued.

The container obtains all records from ResultSet and caches them in memory. During CMF access, the values are retrieved from the cache.

When the multi-item finder method is called, all the CMF (including the PK field) is loaded, so the total number of SQL statements issued is just 1, as shown in the following diagram.

When the Get Accessor Method for a One-to-many or a Many-to-many Relationship is Executed

The explanation that follows uses the following example:

- get accessor method: getEmployees method
- Database table name: EMPLOYEE

When the Multi-item Search Acceleration Option is not Used

The following table shows the timing of SQL statements.

| Example using one-to-many or many-to-many relationship accessor | Timing of SQL statements |
|---|--|
| CompanyLocal fj = companyHome. findByPrimaryKey ("FJ"); | SELECT name FROM COMPANY WHERE name = ? |
| Collection employees = fj.getEmployees(); | SELECT fk_employee_id FROM COMPANY_EMPLOYEE WHERE fk_company_name = ? |
| Iterator iter = employees.iterator(); | - |
| while (iter.hasNext()) { | |
| EmployeeLocal e = (EmployeeLocal)iter.next(); | |
| System.out.println("Id: " + e. getId ()); | SELECT name,age,... FROM EMPLOYEE WHERE id = ? |
| System.out.println("Name: " + e.getName()); | - |
| } | |

When the get accessor method of a CMR in a one-to-many or many-to-many relationship is called, the Primary Key field is loaded from the Join table.

All the CMF is loaded the first time that a Bean instance accesses the CMF.

As the following diagram shows, the total number of SQL statements issued to the table becomes $N + 2$ ("+2" because searches (1) and (2) are necessary).

When the Multi-item Search Acceleration Option is Used

The following table shows the timing of SQL statements.

| Example using one-to-many or many-to-many relationship accessor | Timing of SQL statements |
|---|--|
| CompanyLocal fj = companyHome. findByPrimaryKey ("FJ"); | SELECT name FROM COMPANY WHERE name = ? |
| Collection employees = fj.getEmployees (); | SELECT e.id,e.name,e.age, ... FROM EMPLOYEE e LEFT OUTER JOIN COMPANY_EMPLOYEE ce ON e.id = ce.fk_employee_id WHERE ce.fk_company_name = ? |
| Iterator iter = employees.iterator(); | - |
| while (iter.hasNext()) { | |
| EmployeeLocal e = (EmployeeLocal)iter.next(); | |
| System.out.println("Id: " + e.getId()); | |
| System.out.println("Name: " + e.getName()); | |
| } | |

When a one-to-many or many-to-many relationship accessor method is executed, an SQL statement that obtains all column data is issued.

The container obtains all records from ResultSet and caches them in memory. During CMF access, the values are retrieved from the cache.

When the relationship accessor method is called, all the CMF (including the PK field) is loaded, so the total number of SQL statements issued is 2, as shown in the following diagram.

Note

- When the instance management mode is ReadOnly, record data accessed once within a process is cached. If the same record is accessed again, the cached data will be used if the primary key is known.

This means that when this option is set, performance may be inferior to the default setting, which only searches for the primary key during a multi-item search. For this reason, this option setting is disabled.

- If this function is set with a finder method that has the DISTINCT clause (which returns results with duplicated rows removed) specified in an EJB QL query, the setting will be disabled.

This is because when this function is used with a finder method with the DISTINCT clause specified, duplication checks will be performed not only on the Primary Key field but on the CMF field as well, and performance is better when this function is not used and duplication checks are performed on the Primary Key field only.

Also, some databases use data types that do not support joint use of the DISTINCT clause.

If an attempt is made to use this function with EJB QL that uses the DISTINCT clause, a warning will appear when the search starts.

9.2.4 Correspondence of Entity Bean and a Database

When updating is performed on a single item of data from more than one CMP2.0 Entity Bean, the function for guaranteeing the integrity of data updated by CMP2.0 Entity Beans can be used to assure data integrity and prevent deadlock errors from occurring.

To prevent deadlock errors, a database will add the FOR UPDATE clause to the SELECT statement used to search tables and apply an update lock to rows that are searched. This guarantees data integrity, from searches through to updates, when multiple transactions access the same record.

An update lock guarantees data integrity until a transaction process finishes by locking rows that are referenced during search processing.

Rows that are subject to an update lock cannot be referenced by another transaction.

When a row update lock is used for each CMP2.0 Entity Bean, the update lock is applied during the following processes that cause a container to search a database:

- finder method
- ejbSelect method
- Accessor method to a CMR
- CMF loading

The setting is performed with the Interstage Management Console by selecting [System[> [WorkUnit] > [WorkUnit Name[> [Application Status] > [Module Name] > [Bean Name] > [Deployment Descriptor] > [CMF Mapping Definition]. The default setting is Disabled.

Note

- Search processes that use aggregate functions (such as MAX) that are executed by ejbSelect are not subject to the update lock because they are not permitted to use FOR UPDATE due to database specifications.

If this function is used with a database that does not support the FOR UPDATE clause, an error message will appear and the function will either fail to start or fail to activate.

Check the database manual to determine if the FOR UPDATE clause can be used.

- If the database uses Symfoware and a relationship is specified for EJB applications, an error message will appear and the function will either fail to start or fail to activate when this function is used.
-

9.2.5 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.
-

9.3 Message-driven Bean

For a Message-driven Bean, you can select either JMS or the resource adapter as the target for the receipt of messages (called a "receipt target type"), as described below. After that, this section has information on the following:

- Durable Subscription Function
- Message Backup Function in Abnormal Circumstances

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] > [IIServer name] > [Application Status] > [Application List] > [EJB application name] > [Deployment Descriptor] > [Message-Driven Bean Additional Settings].

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 |

Resource Adapter Definitions

To run a Message-driven Bean, define 'Resource adapter name' on [Message-driven Bean extended information]. To do so, from the Interstage Management Console, select [WorkUnit] > [JServer name] > [Application Status] > [Application List] > [EJB application name] > [Deployment Descriptor] > [Message-Driven Bean Additional Settings].

The default value of each definition is as follows:

| Definition name | Default value |
|-----------------------|----------------------|
| Resource adapter name | EJB application name |

9.3.1 Durable Subscription Function

This functionality is enabled when the receipt target type is JMS.

To use this functionality, set "NonDurable" or "Durable" for subscription-durability (Subscriber persistence) in the deployment descriptor. Settings in or changes to the deployment descriptor are made in Interstage Studio. In EJB applications of EJB2.0 and earlier, these can also be made in [WorkUnit] > [JServer name] > [Application Status] > [Application List] > [EJB application name] > [Deployment Descriptor] > [Detailed Settings [Show]] > [EJB Application Properties] > [Duration Type Subscriber Persistence] of the Interstage Management Console.

Register and Delete Durable Subscriber Definition

If 'Durable' is specified on 'Duration Type Subscriber persistence', specify 'Subscriber name.' JServer registers the Durable Subscriber with the name specified for 'Subscriber name' when the Message-driven Bean is started for the first time.

'Duration Type Subscriber persistence' is set using the Interstage Management Console.

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 the "JMS Operation Commands" chapter 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
```

9.3.2 Message Backup Function in Abnormal Circumstances

This functionality is enabled when the receipt target type is JMS.

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

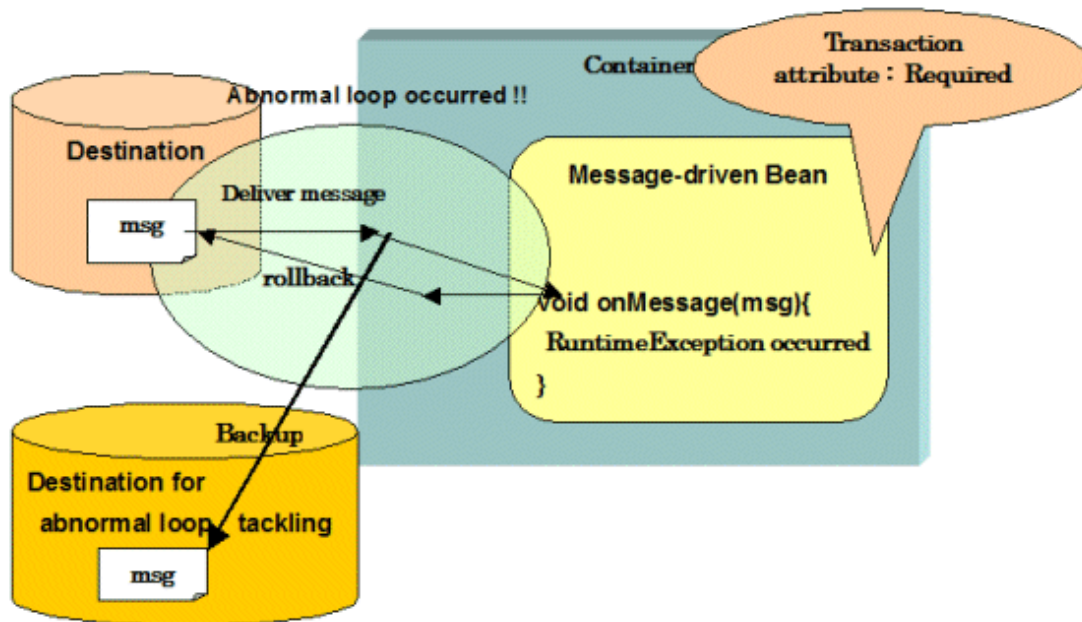
Windows32/64

```
C:/Interstage/EJB/var
```

Solaris32/64 Linux32/64

```
/opt/FJSVejb/var
```

The processing image when this function is used is shown as follows.



Example

How to Restore the Serialized Message

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();
}
```



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.

9.4 Time Monitoring Functions Supported by EJB Service

The EJB Service supports the following time monitoring functions:

- Maximum Time Monitoring Function for Application Processing
- Waiting Time Monitoring Function for Server Return
- Idle-time Monitoring Function of STATEFUL Session Bean
- Timer Deletion of EJB Object
- EJB Timer Service

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 ejbRemove 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. |

| Function | Monitoring | Explanation |
|-------------------|--|---|
| EJB Timer Service | This can monitor any monitoring target in the EJB application. | This functionality creates a timer in the application and can execute callback processing from the EJB container at any time. |

Timeout Setting of each Function

The table below lists the settings of each function for time monitoring.

EJB Timer Service creates the timer in the EJB application, unlike other time monitoring functionality. For details about the method to create the timer, refer to "[9.5 EJB Timer Service](#)".

| 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 "config" in the "CORBA Service Environment Definition" appendix of 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. |
| 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. |

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

9.4.1 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.



Note

If 'Warning message' is selected in 'Forcefully end application on timeout' on the Interstage Management Console or isj2eeadmin command, When the timeout is exceeded for the first time, a message displays to inform the user that the timeout has been exceeded. The process used to display the messages is stopped for 10 minutes

If 'Forcefully stop all the running processes' is selected, the second thread dump output is processed after 10 seconds. This prevents the process being stopped by force before the thread dump has been output.

Because the second thread dump is output 10 seconds after the 'timeout exceeded' message, and the process is forcibly stopped after a further 10 seconds, the process does not stop for 20 seconds. So, even if a message is output, the application might be returned normally until the process stops, and the process may be forcibly stopped after that.

9.4.2 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 "Timeout Monitoring during Operation of CORBA Applications" in the "Designing the OLTP Server" chapter of the OLTP Server User's Guide.

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.

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.

9.4.3 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 | <code>java.rmi.NoSuchObjectException</code> is returned to the client. |
| Local interface | <code>javax.ejb.NoSuchObjectLocalException</code> 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.

9.4.4 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] > [IIServer name] > [Application Status] > [Application List] > [EJB application name] > [Deployment Descriptor] > [Detailed Settings [Show]] > [Interstage Additional Settings] from the Interstage Management Console. Refer to Help for the Interstage Management Console for details of the setting procedure.



Note

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.

9.5 EJB Timer Service

EJB Timer Service is functionality that was added from the EJB2.1 specs. In this functionality, the timers shown below can be created so that callback processing can be executed from the EJB container at any time.

- Callback after a fixed time is exceeded
- Callback after a fixed time is exceeded, and thereafter at fixed intervals
- Callback at a specified time
- Callback at a specified time, and thereafter at fixed intervals

Accordingly, as an example, when you are shopping using a service in which points accumulate, it is possible to create event processing that uses a timer so that every day at 0:00, the points are added up, and a promotional message(=mail) is delivered to the customer for whom the points have accumulated.



Note

This functionality creates the timer in the EJB application. If the IIServer process that is used to create the timer is stopped (this includes being stopped forcibly because there is not enough memory), the timer is also stopped. For this reason, the timer may stop unexpectedly, so it is recommended that some other time monitoring functionality is used if you want a timer to be running constantly.

EJB Applications that can be Used

EJB Timer Service can be used in STATELESS Session Bean and Message-driven Bean.

- Types of Enterprise Bean that can be used:
 - STATELESS Session Bean
 - Message-driven Bean
- Types of Enterprise Bean that cannot be used:
 - STATEFUL Session Bean
- Types of Enterprise Bean for which there are restrictions:
 - Entity Bean (Note)

Note: Callback processing (the `ejbTimeout` method) using the timer is not executed in this version for Entity Bean. The IIServer21149 WARNING message is also output when the IIServer that the Entity Bean using EJB Timer Service was deployed to is started.

Basic Functionality

Timer generation method

The timer is generated by executing the EJB Timer Service createTimer method. Specify the execution time and execution interval in the createTimer method argument. EJB Timer Service executes or gets the EJB context getTimerService method passed from the EJB container in Enterprise Bean.

Timer cancellation method

To cancel the timer that was generated, execute the cancel method for the timer that was generated by Enterprise Bean. The timer can be generated by executing the getTimers method for EJB Timer Service that executed/got the getTimerService method in Enterprise Bean. It is passed in the callback processing (ejbTimeout method) argument.

Execution of callback processing

Enterprise Bean callback processing (ejbTimeout method) is executed at the time specified when the timer is generated using EJB Timer Service. Enterprise Bean must implement the javax.ejb.TimerObject interface.

EJB Timer Service Validity Range

The timer can only be used in generated processes. For this reason, do not get the timer remotely using the Enterprise Bean Remote interface or a Web service endpoint. Its use in clustered structures and process concurrency environments is not supported either. If an EJB application in which EJB Timer Service has been implemented is deployed and started on an JJSERVER for which the process concurrency is 2 or more, the JJSERVER21148 error occurs.

The timer that is generated in EJB Timer Service is deleted when the deployed module is inactive. For this reason, if the following operations are performed, re-register the timer if necessary.

- The JJSERVER is restarted
- Undeploy or deploy by overwriting is performed using the HotDeploy functionality
- The process is restarted automatically after the JJSERVER ends abnormally

Timer Accuracy and Behavior when there is a Processing Delay

If the load is concentrated, the timer start time may jolt out of alignment.

If timer processing does not complete immediately and the execution time of a follow-on timer is received, the follow-on timer waits until completion of the timer that has been executed. After the timer that has been executed is complete, the timer for which the planned execution time has been exceeded is executed immediately. Timers that are executed at fixed intervals are executed consecutively for the number of processes when the planned execution time is exceeded.

Transactions

When the timer is generated

Timers that are generated within the transaction range are deleted when the transaction is rolled back.

When the timer is cancelled

When timers within the transaction range are cancelled, the cancellation is invalidated when the transaction is rolled back. New timers are deleted. Timers that existed before the transaction range continue running.

When callback processing is executed

When callback processing is executed, the transaction runs according to the transaction attribute that is defined in the EJB application, as shown in the table below.

| Transaction attribute | EJB container processing |
|-----------------------|--|
| RequiresNew | The EJB container starts a new transaction and then executes callback processing. |
| Required | If processing in the transaction is rolled back, or an exception occurs, EJB Timer Service retries callback processing once only. If processing in the transaction is rolled back, or an exception occurs again during retry, the JJSERVER21147 WARNING message is output to the container log and timer processing continues. |

| Transaction attribute | EJB container processing |
|-----------------------------------|---|
| Supports NotSupported Never | EJB container executes callback processing without starting the transaction. |
| Mandatory | This cannot be specified. If it is, the IJServer21095 ERROR message is output to the container log when the EJB application is active. |

Instances on which Callback Processing is Executed

Callback processing (ejbTimeout method) is executed on the following instances:

- STATELESS Session Bean

Callback processing is executed on pooled instances.

- Message-driven Bean

Callback processing is executed on new instances.

(The instance is deleted after callback processing is executed.)

- Entity Bean

Not supported. Callback processing is not executed.

The Method of Creating Applications that Use EJB Timer Service

The method of creating applications that use EJB Timer Service is explained in the following section.

9.5.1 EJB Timer Service Access Method

The getTimerService method that is defined in the EJB context (javax.ejb.EJBContext) interface passed from the EJB container is used to access EJB Timer Service from the EJB application.

The EJB context is passed from the EJB container in the setSessionContext method, setEntityContext method, and setMessageDrivenContext method that are defined in each EJB application (javax.ejb.SessionBean, javax.ejb.EntityBean, javax.ejb.MessageDrivenBean) interface.

9.5.2 The Monitoring Start Method

The createTimer method that is defined in the EJB Timer Service (javax.ejb.TimerService) interface is executed and the timer object (javax.ejb.Timer) is generated. The monitoring time countdown starts from the time when the timer object is generated. The types of createTimer method are shown in the table below. The unit of time is milliseconds.

API Defined in the javax.ejb.TimerService interface

| Number | Method name | Explanation |
|--------|--|--|
| 1 | Public javax.ejb.Timer createTimer (java.util.Date initialExpiration, long intervalDuration, java.io.Serializable info) | The timer is generated once at the specified time (initialExpiration), and thereafter at fixed intervals (intervalDuration). If there is information you want to pass, specify it in 'info' (if there is none, specify 'null'). |
| 2 | Public javax.ejb.Timer createTimer (java.util.Date expiration, java.io.Serializable info) | The timer is generated once only at the specified time (expiration). If there is information you want to pass, specify it in 'info' (if there is none, specify 'null'). |

| Number | Method name | Explanation |
|--------|---|--|
| 3 | Public javax.ejb.Timer createTime (long initialDuration, long intervalDuration, java.io.Serializable info) | The timer is generated once from the start of monitoring until the specified time for the maintaining of monitoring (initialDuration), and thereafter at fixed intervals (intervalDuration). If there is information you want to pass, specify it in 'info' (if there is none, specify 'null'). |
| 4 | Public javax.ejb.Timer createTime (long duration, java.io.Serializable info) | The timer is generated once only from the start of monitoring until the specified time for maintaining the monitoring (duration). If there is information you want to pass, specify it in 'info' (if there is none, specify 'null'). |
| 5 | Public java.util.Collection getTimers () | This gets all active timer objects that are associated with the EJB application that this method was executed in. |

[Additional Information]

As well as the createTime method, the getTimers method is also defined in the javax.ejb.TimerService interface. When the getTimers method is executed, timers generated by the EJB application can be collected. Use the equals method of the timer (javax.ejb.Timer) to check whether the timer objects are the same. (Identity cannot be guaranteed if objects are compared using the "==" operator.)

9.5.3 The Time Monitoring Processing Execution Method

For EJB applications in which you want to use EJB Timer Service, the javax.ejb.Timer interface must be implemented in the Enterprise Bean class. Implement this interface in the Enterprise Bean that uses EJB Timer Service, or in the Enterprise Bean parent class.

Only one ejbTimeout method is declared in the javax.ejb.Timer interface.

Implement this interface in Enterprise Bean as shown below.



Example

Example of Implementation (Stateless Session Bean)

```
public class Enterprise Bean name
    implements javax.ejb.SessionBean, javax.ejb.Timer
{
    ...
    public void ejbTimeout(javax.ejb.Timer timer) {
        // Describe the business logic that is executed at the planned time
        // Here
    }
}
```

When the planned time that is registered in EJB Timer Service is reached, the EJB container calls the ejbTimeout method. Describe the business logic that you want to execute at the planned time in the ejbTimeout method.

9.5.4 The Timer Cancellation/Status Reference Method

The APIs shown below are in the timer (javax.ejb.Timer) interface. It is possible to cancel time monitoring and refer to the status.

APIs Defined in the javax.ejb.Timer Interface

| Number | Method name | Explanation |
|--------|---|---|
| 1 | public void cancel () | Cancels timer monitoring. A timer that is cancelled is deleted from the EJB container. |
| 2 | public javax.ejb.TimerHandle getHandle () | Gets the timer handle object. The TimerHandle interface getTimer method can be used to get timer objects again at a later time. |

| Number | Method name | Explanation |
|--------|---|--|
| 3 | public java.io.Serializable getInfo() | Gets the information (info objects passed to the createTimer method) that was passed when the timer was generated. |
| 4 | public java.util.Date getNextTimeout () | Gets the next planned time for execution of the ejbTimeout method. |
| 5 | public long getTimeRemaining () | Gets the remaining time for the timer until the next execution of the ejbTimeout method. The unit is milliseconds. |

API Defined in the javax.ejb.TimerHandle Interface

| Number | Method name | Explanation |
|--------|-----------------------------------|--|
| 1 | public javax.ejb.Timer getTimer() | Gets the Timer object for this TimerHandle object. |

[Additional Information]

- If the timer is executed once only, the timer is deleted from the container after callback processing (ejbTimeout method) is complete.
- NoSuchObjectLocalException is returned when a method for the timer object that was deleted is executed.

9.5.5 Other

The ejbTimeout method can be used locally, and in this case is not called from the client remotely. Therefore it does not have a client identifier. For this reason, the container returns an authentication object without authentication information when the getCallerPrincipal method is executed in the ejbTimeout method.

9.6 Notes in EJB Service

When using EJB applications by deploying them on JJSERVER, 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 JJSERVERs (except an JJSERVER 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 JJSERVER as the Message-driven Bean in which an error occurred also stop.
- If the EJB applications deployed to the JJSERVER use the same remote or home interface, an error occurs when they are started (except when a local call is used or for an JJSERVER 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 JJSERVER.
- **Windows32/64** **Solaris32** **Linux32/64**
When a distributed transaction is used, two or more JJSERVER processes cannot be started concurrently.
- The EJB2.1 spec-compliant EJB application can only be used when the JJSERVER type is as shown below. An error will occur if you try to deploy the EJB2.1 spec-compliant EJB application to an JJSERVER that is not of the following type:
 - Run Web applications and EJB applications on the same JavaVM

Chapter 10 EJB Application Development

10.1 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 Interstage Studio, which is a component-oriented integrated development support tool by Fujitsu, enables user-friendly view operation that integrates a series of procedures.

Interstage Studio provides the development support function for various EJB applications and client applications and so can improve application development productivity.

10.2 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)
- Descriptions in the file in which the EJB application deployment descriptor information is defined (ejb-jar.xml) must be in XML format. For this reason, follow the XML format features when editing ejb-jar.xml.

Describe the following characters as defined entity references (implicitly defined entities).

| Character to be edited | Defined entity reference |
|------------------------|--------------------------|
| < | < |
| > | > |
| & | & |
| ' | ' |
| " | " |

Do not specify tags that have namespace prefixes.

If this type of tag is specified, the deployment may fail, and it may not be possible to use the name conversion functionality.

Example: <pfx:session>

10.3 Deployment of an EJB Application

Refer to Help for the Interstage Management Console for details of deployment using the Interstage Management Console.

10.4 Debugging an EJB Application

Use the IJServer debug function to debug EJB applications.

Refer to "[2.10 Debugging Applications](#)" for details of the IJServer debug procedure.

10.5 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 "[10.2 Developing an EJB Application](#)" to perform the required steps up to application packaging.

For information on the subsequent work steps, see "[10.3 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.

Chapter 11 How to Create Entity Beans

11.1 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 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.

| 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] > [JServer name] > [Application Status] > [Application List] > [EJB application name] > [Deployment Descriptor], and set data at [CMF mapping definition] and [Finder Method Definition].

Refer to Help for the Interstage Management Console for details on the setting procedure.

When Interstage Studio is used to develop EJB applications, the EJB deployment descriptor editor can also be used for CMP definition.

Refer to the Interstage Studio User's Guide for details on the setting procedure.

For CMP2.0

Set a CMP definition as follows: From the Interstage Management Console, select [WorkUnit] > [JServer name] > [Application Status] > [Application List] > [EJB application name] > [Deployment Descriptor], and set data at [CMF mapping definition].

Make a query definition in advance during creation of an EJB application.

Refer to Help for the Interstage Management Console for details on the setting procedure.

11.2 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, according to the instance management mode to be selected.

| 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. |

| Note | Instance management mode | Action |
|---|--------------------------|--|
| <p>Windows32/64 Solaris32 Linux32/64</p> <p>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.</p> | Sequential | Specify ReadWrite when update processing is done, and ReadOnly when retrieval processing is done. |
| <p>When the transaction attribute is specified as other than Mandatory, then execute retrieving multiple Instances (Enumeration or Collection type) may cause exception.</p> | Sequential | Specify the transaction attribute for each EJB application (each Bean) then execute it. |
| <p>Sequential cannot be set for the instance management mode of the following EJB applications:</p> <p>Entity Bean that is deployed to IJServer and for which "No" is specified for "Local call" on the Interstage Management Console</p> <p>Entity Bean that is deployed to IJServer and for which "No" is specified for "Local call" in customization</p> | Sequential | <p>To run the Entity Bean with Sequential specified for the instance management mode, set "Yes" for "Local call."</p> <p>To perform operation with "No" specified for "Local call", change the instance management mode as follows:</p> <p>To perform update processing, specify ReadWrite.</p> <p>To perform only retrieval processing, specify ReadOnly.</p> |

11.3 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

11.3.1 Standard Data Types

Available Standard Data Types

The standard data types that can be used are as follows:

- 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 "Available Standard Data Types" above, 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 JDBC1.0.

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.

| 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 | - | NUMBER | -(*8) |
| 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 | SMALLINT | NUMBER | SMALLINT TINYINT(*8) |
| java.lang.String | CHAR NCHAR VARCHAR NVARCHAR | CHAR(*2) VARCHAR2(*2) LONG(*2) CLOB(*7) | CHAR VARCHAR TEXT NCHAR NVARCHAR NTEXT |
| java.math.BigDecimal | NUMERIC DECIMAL | NUMBER | DECIMAL NUMERIC MONEY SMALLMONEY |
| byte[] | BLOB (*1) | RAW LONG RAW(*2)(*3) BLOB(*7) | BINARY VARBINARY IMAGE |
| java.sql.Date | DATE | DATE | - |
| java.sql.Time | TIME | DATE | - |
| java.sql.Timestamp | TIMESTAMP | DATE TIMESTAMP (*4) | DATETIME (*5)(*6) SMALLDATETIME (*5) |

-: There is no data type to be recommended specifically.

***1)**

Fetch and update operations cannot be performed for a BLOB type column that equals or exceeds 32 KB 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

This restriction does not apply when the Forward CMP Data Using Stream option is set to "Yes".

This option is set with the CMF mapping definition that is accessed with the Interstage Management Console by selecting [System] > [WorkUnit] > [WorkUnit Name] > [Application Status] > [Module Name] > [EJB Application Name] > [Deployment Descriptor] > [CMF Mapping Definition].

*2)

The data sizes that can be handled for Oracle data types are limited as shown below:

- LONG RAW
- LONG

Refer to details with the manual of JDBC of Oracle for the setBytes method and the setString method.

To handle more data than the above limits, use BMP Entity Beans.

This restriction does not apply when the Forward CMP Data Using Stream option is set to "Yes".

This option is set with the CMF mapping definition that is accessed with the Interstage Management Console by selecting [System] > [WorkUnit] > [WorkUnit Name] > [Application Status] > [Module Name] > [EJB Application Name] > [Deployment Descriptor] > [CMF Mapping Definition].

*3)

If Mandatory is specified for the transaction attribute of a CMP1.1 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.

*4)

This is a data type supported in Oracle10g and later.

*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)

*7)

The following conditions must be satisfied for the use of Oracle BLOB/CLOB data types:

- The database server must be Oracle10g R2 or later
- The driver must be a Thin or OCI driver in Oracle10g R2 or later

*8)

For TINYINT, the recommended types are short in SQL Server.

11.3.2 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 "[11.3.1 Standard Data Types](#)".



.....
Data types of the Home interface and Remote interface cannot be used in a CMF.
.....

Chapter 12 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

12.1 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 '[3.10 Referencing Objects](#)' in 'JNDI' 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.

12.2 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.

12.2.1 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.



If the [Conditions] shown below do not apply, the [Methods that cannot be used] that is shown below cannot be used in the Collection interface that is returned from the finder method. If it is used, java.lang.UnsupportedOperationException is returned.

[Conditions]

- CMP1.1 and BMP

This depends on the IJServer type, as shown below. If the conditions shown below are not satisfied, the usable range will be V6.0 and lower.

- Web applications and EJB applications run on the same JavaVM

Specify a transaction attribute that is not "Mandatory" for the transaction attribute.

- Web applications and EJB applications run on separate JavaVMs, or only EJB Web applications are run

Either specify a transaction attribute that is not "Mandatory" for the transaction attribute, or, if "Mandatory" is specified for the transaction attribute, make the setting for "Use Local Calls?" in [Application Environment Settings] of the Interstage Management Console "No".

- There are no conditions in CMP2.0.

[Methods that cannot be used]

- 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.
-

12.3 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.

To reflect the record data in BMP, execute the `ejbStore` method. In CMP1.1 and CMP2.0, the container executes the UPDATE statement for the database.

If processing is executed in multiple transactions simultaneously, a deadlock may occur when the SELECT statement and UPDATE statements are executed simultaneously.

In this case, specify FOR UPDATE in the SELECT statement.

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.

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 |

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.

12.4 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 'J Business Kit Online Manual of Interstage Studio'.

12.4.1 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.

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.FJCNTxFactoryForClient"); // 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.

Note

- Perform the operation to obtain InitialContext (above 1 and 2) in the constructor, so that it is performed only once in an application.
- See "[3.2.1 Environment Setup in 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 Interstage Studio is automatically packed in a jar file. For details of applet development using Interstage Studio, refer to the Interstage Studio User's Guide.

If Interstage Studio 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

Windows32/64

```
jar cvf SampleApplet.jar *.class samplepkg\*.class
```

Solaris32/64 Linux32/64

```
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.

12.4.2 Client Setup (Pre-installed Version Java Library)

Setting Permission for Java Libraries

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 text on the [Policy Entry] screen:
 - "Specify the directory that was used to install JDK or JRE in %JAVA_HOME% (Example: C:/Interstage/JDK6)"

| Item | Set Value |
|------------|--|
| CodeBase: | <For JDK> file:<CORBA client installation directory>/etc/class/ODjava4.jar (*1) file:/ %JAVA_HOME%/jre/lib/- <For JRE> file:<CORBA client installation directory>/etc/class/ODjava4.jar (*1) file:/ %JAVA_HOME%/lib/- |
| signed by: | (None) |

*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.

Permission necessary for usual operation

| Permission type | Setting Permission | | |
|---------------------|--------------------|------------------------------|----------------------|
| | Permission | Target Name: | Actions: |
| Run time permission | RuntimePermission | loadLibrary.DLL name (*2) | Setting not required |
| Property permission | PropertyPermission | com.fujitsu.* | read |

These permissions ensure security during usual operation.

*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 | Dynamic Link Library |
|--|----------------------|
| CORBA Service Client (Client Function) | ODjava4 |
| CORBA Service (Server Function) | ODjvas4 |

Note: This is not valid for Linux (64 bit).

Permission necessary to collect internal logs of CORBA service (*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 (*4) | read |

*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.

*4 %OD_HOME% specifies the installation directory of the CORBA Service or the CORBA Service client. Default is C:\Interstage\ODWIN.

6. Select File | Save from the menu bar.
7. 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 "12.5.1 Digital Signature" for details of specifying policy files.

12.4.3 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 <PARAM> 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

Windows32/64

The names and locations of jar files for Portable-ORB are listed below.

| File names | Storage location |
|-----------------------|------------------------|
| ODporbROI4_plugin.jar | C:\Interstage\Porb\lib |

Information

The following files can also be used instead of ODporbROI4_plugin.jar:

- ODporb4_plugin.jar
- CosNaming4_plugin.jar
- InterfaceRep4_plugin.jar
- ODroi4_plugin.jar

Solaris32/64 Linux32/64

The names and locations of jar files for Portable-ORB are listed below.

| File names | Storage location |
|-----------------------|-------------------|
| ODporbROI4_plugin.jar | /opt/FJSVporb/lib |

Information

The following files can also be used instead of ODporbROI4_plugin.jar:

- ODporb4_plugin.jar
- CosNaming4_plugin.jar
- InterfaceRep4_plugin.jar
- ODroi4_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 below.

Windows32/64

Names and Location of jar File for EJB Service Clients

| File names | Storage location |
|------------------------------|-------------------------------|
| - fjcontainer94_plugin.jar | C:\Interstage\EJB\lib |
| - fjentity94_plugin.jar (*1) | or C:\Interstage\EJBCL\lib |

*1 This is required only when the Entity Bean is accessed directly from the client.

Solaris32/64 Linux32/64

Names and Location of jar File for EJB Service Clients

| File names | Storage location |
|------------------------------|-------------------------------|
| - fjcontainer94_plugin.jar | /opt/FJSVejb/lib |
| - fjentity94_plugin.jar (*1) | or C:\Interstage\EJBCL\lib |

*1 This is required only when the Entity Bean is accessed directly from the client.

 **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.

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,
ODporbROI4_plugin.jar,fjcontainer94_plugin.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>

```

Information

- 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 Solaris/Linux system as a Web server, instead of setting the path, you can substitute a link file.
- PORB HOME is specified by the <PARAMNAME> 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 Internet Explorer. For details of the coding method, refer to the 'J Business Kit Online Manual of Interstage Studio'.

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

Example

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", // 1
"com.fujitsu.Interstage.ejb.jndi.FJCNCtxFactoryForClient");
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.



Note

- 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 "[12.5 Digital Signature in Applets](#)" for information on signing.



See

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 Interstage Studio is automatically packed in a jar file. For details of applet development using Interstage Studio, refer to the 'Interstage Studio User's Guide'.

If Interstage Studio 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.



Example

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%)
```

Note

- 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.

12.4.4 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.

| 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.

| Operation Environment Files | File Name |
|--|------------------|
| Environment Definition File | config |
| Object References Information Storage File | initial_services |
| Object References Search Information File | initial_hosts |

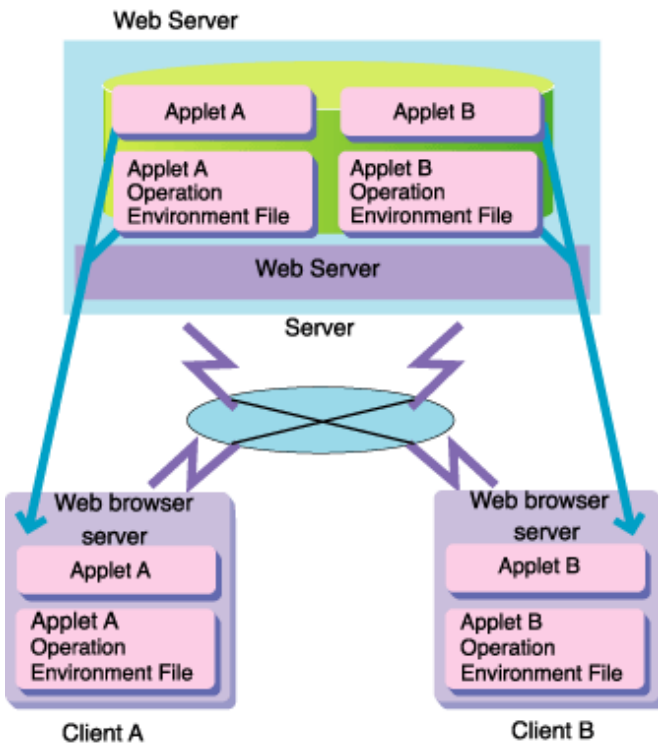
You need to store these operation environment files in the Web server's document root sub directory.

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 the following figure.

Using Different Operation Environments with Different Applets



Applet A that executes on Client A uses the Applet A operation environment file, and Applet 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.

```

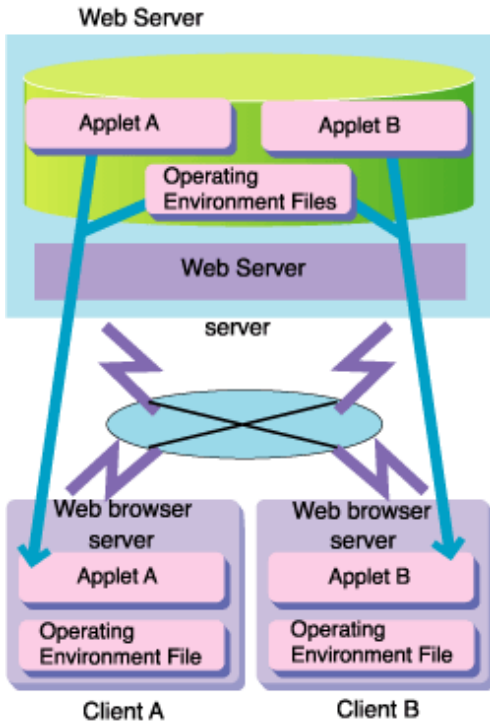
/-
- [Web] - [applet ] - [appletA] - appletA.html
                              - appletA.class
                              - [etc] - appletA operation environment file
[appletB] - appletB.html
                              - appletB.class
                              - [etc] - appletB operation environment file

```

Information

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



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] - ODporb4.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>
```

Information

- 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

Example

Setting Example

```
jbk.plugin.vmooption=-classpath C:\Interstage\JBKDI\jre6\lib\isjeb.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
```

12.5 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 'J Business Kit Online Manual of Interstage Studio'.

12.5.1 Digital Signature

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, 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](#)".

See

Refer to JDK documentation for details about the signature tools. When using the J Business Kit, refer to the 'J Business Kit Online Manual of Interstage Studio'.

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.

Note

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 "Digital Signature Procedures" - "policytool Command Setting (Supplements)" in the chapter "Java Programming Guide "of the "Distributed Application Development Guide (CORBA Service Edition)".

12.6 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 13 Customize by EJB Service Operation Command

This explains how to customize the EJB application execution environment using the operation command of Customize Tool.

Customization using the EJB service operation command allows you to edit the EJB application runtime environment definitions ("Enterprise Bean Definition information"), based on the XML format definition file that you can update.

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 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.

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.

13.1 Customize Flow

Customize edits each type of definition information by executing the `ejbdefimport` command, based on the definition file that is exported from the EJB server by using the `ejbdefexport` command or that is newly created.

This section explains each definition file and the commands that are used for customize.

Definition Files

The table below lists the definition files to be used to customize flow.

| Definition files | Description |
|----------------------------------|--|
| Enterprise Bean Definition files | This file represents the Enterprise Bean definition information in XML format. |

Commands Used in Customize

The table below lists the command used to customize flow.

| Commands | Function |
|---------------------------|--|
| <code>ejbdefexport</code> | Exports the Enterprise Bean definition information to the Enterprise Bean definition file. |
| <code>ejbdefimport</code> | Imports the Enterprise Bean definition file to the Enterprise Bean definition file. |

For details on each command, refer to the Reference Manual (Command Edition).

13.2 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

Windows32/64

- 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
```

Solaris32/64 **Linux32/64**

- 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

Windows32/64

- 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
```

Solaris32/64 **Linux32/64**

- 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

When the `ejbdefexport` command is used, the defined values are extracted 'as is'. If there is an inconsistency in the Enterprise Bean definition information created during deployment and `ejbdefimport` is executed using the Enterprise Bean definition information that was exported using the `ejbdefexport` command, the `ejbdefimport` command checks the inconsistency and may fail to import the definition information, even though the inconsistency is ignored from an operations perspective.

13.3 Contents of Enterprise Bean Definition File

This describes the Enterprise Bean definition file in XML format.

The file contents are as follows.

The Enterprise Bean definition file is structured according to the following format:

```
<ejbdef>
  <ejb-jar>
    <!--ejb-jar tag lower tag-->
  </ejb-jar>
  <fujitsu-bean-definition>
    <!--fujitsu-bean-definition tag lower tag-->
  </fujitsu-bean-definition>
  <fujitsu-cmp-definition(or fujitsu-cmp2x-mapping-definition)>
    <!--fujitsu-cmp-definition tag (or fujitsu-cmp2x-mapping-definition tag)lower tag-->
  </fujitsu-cmp-definition (or /fujitsu-cmp2x-mapping-definition)>
</ejbdef>
```

 **Note**

- The fujitsu-cmp-definition tag and the fujitsu-cmp2x-mapping-definition tag cannot be defined at the same time.
- Definitions under the ejb-jar tag cannot be imported or exported in EJB2.1-compliant EJB applications. Edit the ejb-jar.xml file in the development environment before deployment.

Tag structure and contents

- ejb-jar tag structure and contents
- fujitsu-bean-definition tag structure and contents
- fujitsu-cmp-definition tag structure and contents
- fujitsu-cmp2x-mapping-definition tag structure and contents

How to read the table

- Tag name: XML tag name in definition file
- Value: XML value in definition file
- Meaning: Tag meaning (the Customize Tool displays its item names.)
- Editing: Whether values can be edited (YES or NO)
- Correspondence to Interstage Management Console screen: Correspondence to customize Tool screen
- ejb-jar tag structure and contents
- fujitsu-bean-definition tag structure and contents
- fujitsu-cmp-definition tag structure and contents
- fujitsu-cmp2x-mapping-definition tag structure and contents

ejb-jar tag structure and contents

The ejb-jar tag is only valid for EJB2.0 specification-compliant EJB applications and earlier versions. The tags for using EJB2.0 specification-compliant EJB applications are explained in the table below.

| Tag name | | Meaning |
|----------|--------------------------|----------------------------------|
| ejb-jar? | | |
| | enterprise-beans | |
| | session+ | Session Bean-related definitions |
| | entity+ | Entity Bean-related definitions |

| Tag name | | Meaning |
|----------|----------------------|--|
| | message-driven+ | Message-driven Bean-related definitions |
| | relationships? | CMR mapping-related definitions |
| | assembly-descriptor? | Security and transaction-related definitions |

<ejb-jar><enterprise-beans><session> lower tags

| Tag name | Value | Meaning | Editing | Correspondence to Interstage Management Console Screen |
|-------------------|---|----------------------------|---------|--|
| ejb-name? | Any character string | Enterprise Bean name | NO | EJB Application Information |
| home? | Any character string | Home interface name | NO | |
| remote? | Any character string | Remote interface name | NO | |
| local-home? | Any character string | Local Home interface name | NO | |
| local? | Any character string | Local interface name | NO | |
| ejb-class? | Any character string | Enterprise Bean class name | NO | |
| session-type? | Choose from the following values - Stateful - Stateless | Session Type | NO | |
| transaction-type? | Choose from the following values - Bean - Container | Transaction Type | YES | |
| env-entry+ | | | | Environment Property |
| description? | Any character string | Explanation | NO | |
| 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 | NO | |
| env-entry-type | Choose from the following values - java.lang.Boolean - java.lang.String | Type | YES | |

| Tag name | Value | Meaning | Editing | Correspondence to Interstage Management Console Screen |
|----------|--|---------|---------|--|
| | <ul style="list-style-type: none"> - java.lang.Integer - java.lang.Double - java.lang.Byte - java.lang.Short - java.lang.Long - java.lang.Float - java.lang.Character | | | |
| | <p>env-entry-value?</p> <p>Choose from the following values</p> <ul style="list-style-type: none"> - java.lang.Boolean: True/False Note: This value cannot be omitted (a null char string cannot be specified). - java.lang.String: Any character string - java.lang.Integer: -2147483648 to 2147483647 - java.lang.Double: -1.7976931348623157E308 to 1.7976931348623157E308 - java.lang.Byte: -128 to 127 - java.lang.Short: -32768 to 32767 - java.lang.Long: -9223372036854775808 to 9223372036854775807 - java.lang.Float: -3.4028234663852886E38 to 3.4028234663852886E38 - java.lang.Character: Any character string | Value | YES | |

| Tag name | Value | Meaning | Editing | Correspondence to Interstage Management Console Screen |
|-------------------------|---|------------------------------|---------|--|
| ejb-ref+ | | | | EJB Reference |
| description? | Any character string | Explanation | NO | |
| ejb-ref-name | Any character string | Enterprise Bean JNDI name | NO | |
| ejb-ref-type | Choose from the following values - Session - Entity | Enterprise Bean Type | NO | |
| home | Any character string | Home interface name | NO | |
| remote | Any character string | Remote interface name | NO | |
| ejb-link? | Any character string | Enterprise Bean name | NO | |
| ejb-local-ref+ | | | | Local EJB Reference |
| description? | Any character string | Explanation | NO | |
| ejb-ref-name | Any character string | Enterprise Bean JNDI name | NO | |
| ejb-ref-type | Choose from the following values - Session - Entity | Enterprise Bean Type | NO | |
| local-home | Any character string | Local Home interface name | NO | |
| local | Any character string | Local interface name | NO | |
| ejb-link? | Any character string | Enterprise Bean name | NO | |
| security-role-ref+ | | | | Security Role Reference |
| description? | Any character string | Explanation | NO | |
| role-name | Any character string | Code base Security role name | NO | |
| role-link? | Any character string | Code base Security role name | NO | |
| security-identity? (*1) | | | | Security Identity |
| description? (*1) | Any character string | Explanation | NO | |
| use-caller-identity | None (this tag and 'run-as' are mutually exclusive) | Caller Security is used | YES | |
| run-as | | | | |

| Tag name | | Value | Meaning | Editing | Correspondence to Interstage Management Console Screen |
|-------------------|-----------------------|--|-----------------------|---------|--|
| | description? | Any character string | Explanation | YES | |
| | role-name | Any character string | Security role name | YES | |
| resource-ref+ | | | | | Resource Reference |
| | description? | Any character string | Explanation | NO | |
| | res-ref-name | Any character string | Resource manager name | NO | |
| | res-type | Any character string | Class/Interface name | NO | |
| | res-auth | Choose from the following values - Application - Container | Resource authority | NO | |
| resource-env-ref+ | | | | | Resource Environment Reference |
| | description? | Any character string | Explanation | NO | |
| | resource-env-ref-name | Any character string | Resource manager name | NO | |
| | resource-env-ref-type | Any character string | Class/Interface name | NO | |

*1 This tag is only valid if 'run-as' is specified.

<ejb-jar><enterprise-beans><entity> lower tags

| Tag name | | Value | Meaning | Editing | Correspondence to Interstage Management Console Screen |
|-------------|--|----------------------|----------------------------|---------|--|
| ejb-name? | | Any character string | Enterprise Bean name | NO | EJB Application Information |
| home? | | Any character string | Home interface name | NO | |
| remote? | | Any character string | Remote interface name | NO | |
| local-home? | | Any character string | Local Home interface name | NO | |
| local? | | Any character string | Local interface name | NO | |
| ejb-class? | | Any character string | Enterprise Bean class name | NO | |

| Tag name | Value | Meaning | Editing | Correspondence to Interstage Management Console Screen |
|-----------------------|---|-----------------------|---------|--|
| persistence-type? | Choose from the following values - Bean - Container | Persistence Type | NO | |
| prim-key-class? | Any character string | PrimaryKey Class name | NO | |
| reentrant? | Choose from the following values - True - False | Re-entrant type | NO | |
| cmp-version? | Choose from the following values - 1.x - 2.x | CMP version | YES | |
| abstract-schema-name? | Any character string | Abstract schema name | YES | |
| primkey-field? | Any character string | PrimaryKey field name | NO | |
| env-entry+ | Refer to env-entry of session tag. | | | |
| ejb-ref+ | Refer to ejb-ref of session tag. | | | |
| ejb-local-ref+ | Refer to ejb-local-ref of session tag. | | | |
| security-role-ref+ | Refer to security-role-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. | | | |
| query+ | | | | query |
| description? | Any character string | Explanation | NO | |
| query-method | | | | |
| method-name | Any character string | method | NO | |
| method-params | | | | |
| method-param+ | Any character string | parameter | NO | |
| result-type-mapping? | Choose from the following values - Local - Remote | result type | NO | |
| ejb-ql | Any character string | EJB QL | NO | |

<ejb-jar><enterprise-beans><message-driven> lower tags

| Tag name | Value | Meaning | Editing | Correspondence to Interstage Management Console Screen | | | | | | | |
|---|--|--|------------------|--|-----|--------------------------|---|-------------------------|-----|--|--|
| ejb-name? | Any character string | Enterprise Bean name | NO | EJB Application Information | | | | | | | |
| ejb-class? | Any character string | Enterprise Bean name | NO | | | | | | | | |
| transaction-type? | Choose from the following values - Bean - Container | Transaction Type | YES | | | | | | | | |
| message-selector? | Any character string | Message Selector | YES | | | | | | | | |
| message-driven-destination? | | | | | | | | | | | |
| <table border="1"> <tr> <td>destination-type</td> <td>Choose from the following values - javax.jms.Topic - javax.jms.Queue</td> <td>Destination Type</td> <td>YES</td> </tr> <tr> <td>subscription-durability?</td> <td>Choose from the following values - Durable - NonDurable</td> <td>Subscription Durability</td> <td>YES</td> </tr> </table> | destination-type | Choose from the following values - javax.jms.Topic - javax.jms.Queue | Destination Type | | YES | subscription-durability? | Choose from the following values - Durable - NonDurable | Subscription Durability | YES | | |
| destination-type | Choose from the following values - javax.jms.Topic - javax.jms.Queue | Destination Type | YES | | | | | | | | |
| subscription-durability? | Choose from the following values - Durable - NonDurable | Subscription Durability | YES | | | | | | | | |
| env-entry+ | Refer to env-entry of session tag. | | | | | | | | | | |
| ejb-ref+ | Refer to ejb-ref of session tag. | | | | | | | | | | |
| ejb-local-ref+ | Refer to ejb-local-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 | | | | | | | | | | |

<ejb-jar><relationships> lower tags

| Tag name | Value | Meaning | Editing | Correspondence to Interstage Management Console Screen | | | | | | | | | | | | |
|---|----------------------|----------------------|-------------|--|----|--------------------|----------------------|-------------------|----|-----------------------|--|--|--|--|--|--|
| description? | Any character string | Explanation | NO | CMR Mapping Definition | | | | | | | | | | | | |
| ejb-relation+ | | | | | | | | | | | | | | | | |
| <table border="1"> <tr> <td>description?</td> <td>Any character string</td> <td>Explanation</td> <td>NO</td> </tr> <tr> <td>ejb-relation-name?</td> <td>Any character string</td> <td>ejb relation name</td> <td>NO</td> </tr> <tr> <td>ejb-relationship-role</td> <td></td> <td></td> <td></td> </tr> </table> | description? | Any character string | Explanation | | NO | ejb-relation-name? | Any character string | ejb relation name | NO | ejb-relationship-role | | | | | | |
| description? | Any character string | Explanation | NO | | | | | | | | | | | | | |
| ejb-relation-name? | Any character string | ejb relation name | NO | | | | | | | | | | | | | |
| ejb-relationship-role | | | | | | | | | | | | | | | | |

| Tag name | | Value | Meaning | Editing | Correspondence to Interstage Management Console Screen | |
|----------|-----------------------------|---|----------------------------|----------------------|--|----|
| | description? | Any character string | Explanation | NO | | |
| | ejb-relationship-role-name? | Any character string | ejb relationship role name | NO | | |
| | multiplicity | Choose from the following values - One - Many | multiplicity | NO | | |
| | cascade-delete? | None | Record deletion | NO | | |
| | relationship-role-source | | | | | |
| | description? | Any character string | Explanation | NO | | |
| | | ejb-name | Any character string | Enterprise Bean name | | NO |
| | cmr-field? | | | | | |
| | description? | Any character string | Explanation | NO | | |
| | | cmr-field-name | Any character string | cmr-field-name | | NO |
| | | cmr-field-type? | Any character string | Type | | NO |
| | ejb-relationship-role | The above-mentioned reference | | | | |

<ejb-jar><assembly-descriptor> lower tags

| Tag name | | Value | Meaning | Editing | Correspondence to Interstage Management Console Screen |
|--------------------|----------------------|---|-------------------------|---------|--|
| security-role+ | | | | | Security Role |
| | description? | Any character string | Explanation | NO | |
| | role-name | Any character string | Security role name | NO | |
| method-permission+ | | | | | 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 | YES | |
| | unchecked | - | method permission check | YES | |
| | method+ | | | | |
| description? | Any character string | Explanation | NO | | |
| | ejb-name | Any character string | Enterprise Bean name | NO | |

| Tag name | | Value | Meaning | Editing | Correspondence to Interstage Management Console Screen |
|-----------------------------|-----------------|---|-----------------------|---------|--|
| | method-intf? | Choose from the following values - Home - Remote | Interface name | NO | |
| | method-name | Any character string | Method name | NO | |
| | method-params? | | Parameter | | |
| | method-param+ | Any character string | Method parameter list | NO | |
| container-transaction+ (*1) | | | | | Transaction |
| | method+ | | | | |
| | description? | Any character string | Explanation | NO | |
| | ejb-name | Any character string | Enterprise Bean name | NO | |
| | method-intf? | Choose from the following values - Home - Remote | Interface name | NO | |
| | method-name | Any character string | Method name | NO | |
| | method-params? | | Parameter | | |
| | method-params? | Any character string | Method parameter list | NO | |
| | trans-attribute | Choose from the following values - NotSupported - Required - Supports - RequiresNew - Mandatory - Never | Transaction Attribute | YES | |

*1 This can be specified only when <transaction-type> is "Container".

fujitsu-bean-definition tag structure and contents

| Tag name | Value | Meaning | Editing | Correspondence to Interstage Management Console Screen |
|---|--|---|---------|--|
| fujitsu-bean-definition? | | | | |
| description? | Any character string | Memo | YES | Interstage Extended Information |
| version-entry? (*1) | | | | |
| deploy-ejb-version? | The following value fixation - 1.1 - 2.0 - 2.1 (*8) | Based on EJB specification version | NO | |
| deploy-java-version? | Choose from the following values - 1.1 - 1.2 - 1.3 - 1.4 | JDK version used for deployment | NO | |
| base? | | | | |
| jndi-name? (*2) | Any character string | Enterprise Bean JNDI name | NO | EJB Application name |
| tran-timeout? | Choose from the following values - 0 to 2147483647 Default value is 0. | Transaction timeout | YES | - |
| Windows32/64 Solaris32 Linux32/64 tran-kind? (*1) | Choose from the following values - Local: default - Global | Distributed Transaction | YES | |
| local-mode? | Choose from the following values - False: default - True | Use of local invocation | YES | Interstage Extended Information |
| redirect? (*1) | | | | - |
| redirect-mode? | Choose from the following values - False: default - True | Standard output/ Standard error output mode | YES | |
| redirect-path? | Any character string | Standard output/ Standard error output mode | YES | |
| trace? (*1) | | | | - |
| trace-mode? | Choose from the following values - None - Buffer | Trace mode | NO | |

| Tag name | | Value | Meaning | Editing | Correspondence to Interstage Management Console Screen |
|------------------|--|---|--|---------|--|
| | | - File | | | |
| | trace-level? | Choose from the following values - 0 to 2 | Trace level | NO | |
| | trace-buffer? | 0 or more | Trace buffer | NO | |
| | trace-path? | Any character string | Trace path | NO | |
| impl? | | | | | EJB Application information |
| | impl-home-intfrep? | Any character string | Home interface repository ID | NO | |
| | impl-remote-intfrep? | Any character string | Remote interface repository ID | NO | |
| session-eb? (*2) | | | | | - |
| | session-timeout? (*3) | Choose from the following values - 0 to 2147483647 Default value is 0. | Session timeout value | YES | |
| | session-idle-timeout? (*3) | Choose from the following values - 0 to 2147483647 Default value is 1800. | No-communication monitoring time | YES | |
| | max-ejboject? (*3) | Choose from the following values - 1 to 2147483647 Default value is 1024. | Number of Stateful Beans that are connected concurrently | YES | |
| | initial-stateless-instance-count? (*4) | Choose from the following values - 0 to 2147483647 Default value is 0. | Number of Stateless Bean instances that start by default | YES | |
| entity-eb? (*5) | | | | | Interstage Extended Information |
| | entity-timeout? | Choose from the following values - 1 to 2147483647 Default value is 120. | EJB object time out value of Entity Bean | YES | |
| | entity-instance-type? | Choose from the following values - ReadWrite: default - ReadYESonly - Sequential | Entity Bean instance management mode | YES | |
| | entity-instance-size? | Choose from the following values - 1 to 2147483647 Default value is 100. | Number of Entity Bean instances | YES | |

| Tag 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: default | Entity Bean instance generation mode | YES | |
| | entity-batch-operations? (*7) | Choose from the following values - True: default - False | Mass Update of Multiple Records | YES | |
| | message-driven-eb? (*6) | | | | Message-driven Bean Extended Information |
| | message-type? | Choose from the following values - JMS - resourceadapter | Message type | YES | |
| | jms? | | | | |
| | max-messages? (*1) | Choose from the following values - 1 to 10000000 Default value is 1. | Max message | NO | |
| | subscription-name? | Any character string Note: This value cannot be omitted (a null char string cannot be specified). | Subscriber identifier | YES | |
| | connection-factory-name? | Any character string Note: This value cannot be omitted (a null char string cannot be specified). | Connection Factory name | YES | |
| | bean-pool-size? | Choose from the following values - 1 to 10000000 Default value is 8. | Number of initial start instances | YES | |
| | destination-name? | Any character string Note: This value cannot be omitted (a null char string cannot be specified). | Destination name | YES | |
| | retry-count? | Choose from the following values - 0 to 50 Default value is 0. The value can only be edited in the following cases when import is performed: - When "Container" is specified as the transaction- | Retry count | YES | Message backup definition for when an error occurs |

| Tag name | | Value | Meaning | Editing | Correspondence to Interstage Management Console Screen |
|----------|---------------------------------|--|--|---------|--|
| | | <p>type of the message-driven tag</p> <ul style="list-style-type: none"> - When "Required" is specified as the trans-attribute of the container-transaction tag | | | |
| | backup-connection-factory-name? | <p>Any character string</p> <p>The value can only be edited in the following cases when import is performed:</p> <ul style="list-style-type: none"> - When "Container" is specified as the transaction-type of the message-driven tag - When "Required" is specified as the trans-attribute of the container-transaction tag | JMS Connection Factory name for use against error loop | YES | |
| | backup-destination-name? | <p>Any character string</p> <p>The value can only be edited in the following cases when import is performed:</p> <ul style="list-style-type: none"> - When "Container" is specified as the transaction-type of the message-driven tag - When "Required" is specified as the trans-attribute of the container-transaction tag | Destination name for use against error loop | YES | |
| | resourceadapter? | | | | Message-driven Bean Extended Information |
| | resourceadapter-name? | <p>Any character string</p> <p>Note: This value cannot be omitted (a null char string cannot be specified).</p> | Resource adapter name | YES | |
| | bean-pool-size? | <p>Choose from the following values</p> <ul style="list-style-type: none"> - 1 to 1000000 <p>Default value is 8.</p> | Pool size | YES | |
| | runas-entry? | | | | Security Identity |
| | userid | Any character string | User ID | YES | |
| | password | Any character string | Password | YES | |

*1: This tag is invalid in this version.

*2: This is only valid when it is specified in Session Bean.

*3: This is only valid when it is specified in STATEFUL Session Bean.

*4: This is only valid when it is specified in STATELESS Session Bean.

*5: This is only valid when it is specified in Entity Bean.

*6: This is only valid when it is specified in Message-driven Bean.

*7: This is only valid when it is specified in CMP1.1.

*8: Definitions under the ejb-jar tag cannot be imported or exported in EJB2.1-compliant EJB applications. Edit the ejb-jar.xml file in the development environment before deployment.

fujitsu-cmp-definition tag structure and contents

| Tag name | Value | Meaning | Editing | Correspondence to Interstage Management Console Screen |
|------------------------------|---|--|---------|--|
| fujitsu-cmp-definition? (*1) | | | | |
| datasource-name? | Any character string | Datasource name | YES | CMF Mapping Definition |
| schema-name? | Any character string | Schema name | YES | |
| table-name? | Any character string | Table name | YES | |
| select-for-update? | Choose from the following values - False: default - True | Addition of FYESR UPDATE clause to findByPrimaryKey method | YES | finder Method Definition |
| stream-data? | Choose from the following values - False: default - True | stream transmission of CMP data | YES | CMF Mapping Definition |
| field-map+ | | | | |
| field-map-entry+ | | | | |
| 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 | NO | |
| field-type? | Any character string | Type | NO | |
| dbcolumn-name | Any character string | DB Column name | YES | |
| finder-map+ | | | | |
| finder-map-entry+ | | | | |
| 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 | NO | finder Method Definition |
| finder-query-string | Any character string | Query | YES | |

*1 This is specified in exclusion to fujitsu-cmp2x-mapping-definition. It is only valid when specified in CMP1.1.

fujitsu-cmp2x-mapping-definition tag structure and contents

| Tag name | | Value | Meaning | Editing | Correspondence to Interstage Management Console Screen |
|--|----------------------|---|---|---------|--|
| fujitsu-cmp2x-mapping-definition? (*1) | | | | | |
| | datasource-name? | Any character string | Data source name | YES | CMF Mapping Definition |
| | Ejb? | | | | |
| | ejb-name | Any character string | Enterprise Bean name | NO | |
| | schema-name | Any character string Note: This value cannot be omitted (a null char string cannot be specified). | schema-name | YES | |
| | table-name | Any character string | table-name | YES | |
| fujitsu-cmp2x-mapping-definition? | | | | | |
| | datasource-name? | Any character string | Data source name | YES | CMF Mapping Definition |
| | ejb? | | | | |
| | ejb-name | Any character string | Enterprise Bean name | NO | |
| | schema-name | Any character string Note: This value cannot be omitted (a null char string cannot be specified). | schema-name | YES | |
| | table-name | Any character string | table-name | YES | |
| | optimize-finders? | Choose from the following values - False: default - True | Increasing the speed of multi-item searches | YES | |
| | concurrency-mode? | Choose from the following values - for-update - normal: default | Guaranteeing integrity of data updated | YES | |
| | stream-data? | Choose from the following values - False: default - True | stream transmission of CMP data | YES | |
| | field-map2x | | | | |
| | field-map-entry2x+ | | | | |
| default-dbcolumn-name | Any character string | Internal field name | NO | | |

| Tag name | | Value | Meaning | Editing | Correspondence to Interstage Management Console Screen | |
|-------------------------|--|--|----------------|---------|--|--|
| | is-primary-key | Choose from the following values - True - False | Primary key | NO | | |
| | foreign-key? | | | | CMR Mapping Definition | |
| | foreign-ejb-name | Any character string | External key | NO | | |
| | field-name | Any character string | CMP field name | NO | CMF Mapping Definition | |
| | dbcolum-name | Any character string | DB column name | YES | | |
| Join-object+ | | | | | CMR Mapping Definition | |
| join-name | Any character string | Internal Class name | NO | | | |
| schema-name | Any character string | schema-name | YES | | | |
| table-name | Any character string | table-name | YES | | | |
| optimize-relationships? | Choose from the following values - False: default - True | Improvement in the speed of get access of relationship | YES | | | |
| source | | | | | | |
| ejb-name | Any character string | Enterprise Bean name | NO | | | |
| cmr-field-name? | Any character string | CMR field name | NO | | | |
| sink | | | | | | |
| ejb-name | Any character string | Enterprise Bean name | NO | | | |
| cmr-field-name? | Any character string | CMR field name | NO | | | |
| field-map2x | Refer to field-map2x of ejb tag. | | | | | |

*1 This is specified in exclusion to fujitsu-cmp-definition. It is only valid when specified in CMP2.0.

13.4 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>
      <entity>
        <ejb-name>EjbCmp11</ejb-name>
        <home>com.fujitsu.interstage.j2eesamples.ejb.cmp11.EjbCmp11Home</home>
```

```

    <remote>com.fujitsu.interstage.j2eesamples.ejb.cmp11.EjbCmp11Remote</remote>
    <ejb-class>com.fujitsu.interstage.j2eesamples.ejb.cmp11.EjbCmp11</ejb-class>
    <persistence-type>Container</persistence-type>
    <prim-key-class>com.fujitsu.interstage.j2eesamples.ejb.cmp11.EjbCmp11PrimaryKey</prim-key-
class>
    <reentrant>False</reentrant>
    <cmp-version>1.x</cmp-version>
  </entity>
</enterprise-beans>
<assembly-descriptor>
  <container-transaction>
    <method>
      <ejb-name>EjbCmp11</ejb-name>
      <method-name>*</method-name>
    </method>
    <trans-attribute>Required</trans-attribute>
  </container-transaction>
</assembly-descriptor>
</ejb-jar>
<fujitsu-bean-definition>
  <base>
    <jndi-name>EjbCmp11</jndi-name>
    <tran-timeout>0</tran-timeout>
    <local-mode>False</local-mode>
  </base>
  <impl>
    <impl-home-intfref>RMI:com.fujitsu.interstage.j2eesamples.ejb.cmp11.EjbCmp11Home:
0000000000000000</impl-home-intfref>
    <impl-remote-intfref>RMI:com.fujitsu.interstage.j2eesamples.ejb.cmp11.EjbCmp11Remote:
0000000000000000</impl-remote-intfref>
  </impl>
  <session-eb>
    <session-timeout>0</session-timeout>
    <session-idle-timeout>1800</session-idle-timeout>
    <max-ejboject>1024</max-ejboject>
    <initial-stateless-instance-count>0</initial-stateless-instance-count>
  </session-eb>
  <entity-eb>
    <entity-timeout>120</entity-timeout>
    <entity-instance-type>ReadWrite</entity-instance-type>
    <entity-instance-size>100</entity-instance-size>
    <entity-instance-create-type>As Required</entity-instance-create-type>
    <entity-batch-operations>True</entity-batch-operations>
  </entity-eb>
</fujitsu-bean-definition>
<fujitsu-cmp-definition>
  <datasource-name>EjbCmp11</datasource-name>
  <schema-name>EJB</schema-name>
  <table-name>EMPLOYEE</table-name>
  <select-for-update>False</select-for-update>
  <field-map>
    <field-map-entry>
      <field-name>ID</field-name>
      <field-type>int</field-type>
      <dbcolumn-name>ID</dbcolumn-name>
    </field-map-entry>
    <field-map-entry>
      <field-name>NAME</field-name>
      <field-type>String</field-type>
      <dbcolumn-name>NAME</dbcolumn-name>
    </field-map-entry>
    <field-map-entry>
      <field-name>ADDRESS</field-name>

```

```
<field-type>String</field-type>
  <dbcolumn-name>ADDRESS</dbcolumn-name>
</field-map-entry>
<field-map-entry>
  <field-name>TELEPHONE</field-name>
  <field-type>String</field-type>
  <dbcolumn-name>TELEPHONE</dbcolumn-name>
</field-map-entry>
</field-map>
<finder-map>
  <finder-map-entry>
    <finder-key-name>findALL()</finder-key-name>
    <finder-query-string></finder-query-string>
  </finder-map-entry>
</finder-map>
</fujitsu-cmp-definition>
</ejbdef>
```

Part 4 Web Service Edition

| | |
|--|-----|
| Chapter 14 Interstage Web Service Functions..... | 322 |
| Chapter 15 Developing Web Services..... | 326 |
| Chapter 16 Interstage Web Service Operation..... | 356 |

Chapter 14 Interstage Web Service Functions

Interstage Web Services are provided in the following products:

- Interstage Application Server Enterprise Edition
- Interstage Application Server Standard-J Edition

This chapter explains Interstage Web service functions.

Web services are designed to enable server programs (Web service applications) created by any method to be accessed over networks using Internet-standard technology.

They employ the SOAP communication protocol and use WSDL to encode the interface information required to use (that is, to access) server programs.

SOAP

The SOAP (Simple Object Access Protocol) protocol is an XML-based protocol for linking programs that has been prescribed by the W3C (World Wide Web Consortium). The SOAP protocol is platform independent and has excellent flexibility and extensibility.

WSDL

WSDL (Web Services Description Language) is an XML-based markup language for defining interface information for Web services that has been prescribed by the W3C (World Wide Web Consortium). WSDL also refers to interface information that has been marked up using this language. WSDL records information for each Web service, such as address information for accessing the service and the SOAP data formats for specific accesses.



Point

WSDL information can be made up of multiple files using an import function. For the Interstage Application Server, all of these files, including imported files, are referred to as a single WSDL unless noted otherwise.

14.1 Web Service Standards

The standard rules relating to Web service development and operation are shown below.

Web Services for J2EE 1.1

Web Services for J2EE 1.1 prescribes creation methods and deployment processing for J2EE Web service applications, such as the APIs, deployment descriptors and module components used by Web service applications and Web service clients.

This standard allows Web service applications to be developed using standard methods, in the same way as Web applications and EJB applications, and also allows Web service applications to be deployed to any product based on the same specification.

The implementation of Web service applications is referred to as the 'Web service end point'.

JAX-RPC 1.1 and SAAJ 1.2 are used for APIs. These various Web service application implementations and the address (URL) definitions for accessing them are referred to as 'ports'. A Web service is made up of one or more ports.

Web service clients also run as J2EE application clients.



Note

Interstage Application Server supports the following functions:

- Web service end points in the Servlet container or EJB container are provided as Web services using standard methods.
- Applications and J2EE application clients in Servlet containers or EJB containers become Web service clients and can use Web services.

SOAP Messages with Attachments

These rules set out the message formats used to send and receive attachments using SOAP. The MIME specifications are used as the mechanism for attachments.

JAX-RPC 1.1

JAX-RPC 1.1 is a standard Java API for implementing Web service applications and Web service clients. From Version 1.1, JAX-RPC also supports WS-I Basic Profile 1.0 (WS-I BP 1.0).

Refer to the following JAX-RPC Web site for more information about JAX-RPC 1.1:

- <http://jcp.org/en/jsr/summary?id=101>



Enumeration is not supported in Interstage Application Server.

SAAJ 1.2

SAAJ 1.2 is a standard Java API for referencing and updating the content of the SOAP messages that are sent and received by Web services at the XML level. SOAP attachments files can also be referenced and updated. From Version 1.2 the XML section can be used as a DOM Level2.

Refer to the following SAAJ 1.2 Web site for more information about JAX-RPC 1.1:

- <http://java.sun.com/webservices/saaj/docs.html>

Refer to the "Low-level Processing of SOAP Messages" appendix for more information about how to use the SAAJ APIs.

WS-I Basic Profile 1.0

WS-I Basic Profile 1.0 is a standard that prescribes additional clarifications and restrictions for the SOAP, WSDL and UDDI (Universal Description, Discovery and Integration) standard Web service specifications and other specifications that are used at the same time (such as HTML), from the perspective of improving the interconnectivity of Web services, as follows:

- Clarifies ambiguities in the specification
- Prescribes functions that are allowed by the specification but should not be used (i.e. it limits the scope of what should be used)

Refer to "[15.5 Developing Applications Based on the WS-I Basic Profile and Attachments Profile](#)" for more information.



The scope of WS-I Basic Profile 1.0 for Interstage Application Server does not include UDDI.

WS-I Attachments Profile 1.0

Based on the SOAP Messages with Attachments, these rules clarify the definition of SOAP specifications and the range of use for SOAP and WSDL, in order to improve inter-connectivity of Web services where attachments are used.

When the Web Service conforms to the Attachments Profile, inter-operability improves between systems that send and receive SOAP messages with attachments. For the Interstage Application Server, implementing the "[development that conforms with WS-I Basic Profile and Attachments Profile](#)" when developing a Web service that uses attachments means that the Web service can also conform to the Attachments Profile.

14.2 Interstage Web Service Basic Functions

Interstage Web services provide the following functions:

Web Service Server Function (the Server that Provides the Web Service)

Deploying Web service applications based on JAX-RPC to IJServer makes it possible to create Web services that can be called using SOAP. A WSDL is also generated to provide interface information for the Web service being provided.

Web Service Client Function (for Using the Web Service)

This function makes it possible to develop and operate Web service client applications that use SOAP to access (call) the Web service and that are based on JAX-RPC.

Web Service Development Command (iswsgen)

This command automatically generates development assets (Service interfaces, implementation classes and definition files) for Web service applications and Web service clients from service end point interfaces and WSDL files.

The ability to generate files and so on automatically allows users to develop Web services more efficiently.

14.3 Execution Environment for Web Services

The Web service execution environment is as follows:

- Web services can be run by deploying Web service applications to IJServer.
- Web service clients can be run in execution environments for IJServer and J2EE application clients.

Refer to "[2.3 Deploying and Setting J2EE Applications](#)" for more information about deploying Web service applications.

14.3.1 Execution Environment for Web Service Applications

The execution environment for the server side providing the Web service is as follows.

For IJServer

- Web service applications can be deployed and undeployed using commands or the Interstage Management Console.
- Web service applications can be operated either by using commands or by using the Interstage Management Console.
 - Monitoring for Web service applications (only Interstage Management Console)
 - Activation and deactivation for Web service applications
- Web services based on the WS-I Basic Profile 1.0.
- Web services based on the WS-I Attachments Profile 1.0



- Only the IJServer of the type that "operates the Web application and the EJB application on the same JavaVM" can be used.
- Only XML parsers that support JAXP 1.2 or later can be used.

14.3.2 Execution Environment for Web Service Clients

The execution environment for the client side that calls Web services is as follows:

- IJServer environment
- J2EE application client execution environment

Refer to "[16.2 How to Operate Web Services \(the Client Function\)](#)" for more information about the environment settings.



- Cannot operate in an environment running Java version 1.3 or earlier

- Only XML parsers that support JAXP 1.2 or later can be used.



Chapter 15 Developing Web Services

This chapter explains how Web service applications and Web service client applications are developed.

Developing Web Service Applications and Web Service Client Applications

Communications between Web services and Web service clients are conducted based on interface information defined in WSDL files.

WSDL files are created as part of the development of Web service applications. These WSDL files are used to generate stubs, and Web service client applications are developed as applications that use these stubs.

Web service applications and Web service client applications can be developed independently via WSDL files.

Using Tools to Generate Files

During the development of Web service applications and Web service client applications, files such as WSDL files and stub files are generated using the *iswsgen* command.

Refer to "iswsgen" in the Reference Manual (Command Edition) for more information about the *iswsgen* command.

The following subcommands of the *iswsgen* command are available to be used in development for different purposes:

- To develop a Web service application - *wsdl* subcommand:

Executed by specifying the *wsdl* subcommand in the *iswsgen* command

Generates the WSDL files and other files required for developing Web service applications from the service point interface.

- To develop a Web service client application - *client* subcommand:

Executed by specifying the *client* subcommand in the *iswsgen* command

Generates the files required for developing Web service client applications from WSDL interface information files.

Interstage Studio is a component-oriented integrated development support tool developed by Fujitsu. Using Interstage Studio makes it possible to perform these sequences of operations using integrated, easy-to-use views.

Refer to the Interstage Studio manual for details.

Web Services Based on the WS-I Basic Profile

Web services that are based on WS-I Basic Profile 1.0 and Attachments Profile 1.0 can be developed and operated using this product. Refer to "15.5 Developing Applications Based on the WS-I Basic Profile and Attachments Profile" for information about developing Web services based on this standard.



Note

- In cases where large amounts of data (excluding the "attachment file") is communicated, where large amounts of memory is required when it's received, processing time may exceed limits or there may be insufficient memory to process the communication.
- As a guide, if the data size exceeds 100kb (including XML tags etc), consider using the "attachment file".
- However, memory consumption levels depend on the format and content of the data, therefore can vary greatly. Also, the amount of high volume processing that can be done simultaneously as well as performance requirements can vary depending on the system being used. Therefore we recommend that the size guide above be used and the content and format of the actual business data be verified beforehand.

15.1 Developing Web Services (Server Function)

This section explains the development procedures and operations for developing Web services (the server function).

To provide a Web service, develop a WAR file for the Web service application or an ejb-jar file that contains STATELESS Session Bean and deploy it to the execution environment.

As part of Web service application development, develop the following resources and package them as a WAR file or an ejb-jar file:

- Service endpoint interface and Web service endpoint

These are Java application resources based on JAX-RPC. The service endpoint interface is the Java interface that defines the interface provided as a Web service. The Web service endpoint is the Java class that implements the actual processing logic for the Web service.

In STATELESS Session Bean, the Web service endpoint is an Enterprise Bean class.

- WSDL file

The WSDL file is the file where the interface information for calling the Web service via SOAP communications is defined.

- Deployment descriptor

The deployment descriptor is a file that specifies various definitions for the Web service or Web service application.

Web Service Development Flow

1. Create the Java interface (service endpoint interface) that defines the interface provided as a Web service and generate a WSDL file. Also create the Java class (Web service endpoint) that implements the actual processing logic for the Web service.
2. Editing the deployment descriptors
Edit the deployment descriptor.
3. Packaging the files into a WAR file or ejb-jar file/EAR file
Package the necessary files using the jar command.
4. Deploy the Web service
Deploy the files to J2SE.
5. Debug the application
Repeat the above steps when the application has been modified.

If Interstage Studio is used, all of these steps can be performed with Interstage Studio.

15.1.1 WAR/ejb-jar File Configuration for Web Service Applications

WAR Files

- Store Java resources for the Web service application under 'WEB-INF'.
- Store webservices.xml and <WSDL file name>_mapping.xml under 'WEB-INF'.
- Store the WSDL file, and the schema and other files imported by the WSDL file under 'WEB-INF/wsdl'.
- Refer to "Configuring the Web Application Directory" in the "Web Application Development" chapter for information about files other than those above.
- The files under 'WEB-INF/wsdl' are included in the public WSDL. Refer to "[Obtaining the Public WSDL](#)" for information about the public WSDL.

ejb-jar Files

- Store webservices.xml, and <WSDL file name>_mapping.xml under 'META-INF'.
- Store the WSDL file, and the schema and other files imported by the WSDL file under 'META-INF/wsdl'.
- Refer to the "EJB Application Development" chapter for information about files other than those above.
- The files under 'META-INF/wsdl' are included in the public WSDL. Refer to "[Obtaining the Public WSDL](#)" for information about the public WSDL.

15.1.2 Developing Web Service Applications

Develop Web service applications using the following procedure:

Define the Web Service Interface

(1) Define the service endpoint interface

The service endpoint interface is the Java interface that defines the public interface for the Web service application provided as a Web service.

Create a Java interface that meets the following conditions as the interface for the Web service application.

Creating this Java interface defines what kind of methods will be provided as the Web service.

- The interface should inherit the `java.rmi.Remote` interface
- Methods should throw `java.rmi.RemoteException` exceptions.
- Only the Java data types shown in "[15.3 Mapping of XML and Java Data Types](#)" should be used for method parameters (arguments and return values).
- The interface should not overload methods (should not define multiple methods with the same name but different data types or arguments).
- The method parameters (arguments and return values) cannot include `EJBObject` and `EJBLocalObject`.
- The following will not be included in array type data used in method parameters (argument and return values), structure type and Bean type members.
 - `EJBObject`, `EJBLocalObject`
 - EJB application Local interface, Remote interface, Local Home interface, Remote Home interface, Timer interface, TimerHandle interface
 - The recovery value collected by the CMP finder method

To create a STATELESS Session Bean that is called from both the Web service and RMIoverIIOP, create both the Remote interface/Home interface and the service endpoint interface.

For a STATELESS Session Bean that is called from only the Web service, it is also possible to prepare only the service endpoint interface (and not the Remote interface).



Example

```
package com.example;
public interface StockQuoteProvider extends java.rmi.Remote {
    float getLastTradePrice (String tickerSymbol) throws java.rmi.RemoteException;
}
```

(2) Generate WSDL files

Generate a WSDL file using the `iswsgen wsdl` command. For 'interface name', specify the class name of the service endpoint interface created in '(1) Define the service endpoint interface'.

For details on the `iswsgen` command, refer to the Reference Manual (Command Edition).

```
iswsgen wsdl [Option] "Interface Name"
```

After this command is executed, the following files will be generated.

- WSDL file

Interface information for the Web service is defined in this file. However, by default a provisional value is set for the connection destination URL information. A WSDL file with the correct URL information can be obtained after the Web service application has been deployed to IJServer.

- <WSDL file name>_mapping.xml

Include this file with the application at the final packaging stage. (This file specifies the associations between the application and the XML used by the SOAP protocol.)

Example

WAR files

```
iswsgen wsdl com.example.StockQuoteProvider
```

ejb-jar files

```
iswsgen wsdl -module ejb com.example.StockQuoteProvider
```

Implement the Web Service End Point

The Web service end point is a Java class that implements the process logic of the Web service applications.

In STATELESS Session Bean, the Web service endpoint is an Enterprise Bean class.

The process logic of the Web service applications generates a Java class that satisfies the following conditions:

- The class is 'public' but is neither 'final' nor 'abstract'.
- It has the 'public' default constructor.
- The 'finalize()' method is not defined.

- WAR files

It implements the service end point interface created.

- ejb-jar files

Business methods that satisfy the following conditions for each service endpoint interface are defined in the Enterprise Bean class.

- Same name
- Same number/same type argument and return values
- Throws the same exception as the service endpoint interface

When a Web service is called, a method corresponding to the call is called for the object of this Java class.

If the Initialization or Postprocessing Requires Customization

In WAR files, initialization for startup/postprocessing for closure can be defined by implementing the interface in the Web service endpoint.

```
package javax.xml.rpc.server;
public interface ServiceLifecycle {
    public void init(Object context) throws javax.xml.rpc.ServiceException;
    public void destroy();
}
```

The JAX-RPC Service Endpoint calls the 'init' method when it is instantiated, and the 'destroy' method when the instance is destroyed.

The following ServletEndpointContext object is passed to the parameter of the 'init' method.

```
package javax.xml.rpc.server;
public interface ServletEndpointContext {
    public java.security.Principal getUserPrincipal();
    public javax.xml.rpc.handler.MessageContext getMessageContext();
    public javax.servlet.http.HttpSession getHttpSession();
    public javax.servlet.ServletContext getServletContext();
}
```

Point

When the service endpoint interface and Web service endpoint are compiled, set the following for the classpath and Java VM option:

classpath:

Windows32/64

```
C:\Interstage\J2EE\lib\isws.jar
```

Solaris32/64 Linux32/64

```
/opt/FJSVj2ee/lib/isws.jar
```

Java VM option:

Windows32/64

```
-J-XX:EndorsedClassPath=C:\Interstage\J2EE\lib\isws-saa-j-api.jar
```

Solaris32/64 Linux32/64

```
-J-XX:EndorsedClassPath=/opt/FJSVj2ee/lib/isws-saa-j-api.jar
```

To use attached files or execute initialization and post-processing customization, in addition to the above, add the following to the classpath:

Windows32/64

```
C:\Interstage\J2EE\lib\isj2ee.jar
```

Solaris32/64 Linux32/64

```
/opt/FJSVj2ee/lib/isj2ee.jar
```

15.1.3 Editing the Deployment Descriptors

Mark up the following deployment descriptors.

- webservicess.xml
- WAR files
 - web.xml
- ejb-jar files
 - ejb-jar.xml

Refer to "[15.6.1 Format for webservicess.xml](#)" and "[15.6.2 webservicess.xml Tag](#)" for information about webservicess.xml.

Refer to "[15.6.3 web.xml](#)" for information about web.xml. In ejb-jar.xml, define the service endpoint interface name that was created in the deployment descriptor file as shown below.



Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar version="2.1" xmlns=http://java.sun.com/xml/ns/j2ee
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/ejb-jar_2_1.xsd">

  <display-name>BankJAR</display-name>
  <enterprise-beans>
    <session>
      <ejb-name>CreditCardEndpointBean</ejb-name>
      <home>com.fujitsu.j2ee.bank.creditcardservice.StatelessHome</home>
      <remote>com.fujitsu.j2ee.bank.creditcardservice.StatelessRemote</remote>
      <service-endpoint>
        com.fujitsu.j2ee.bank.creditcardservice.CreditCardIntf
```

```

    </service-endpoint>
    <ejb-class>
        com.fujitsu.j2ee.bank.creditcardservice.CreditCardEndpointBean
    </ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Container</transaction-type>
</session>
</enterprise-beans>

<assembly-descriptor>
    <container-transaction>
        <method>
            <ejb-name>CreditCardEndpointBean</ejb-name>
            <method-intf>ServiceEndpoint</method-intf>
            <method-name>validateCreditCard</method-name>
        </method>
        <trans-attribute>Required</trans-attribute>
    </container-transaction>
</assembly-descriptor>
</ejb-jar>

```

15.1.4 Packaging the Files into a WAR File or ejb-jar File/EAR File

Package a WAR file or ejb-jar file/EAR file using the path configuration that was specified in webservicess.xml in '15.1.3 Editing the Deployment Descriptors'.

Refer to "Configuring the Web Application Directory" in the "Web Application Development" chapter for details on how to configure the WAR files.

For details about the ejb-jar file structure, refer to the "EJB Application Development" chapter. The ejb-jar files for the Web service must be packaged in EAR file format. If a lone ejb-jar file for the Web service is deployed, an error occurs when the file is deployed.



Example

```
jar cvf StockService.war .
```



Note

In normal cases, store Web service application classes, service endpoint interface classes, and classes (in the case of a Bean/structure, the member class is included) used in arguments/return values in the following:

- WAR files
 - Store the class in WEB-INF/classes or store them as Jar files in WEB-INF/lib.
- ejb-jar files
 - Store the classes in the ejb-jar file directly.

If these classes are loaded using a class loader other than 'Webapp' (for WAR files) or 'Application' (for ejb-jar files), do not use the HotDeploy function for the Web service application. For details about the IJServer class loader configuration, refer to "Structure of a Class Loader" in the "Design of J2EE Application" chapter.

15.1.5 Settings Relating to HTTP Connections

Using Session Management

In WAR files, session management makes continuous processing possible because it enables the results of previous processing to be inherited when processing multiple requests from the same Web service client application.

Session management uses the `javax.servlet.http.HttpSession` object, which is a Servlet API. `HttpSession` objects can be obtained using the `getSession` method of the `ServletContext` object.

Refer to "[If the Initialization or Postprocessing Requires Customization](#)" for information about how to get the `ServletContext` object.

Session Timeout Settings

The default timeout period for session management is 30 minutes. To change the timeout period, edit the Web application file (`web.xml`).

Using HTTP Basic Authentication

HTTP basic authentication settings can be made for calls to the Web service. To make these settings, edit the Web application file (`web.xml`).



Note

- Session management is enabled if the Web service client supports session management using HTTP cookies.
- The WSDL information does not specify whether the Web service performs session management using HTTP cookies. Use any desired method to make arrangements with the Web service client about whether to use session management using HTTP cookies.
- The WSDL information does not make specifications regarding HTTP basic authentication for the Web service. Use any desired method to make arrangements with the Web service client about HTTP basic authentication.

15.1.6 Providing Interface Information for the Web Service

Deploying the Web service application to IJServer makes it possible to obtain the public WSDL for the Web service.

If necessary, use any desired method to provide the WSDL to the Web service users.

Refer to "[Obtaining the Public WSDL](#)" for information about obtaining public WSDLs.

15.2 Developing Applications that Call Web Services (Client Function)

To call Web services, generate a stub that has a function for calling a Web service from the WSDL for the Web service being used (called). Then develop and execute an application that calls this stub's methods.

Client Application Development Flow

The flow for developing client applications is different for the following applications:

- Applications (such as servlets) that run on IJServer
- Other applications

The procedures for each type of application are as follows:

Common Procedures

1. Obtaining interface information for the Web service
Obtain the WSDL for the Web service to be used.
2. Generating a stub
Use the `iswgen` client command to generate the files required for development from the WSDL interface information.
3. Developing the Web service client application
Use the JAX-RPC APIs to implement the client application.

Applications that run on IJServer

1. Packaging the files into a WAR file or `ejb-jar` file/`EAR` file
Package the files that were generated as part of the common procedure.

2. Deploying the files

Deploy the files to IJServer.

3. Debugging the application

Repeat the above six steps when the application has been modified.

Other applications

1. Debugging the application

15.2.1 Obtaining Interface Information for the Web Service

Obtain the WSDL for the Web service to be used.

The WSDL can be obtained in any way that is appropriate for the Web service.

Refer to "[15.1.6 Providing Interface Information for the Web Service](#)" if it is developed with the Web service as a set.

15.2.2 Generating a Stub

Generate Files

Use the *iswsgen* client command to generate the required files from the WSDL interface information. Refer to "*iswsgen*" in the Reference Manual (Command Edition) for more information on the *iswsgen* command.

```
iswsgen client [option] "WSDL file"
```

After this command is executed, the following files will be generated.

Service end point interface source

This is the Java interface source that defines the Web service application public interface provided as a Web service.

It inherits the `java.rmi.Remote` interface.

The WSDL `PortType` element name is used as the interface name.

Service interface source

This is the Java interface source that indicates the Web service described in the WSDL file. It can obtain the Web service application stub.

It inherits the `javax.xml.rpc.Service` interface.

The WSDL `Service` element name is used as the interface name.

User definition type class source

This generates the Java class source when the user definition type is specified as the WSDL operation parameter.

The user definition type name is used as the class name.

Stub/service implementation

This is the source of the classes that implement that service endpoint interface and the `Service` interface. All class names start with "`_isws_`".

Compile the Java Source Files that have been Generated

Compile all the Java source files that have been generated. Set the following jar files in the class path and Java VM option when compiling the source files that have been generated.

class path:

`Windows32/64`

```
C:\Interstage\J2EE\lib\isws.jar
```

`Solaris32/64 Linux32/64`

```
/opt/FJSVj2ee/lib/isws.jar
```

Java VM option:

Windows32/64

```
-J-XX:EndorsedClassPath=C:\Interstage\J2EE\lib\isws-saa-j-api.jar
```

Solaris32/64 Linux32/64

```
-J-XX:EndorsedClassPath=/opt/FJSVj2ee/lib/isws-saa-j-api.jar
```

To call a Web service that uses the attachment files from the client application, in addition to the above, add the following to the class path:

Windows32/64

```
C:\Interstage\J2EE\lib\isj2ee.jar
```

Solaris32/64 Linux32/64

```
/opt/FJSVj2ee/lib/isj2ee.jar
```



Note

The service end point interface or the user-defined-type class generated by the *iswsgen client* command may differ from those of the Web service applications.

To develop client applications, those generated by the *iswsgen client* command must be used.

15.2.3 Developing the Web Service Client Application

Web service clients are any classes that perform processing that calls Web services using a stub generated from a WSDL.

There are two methods for the Web service client to call the Web service. These are as follows:

- Obtaining the Service object using the JAX-RPC ServiceFactory class
- Lookup of the Service object using JNDI

Processing for calling the Web service uses the following objects:

- The `javax.xml.rpc.ServiceFactory` provided by JAX-RPC (when the ServiceFactory class is used)

This object generates the Service object.

- The `javax.naming.InitialContext` object (when JNDI is used)

Obtains the Service object using the JNDI service provider provided in Interstage.

- The Service interface and Service object generated from the WDSL

These classes correspond to the entire Web service being called. The Service object implements the Service interface and provides a function for generating a stub object. The Service object accesses the Web service using a package name that is based on the 'targetNamespace' definition in the WSDL and a class name that is based on the 'name' definition under the 'service' definition in the WSDL.

- These classes correspond to the Web service application that is being called. The stub object implements the service endpoint interface and provides functions for calling the Web service. The stub object access the Web service using a package name that is based on the 'targetNamespace' definition in the WSDL and a class name that is based on the 'name' definition under the 'portType' definition in the WSDL.

Obtaining the Service Object Using the JAX-RPC ServiceFactory Class

The method to obtain the Service object using the ServiceFactory class can be used in any Java application. In this case, call the Web service according to the following procedure:

1. Generate the ServiceFactory object.
2. Get the Service object from the ServiceFactory object
3. Get the stub object from the Service object
4. Call the stub object's methods.

Example

In the following processing example, the Service interface class name is "StockQuoteProviderService", the service endpoint interface class name is "StockQuoteProvider", and the name of the operation for the Web service that is called is "getLastTradePrice".

```
ServiceFactory sf = ServiceFactory.newInstance(); // (1)
StockQuoteProviderService sqs =
    (StockQuoteProviderService) sf.loadService(StockQuoteProviderService.class); // (2)
StockQuoteProvider sqp = sqs.getStockQuoteProviderPort(); // (3)
float price = sqp.getLastTradePrice(tickerID); // (4)
```

Lookup of the Service Object Using JNDI

When the Web service is called from a Web or EJB application, or a J2EE application client, it is possible to look up the Service object using JNDI. To obtain the Service object in other Java applications, use the ServiceFactory class.

If JNDI is used, call the Web service according to the following procedure:

1. Generate the InitialContext object.
2. Obtain the Service object using the InitialContext lookup method.
3. Obtain the stub object from the Service object.
4. Call the stub method.

Specify the following char string for the lookup argument:

```
java:comp/env/[Value specified in <service-ref-name> of the deployment descriptor]
```

For details about the deployment descriptor, refer to "Service Reference Description" in the "Developing Web Services" chapter.

Example

```
InitialContext ic = new InitialContext (); // (1)
StockQuoteProviderService sqs =
    (StockQuoteProviderService) ic.lookup("java:comp/env/service/StockQuote"); // (2)
StockQuoteProvider sqp = sqs.getStockQuoteProviderPort(); // (3)
float price = sqp.getLastTradePrice(tickerID); // (4)
```

Note

- With the JAX-RPC ServiceFactory object, use the following method:

- loadService(java.lang.Class class1)

The following methods cannot be used:

- createService(QName serviceName)

- createService(java.net.URL wsdlDocumentLocation, QName serviceName)
- loadService(java.net.URL url, java.lang.Class class1, java.util.Properties properties)
- loadService(java.net.URL url, QName qname, java.util.Properties properties)
- With the Service object, use the methods specific to the target Web service that has been generated according to the WSDL and in response to the Web service being called. The generic methods defined in the javax.xml.rpc.Service interface provided by the JAX-RPC API cannot be used.
- If session management is being used, get a new Service object every time a new session is started for a particular connection destination. Refer to "[15.2.5 Settings Relating to HTTP Connections](#)" in the "Developing Web Services" chapter for information about using session management.
- A lot of internal processing is performed when Service objects are obtained. It is recommended that Service objects that have already been obtained be retained and reused rather than obtaining a new Service object for each request, in order to improve the performance of the application.
- A single stub object cannot be used simultaneously over multiple threads. Stub objects must either be used exclusively with each thread, or a new stub object must be obtained for each request.
- If the Service object is looked up using JNDI, a single InitialContext object cannot be used in more than one thread.
- In case of communication errors and other problems, it is recommended that all exceptions that may occur during the processing be caught, and all information and stack traces should be output to logs.
- If an error occurs during HTTP communication with the Web service, or before communication with the Web service is established, a fault is returned from the Web service but the exception message may contain strings such as "ISWSFault", "faultCode:", or "faultString:".

15.2.4 Editing the Deployment Descriptor

To obtain Service objects using JNDI, the deployment descriptor must be edited.

For details about editing the Web service client application deployment descriptor, refer to "[15.6.4 Service Reference Description](#)" in the "Developing Web Services" chapter.

This task is unnecessary for obtaining Service objects using ServiceFactory.

15.2.5 Settings Relating to HTTP Connections

This section explains how to set up additional HTTP connection information for Web service client applications.

How to Specify the Connection Destination URL

Stub objects make remote calls to the default connection destination URL defined by the WSDL. To specify a connection destination URL explicitly, specify the connection destination URL as a property for the stub object.

To make a specification to the stub object using its properties, use the `_setProperty` method of the `javax.xml.rpc.Stub` interface to set the connection destination URL as a property.

```
import javax.xml.rpc.Stub;

.....

StockQuoteProviderService abf =
(StockQuoteProviderService) sf.loadService(StockQuoteProviderService.class);
    StockQuoteProvider quoteProvider = abf.getStockQuoteProviderPort();
    ((Stub)quoteProvider)._setProperty("javax.xml.rpc.service.endpoint.address",
        "http://anotherhost/StockService/services/StockQuoteProvider");
        //specify=the URL-to-connect-to

float quote = quoteProvider.getLastTradePrice(tickerID); // connect-to-URL-specified
```

The property relating to the connection destination URL is as follows:

| KEY | Value (java.lang.String) |
|--|---|
| javax.xml.rpc.service.endpoint.address | URL at connection destination(The default value is URL specified with WSDL) |

How to Set User Names and Passwords for Web Services

If the Web service is performing HTTP basic authentication, specify the authentication information as a property for the stub object.

The properties relating to the user names and passwords for Web services are as follows:

| KEY | Value (java.lang.String) |
|--------------------------------------|--------------------------|
| javax.xml.rpc.security.auth.username | User Name |
| javax.xml.rpc.security.auth.password | Password |

How to Specify the Response Timeout

To specify the timeout period after which idle connections will be disconnected, specify the timeout period (in milliseconds) as a property for the stub object.

The property relating to the timeout period is as follows:

| KEY | Value (java.lang. Integer) |
|--|---|
| Com.fujitsu.interstage.isws.client.connect.timeout | TimeOut(The default value is10 minutes) |

How to Specify the Use of Session Management

To specify whether session management using HTTP cookies be used, use a java.lang.Boolean object to specify the use or disuse of session management as a property for the stub object.

The property relating to the use of session management is as follows:

| KEY | Value (java.lang. Boolean) |
|--------------------------------|---|
| javax.xml.rpc.session.maintain | Boolean.TRUE(Use) Boolean.FALSE(Not Use=default) |

Note

- Session management takes effect if the Web service being called performs session management using HTTP cookies.
- Cookies containing session management information are not saved when Web service client applications terminate.
- If processing continues after a session has timed out, a new session will be established.

Point

The properties of the stub object can be set in an external file, as well as in the program.

Proxy Settings

For details, refer to "Connection via Proxy Servers" in the "Interstage Web Service Operation" chapter.

15.3 Mapping of XML and Java Data Types

This section explains the Java data types used by Web service applications and Web service client applications, and the corresponding XML data types.

Note

The XML namespace prefixes used in this section are given below. Namespace prefixes not used in this section can be used to execute applications and commands.

- The namespace expressed by 'xsd' is 'http://www.w3.org/2001/XMLSchema'.
- The namespace expressed by 'soapenc' is 'http://schemas.xmlsoap.org/soap/encoding/'.
- The namespace expressed by 'wsdl' is 'http://schemas.xmlsoap.org/wsdl/'.
- The namespace expressed by 'mime' is 'http://schemas.xmlsoap.org/wsdl/mime/'.
- The namespace expressed by 'wsibp' is 'http://ws-i.org/profiles/basic/1.1/xsd'.
- The namespace expressed by 'apachesoap' is 'http://xml.apache.org/xml-soap'.

15.3.1 Simple Types

The following table shows the data types that can be used as parameters (arguments or return values) of Web service applications and the corresponding types used with XML.

Primitive Types

| Java data types used as parameters for Web service applications | Data types used with XML |
|---|--------------------------|
| Int | xsd:int |
| Short | xsd:short |
| long | xsd:long |
| byte | xsd:byte |
| float | xsd:float |
| double | xsd:double |
| boolean | xsd:boolean |

Reference Types

For simple reference types, the data types used with XML for some Java types are different depending on the style protocol.

| Java data types used as parameters for Web service applications | Data types used with XML | |
|---|--------------------------|-----------------|
| | Default(Use literal) | Use encoded |
| java.lang.Integer | xsd:int | soapenc:int |
| java.lang.Short | xsd:short | soapenc:short |
| java.lang.Long | xsd:long | soapenc:long |
| java.lang.Byte | xsd:byte | soapenc:byte |
| java.lang.Float | xsd:float | soapenc:float |
| java.lang.Double | xsd:double | soapenc:double |
| java.lang.Boolean | xsd:boolean | soapenc:boolean |

| Java data types used as parameters for Web service applications | Data types used with XML | |
|---|--------------------------|-----------------|
| | Default(Use literal) | Use encoded |
| java.lang.String | xsd:string | soapenc:string |
| java.math.BigDecimal | xsd:decimal | soapenc:decimal |
| java.math.BigInteger | xsd:integer | soapenc:integer |
| java.util.Calendar | xsd:dateTime | xsd:dateTime |
| javax.xml.namespace.QName | xsd:QName | xsd:QName |
| java.net.URI | xsd:anyURI | xsd:anyURI |
| byte[] | xsd:base64Binary | soapenc:base64 |

Other XML Simple Types

The following XML data types are not used during Java-based development, but can be used with WSDL and SOAP communications. If these types are used, the data correspondences are as shown in the following table.

If these data types are used, the application must ensure that the XML values in SOAP communications conform to the XML data types.

| Data types used with XML | Corresponding Java data types |
|--------------------------|-------------------------------|
| xsd:nonPositiveInteger | java.math.BigInteger |
| xsd:negativeInteger | java.math.BigInteger |
| xsd:nonNegativeInteger | java.math.BigInteger |
| xsd:unsignedLong | Long |
| xsd:unsignedInt | Int |
| xsd:unsignedShort | Short |
| xsd:unsignedByte | java.math.BigInteger |
| xsd:positiveInteger | java.math.BigInteger |
| xsd:normalizedString | java.lang.String |
| xsd:duration | java.lang.String |
| xsd:time | java.util.Calendar |
| xsd:date | java.util.Calendar |
| xsd:gYearMonth | java.lang.String |
| xsd:gYear | java.lang.String |
| xsd:gMonthDay | java.lang.String |
| xsd:gDay | java.lang.String |
| xsd:gMonth | java.lang.String |
| xsd:token | java.lang.String |
| xsd:language | java.lang.String |
| xsd:NMTOKEN | java.lang.String |
| xsd:Name | java.lang.String |
| xsd:NCName | java.lang.String |
| xsd:ID | java.lang.String |
| xsd:hexBinary | byte[] |
| soapenc:base64 | byte[] |

Using 'nillable' Elements

Defining an element with no 'nillable' attributes or with a 'nillable' attribute value of false (false/0) in the literal WSDL schema definition refers to an element that is defined to not become 'nil' (equivalent to 'null' in Java) in the actual communication.

A Java variable that corresponds to such an element and is being transmitted must not have a 'null' value.

When an element with a 'nil' value is received for the data type that cannot have a 'null' Java value (primitive type), an initial value defined in the Java language specifications may sometimes be set as the variable.

15.3.2 Structure Type/ Bean Type

This section explains structured types and Bean types.

Structured Types

Structured types are data types that hold any number of data items (called 'members') as a public instance field. Members can be defined freely so long as they are supported data types.

The following table shows structured types and the corresponding types used with XML.

| Java data types used as parameters for Web service applications | Data types used with XML |
|---|--|
| Any class defined as a structured type | Structure(defined as an xsd:complexType whose content is xsd:sequence) |

Define structured types as public classes that meet the following conditions:

- The class must hold each member as a public instance field. (Members should not be qualified by 'transient' or 'final'.)
- Public instance field data types must all be supported data types.
- The class must have a public default constructor.
- The class must not implement the java.rmi.Remote interface.



Example

Example of a class defined as a structured type

```
package com.example; //packagename
public class Person {
    public String name; //Declare each member as a public instance field
    public int age; //As above
    public Person () { //public Public default constructor (with no arguments)
        name="nobody";
        age=0;
    }
    public Person (String myname, int myage){ // (Public constructor (optional))
        name = myname;
        age = myage;
    }
}
```

The following example shows the WSDL definition for the XML structure corresponding to the Java code above.

```
<xsd:complexType name="Person">
  <xsd:sequence>
    <xsd:element name="age" type="xsd:int"/>
  </xsd:sequence>
</xsd:complexType>
```

```

    <xsd:element name="name" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>

```

Note

- As well defining members as public instance fields, members can also be defined by providing set/get methods in the same way as for Bean types. In this case, define members so that members defined by get/set methods and members defined by public instance fields do not have the same member names (property names). Define member names (property names) using completely different names, rather than simply changing the case of some letters.
- When a different structure type such as the Bean or other structure type has been inherited and the 'public' instance fields with the same name are declared for both the structure type and the super class, only the field declared for the super class is used for the WSDL generation or the SOAP communications.
- In Java classes, the order of the members is not guaranteed. For this reason, when WSDL is generated from Java, the iswsgen wsdl command retains consistency in the order of the members on WSDL by sorting them by name.

Bean Types

Bean types are data types that are similar to structured types that hold any number of data items (called members) as properties. Members can be defined freely so long as they are supported data types.

With Bean types, members are defined by providing set/get methods.

The following table shows Bean types and the corresponding types used with XML.

| Java data types used as parameters for Web service applications | Data types used with XML |
|---|--|
| Any class defined as a Bean type | Structure(defined as an xsd:complexType whose content is xsd:sequence) |

Define Bean types as public classes that meet the following conditions:

- The class holds each member as an instance property (a set/get method pair).
- The data types for the instance properties (set/get method pairs) must all be supported data types. The class must have a public default constructor.
- The class must not implement the java.rmi.Remote interface.

Example

Example of a class defined as a Bean type

```

package com.example; // packagename
public class PersonBean {
    //Declare each member as a property (set/get method pair)
    private String _name;
    // This variable is private, and so cannot be directly referenced as a member
    public void setName(String name) { _name = name; }
    public String getName() { return _name; } // As above
    private int _age;
    // This variable is private, and so cannot be directly referenced as a member
    public void setAge(int age) { _age = age; }
    public int getAge() { return _age; }
    public PersonBean() { // Public default constructor (with no arguments)
        _name="nobody";
        _age = 0;
    }
}

```

```

public PersonBean(String name, int age){ // Public constructor (optional)
    _name=name;
    _age=age;
}
}

```

The following example shows the WSDL definition for the XML structure corresponding to the Java code above.

```

<xsd:complexType name="Person">
  <xsd:sequence>
    <xsd:element name="age" type="xsd:int"/>
    <xsd:element name="name" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>

```

Note

- Members can also be defined by providing set/get methods in the same way as for Bean types. In this case, define members so that members defined by get/set methods and members defined by public instance fields do not have the same member names (property names). Define member names (property names) using completely different names, rather than simply changing the case of some letters.
- When a different structure type such as the Bean or other structure type has been inherited and the 'public' instance fields with the same name are declared for both the Bean type and the super class, only the field declared for the super class is used for the WSDL generation or the SOAP communications.
- In Java classes, the order of the properties is not guaranteed. For this reason, when WSDL is generated from Java, the iswsngen wsdl command retains consistency in the order of the properties on WSDL by sorting them by name.

Other XML Structures

The following XML data types are not used during Java-based development, but can be used with WSDL and SOAP communications.

- Types defined as 'xsd:complexType' that contain 'xsd:all'
- Structures where members are defined as 'xsd:attribute'

15.3.3 Array Type

Supported data types can be placed in arrays and used as array types.

For array types, the data type used with XML depends on the style protocol that has been selected.

| Java data types used as parameters for Web service applications | Data types used with XML | |
|---|--|--|
| | Default(Use literal) | Use encoded |
| Array | Xsd:element where maxOccurs='unbounded' has been specified | Specify an element type in the 'soapenc:arrayType' attribute with 'soapenc:Array' used as the base type. |

Example

- The following example shows the WSDL definition for an XML array corresponding to a Java 'int' type array (int[]) when 'literal' is used.

```

<xsd:element name="age" maxOccurs="unbound" type="xsd:int"/>

```

- The following example shows the WSDL definition for an XML array corresponding to a Java 'int' type array (int[]) when 'encoded' is used.

```
<xsd:complexType name="ArrayOf_xsd_int">
  <xsd:complexContent>
    <xsd:restriction base="soapenc:Array">
      <xsd:attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:int[]" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
```

Other XML Arrays

The following XML data types are not used during Java-based development, but can be used with WSDL and SOAP communications.

- xsd:element where 'maxOccurs' has been specified as 2 or more
- complexType where 'soapenc:Array' is the base type and where the content has been defined so that maxOccurs is 'unbounded' or a single 'xsd:element' for which a value of 2 or more has been specified

15.3.4 Attachment File Type

The attachment file type is a data type which is not converted to XML during actual communications. The data is sent and received without modification as an attachment to a SOAP message.

As with a common data type, the attachment file type can be used with the parameters of the Web service applications (arguments, return values) and the members of the Bean or structure type.

The following table lists the class used as the attachment file type and the type that corresponds to use with XML:

| Java data types used as parameters for Web service applications | Data types used with XML |
|---|---|
| javax.activation.DataHandler | wsibp:swaRef (Alternatively, mapped as a MIME type in the MIME binding 'part' element) |



Point

- 'wsibp:swaRef' is an XML data type with an element that references the attached files from a SOAP message.
- Actual data is sent and received as the attachment files of a SOAP message.
- Using the iswsgen wsdl command option, the data type used with XML can be changed to 'apachesoap:DataHandler'.
- In the following instances, the WSDL maps the data type directly to the MIME binding part as a MIME type, and does not use the XML data type, 'wsibp:swaRef'.
 - When a specification other than 'DOCUMENTLITERALWRAPPED' is made in the -styleuse option using the iswsgen wsdl command; AND
 - 'javax.activation.DataHandler' is declared directly in the argument or the return value (not a member of the Bean type or the structure type, or an array) of the service end point interface

'javax.activation.DataHandler' is a class provided by JAF (Java Activation Framework). It retains various information and data contents of MIME. Refer to the Java documentation for details of this class.

Other Attachment File Data Types

The following Java classes can be also used as attachment file data types. However, to use these Java classes, the original byte column data may not be retained accurately by, for example, the conversion of the byte columns and the Java objects. The WSDL generated by the service end point interface using these classes may not conform to the WS-I Attachments Profile.

| Java data types used as parameters for Web service applications | Data types used with XML |
|---|---|
| java.awt.Image | apachesoap:Image (Alternatively, mapped as the MIME type image/jpeg in the MIME binding 'part' element) |
| javax.mail.internet.MimeMultipart | apachesoap:MimeMultipart (Alternatively, mapped as the MIME type multipart/* in the MIME binding 'part' element) |
| javax.xml.transform.Source (Note) | apachesoap:Source (Alternatively, mapped as the MIME type text/xml in the MIME binding 'part' element) |

Note) Use the 'javax.xml.transform.source.StreamSource' class as the implementation class of the 'javax.xml.transform.Source' interface. Objects of other classes are not supported.

Mapping from 'mime:content' to the Java Class

If the attachment files are specified in the 'mime:content' element in the WSDL, they are mapped as follows to the Java classes:

| Type attributes of the 'mime:content' element | Java class |
|---|-----------------------------------|
| image/jpeg | java.awt.Image |
| text/plain | java.lang.String |
| multipart/* | javax.mail.internet.MimeMultipart |
| text/xml | javax.xml.transform.Source |
| application/xml | javax.xml.transform.Source |
| Other MIME types (Note) | javax.activation.DataHandler |

Note) The image/gif type is not supported.

If the 'mime:content' element is used, the type and the element attributes of the 'wsdl:part' element referenced by the 'part' attribute of the 'mime:content' element is not mapped to the service end point interface to be generated. Instead, the above Java classes will be used.

Point

- 'apachesoap:Image', 'apachesoap:MimeMultipart', and 'apachesoap:Source' are the XML data types with an element that references the attachment files from a SOAP message.
- The actual data is sent and received as the attachment files to a SOAP message.
- To use these Java classes, the use of the 'apachesoap' namespace must be specified using the iswsgen wsdl command option.
- In the following instances, the WSDL maps the data type directly to the MIME binding 'part', and the XML data types shown above are not used.
 - When a specification other than 'DOCUMENTLITERALWRAPPED' is made in the -styleuse option using the iswsgen wsdl command; and
 - the Java data type of the attachment files is declared directly in the argument or the return value (not a member of the Bean type or the structure type, or an array) of the service end point interface
- **Solaris32/64 Linux32/64**
If java.awt.Image is used in the Java application, it may be necessary to prepare an environment for operating X Window system applications or to execute the application in headless mode. For details, refer to the Java documentation.

When the 'part' element of the MIME binding is other than the above MIME types, 'javax.activation.DataHandler' is used as the Web service application parameter. However, 'image/gif' of MIME type cannot be used.

Note

To receive an attachment file larger than a certain size, a temporary file is generated internally to save memory. Temporary files may be also generated when an API is used in relation to the attachment files.

Tuning related to the generation of temporary files is performed using the Web service setting files.

Refer to "Web Service Setting File" in the "Interstage Web Service Operation" chapter for details.

15.3.5 Using Data Types as out/inout Parameters

To use supported data types as out parameters or inout parameters, use a Java Holder class.

Point

- 'out' parameters are arguments where the value is returned from the server without values being sent from clients.
- 'inout' parameters are arguments where values are returned from the server as well as being sent from clients.

Holder Class

The format of Holder classes is as follows:

| Class Name | Class body elements | Explanation |
|-------------------------------|---|---|
| 'content type name' Holder | public content type value | Public instance field The value held by the instance |
| | public 'content type name'Holder() | Public default constructor Creates an instance by setting the value field to the initial value (0 or false for primitive types and null for reference types) |
| | public 'content type name'Holder(content type initialValue) | Public constructor Creates an instance with the value field set to initialValue. |

Web service applications and Web service clients use Holder classes as the Java arguments corresponding to the out/inout parameters of the Web service. For Web service applications, the value is returned to the service client as the 'out' parameter by setting a value in the 'value' field of the Holder object received as an argument.

After Web service clients call the Web service, the value returned in the 'value' field of the Holder object used as an argument is set.

In the Web service client, when the Web service is returned from a call, a value returned as an 'out' parameter by the Web service is set in the 'value' field of the 'Holder' object that was used as an argument.

```
//WebService method
public int serverMethod(int inParam, IntHolder inoutParam, StringHolder outParam)
    throws RemoteException {
    int number = inoutParam.value; // The inout parameter references
    the input value and performs the required processing
    :
    // A value is substituted in the "value" field for the parameter
    //(this value is returned to the client when the method returns)
    inoutParam.value = 10;
    outParam.value = "abc";
    return 0;
}
```

Example

Holder class usage example (Web service client)

```
//Prepare parametersint
inParam = 123;IntHolder inoutParam = new IntHolder(987);
StringHolder outParam = new StringHolder();

//Call the Web service (if the server code is as in the example above, "result" will be set to 0)
int result = portStub.serverMethod(inParam, inoutParam, outParam);

//Use the value of the returned out/inout parameter (get the value from the "value" field of the out/
inout parameter))
int inoutResult = inoutParam.value; //(10 if the server code is as in the example above)
String outResult = outParam.value; //("abc" abc if the server code is as in the example above)
:
```

Data Types where Standard Holder Classes are Provided

For many simple data types, standard Holder classes are provided in the `javax.xml.rpc.holders` package, so use these Holder classes.

- `StringHolder`, `BooleanHolder`, `BooleanWrapperHolder`, `ByteHolder`, `ByteWrapperHolder`, `DoubleHolder`, `DoubleWrapperHolder`, `FloatHolder`, `FloatWrapperHolder`, `IntHolder`, `IntegerWrapperHolder`, `LongHolder`, `LongWrapperHolder`, `CalendarHolder`, `QNameHolder`, `ShortHolder`, `ShortWrapperHolder`, `BigDecimalHolder`, `BigIntegerHolder`, `ByteArrayHolder`

Holder classes whose name includes 'Wrapper' are wrapper Holder classes for the corresponding primitive types. (For example, `BooleanWrapperHolder` is a Holder class for the `java.lang.Boolean` class. The Boolean type Holder class is `BooleanHolder`.)

The following Holder classes can be used in the Interstage Web service:

| Java class | Holder class |
|--|---|
| <code>java.net.URI</code> | <code>Com.fujitsu.interstage.isws.apis.holders.URIHolder</code> |
| <code>javax.activation.DataHandler</code> | <code>Com.fujitsu.interstage.isws.apis.holders.DataHandlerHolder</code> |
| <code>java.awt.Image</code> | <code>Com.fujitsu.interstage.isws.apis.holders.ImageHolder</code> |
| <code>javax.mail.internet.MimeMultipart</code> | <code>Com.fujitsu.interstage.isws.apis.holders.MimeMultipartHolder</code> |
| <code>javax.xml.transform.Source</code> | <code>Com.fujitsu.interstage.isws.apis.holders.SourceHolder</code> |

Data Types where Standard Holder Classes are not Provided

If standard Holder classes are not provided (such as for arrays and structures), prepare a custom Holder class that meets the following conditions:

- The Holder class must be a public class that meets the Holder class format described in 'Holder class'.
- The Holder class must implement the `javax.xml.rpc.holders.Holder` interface.
- The Holder class must belong to a package where 'holders' has been added to the package for the data type class being used. (For example, the package for the Holder class for the `com.example.Person` data type class will be `com.example.holders.PersonHolder`.)

Example

Holder class definition example (where the content type is the 'com.example.Person' class)

```
package com.example.holders;
final public class PersonHolder implements javax.xml.rpc.holders.Holder {
    public com.example.Person value;
    public PersonHolder() {}
}
```

```
public PersonHolder(com.example.Person initial) { value = initial; }  
}
```

Using 'out' Parameters with Web Service Applications

If a Holder class is used for a parameter in the service endpoint interface, the parameter will become an 'inout' parameter in the WSDL generated by the *iswsgen* wsdl command, etc.

If it is necessary to use an 'out' parameter rather than an 'inout' parameter, modify the WSDL so that parameter for the operation is an 'out' parameter, and delete this parameter from the input message.

When modifying the WSDL, change only the relevant section by referring to the WSDL specification and other materials to ensure that the modification is made correctly.

15.4 WSDL that Can be Used

This section explains rules that the WSDL used by Interstage Web Service must comply with, in addition to the WSDL specification.



The XML namespace prefixes used in this section are as follows:

- The namespace expressed by "xsd":"http://schemas.xmlsoap.org/wsdl/"
- The namespace expressed by "soap ":" http://schemas.xmlsoap.org/wsdl/soap/"
- The namespace expressed by "mime":"http://schemas.xmlsoap.org/wsdl/mime/"
- The namespace expressed by "wsibp":" http://ws-i.org/profiles/basic/1.1/xsd"

Type Definitions

- Refer to "[15.3 Mapping of XML and Java Data Types](#)" for information about the various data types.
- Take note of the schema definitions that cause inconsistent interpretation of the XML instances (for example, a 'sequence' model with contiguous 'element' definitions of the same name appearing at inconsistent frequency). During communications, data with such definitions can also cause inconsistent interpretation at the time of reception.
- Do not include multiple 'any' definitions in one 'complexType' definition.

Message

- If the style is 'document', there must be no more than one 'wsdl:part' tag under the 'wsdl:message' tag.
- 'wsdl:part' tags, either the 'type' attribute or the 'element' attribute can be specified but not both.

Port type

- "wsdl:part" tags, either the "type" attribute or the "element" attribute can be specified but not both.
- The value that is specified for the "name" attribute of the "operation" element must be unique within a "wsdl:portType" tag.

Bindings

Only SOAP bindings and MIME bindings can be used for "wsdl:binding".

Service

Make sure that there is at least one "wsdl:service" tag in each WSDL.

WSDL Extensions

Only WSDL extension elements and attributes that have been prescribed by SOAP bindings and MIME bindings can be used.

SOAP Bindings

- The value of the "transport" attribute of the "soap:binding" tag must be "http://schemas.xmlsoap.org/soap/http".
- The "soap:header" tag cannot be used.
- The "style" specification must be consistently either "document" or "rpc" within a WSDL.
- If "style" is "document", only "literal" can be specified for "use".
- The "use" specification must be consistently either "literal" or "encoded" within a WSDL.
- The "use" specification cannot be omitted.
- If the "encodingStyle" attribute is specified, the value must be "http://schemas.xmlsoap.org/soap/encoding/".

MIME Bindings

A WSDL with the following contents is not supported:

- A 'mime:part' element that contains multiple 'mime:content' elements
- A 'wsibp:swaRef' data type that is referenced from a location other than the 'type' attribute of the 'xsd:element' element or the 'type' attribute of the 'wsdl:part' element.

Other Restrictions Windows32/64

If the WSDL has one of the following names that matches the DOS device name, it cannot be used:

- Type name
- Port type name
- Service name
- Namespace configuration element (a character string separated by '.', '/' or ':')

If this is the case, an error such as *isws15208*, *isws10443*, or *isws10314* may be generated by the *iswsgen* command or the *iswsgen client* command.

When this applies only to the namespace configuration element, the WSDL can be used by executing the *iswsgen* command as follows:

- Using the *-PkgNSmappingFile* option, specify so that the DOS device name is not included in the Java package name configuration element.

15.5 Developing Applications Based on the WS-I Basic Profile and Attachments Profile

Creating systems based on the Profile developed by WS-I is one way of guaranteeing interoperability between different platforms. Systems based on the WS-I Basic Profile 1.0 and WS-I Attachments Profile 1.0 can be constructed using this product.

Note the following points when creating systems based on the Profile.

Generating WSDL Files Using the *iswsgen wsdl* Command

For the WS-I Basic Profile 1.0 standard, the "use" attributes of WSDL files must be "literal". When you use the default value of the *iswsgen wsdl* command, a WSDL file with "literal" specified in the 'use' attribute is generated. Take note if the specification is to be changed using the *-styleuse(-y)* option. Refer to "iswsgen" in the Reference Manual (Command Edition) for information on how to set options for the *iswsgen wsdl* command.

Creating and Editing WSDL Files

When editing WSDL files that have been created by a user or generated using the `iswsgen wsdl` command, the final creator of the WSDL file must ensure that it complies with the WS-I Basic Profile 1.0 standard.

Create and edit WSDL files within the range of the WS-I Basic Profile 1.0 standard by referring to the latest version of the WS-I Basic Profile 1.0 standard and its errata.



Please confirm up-to-date and the latest errata referring to the following. .

```
http://www.ws-i.org/
```

Using Parameters at the Time of Application Execution

If the application has been generated by specifying the 'RPCLITERAL' in the `-styleuse(-y)` option at the time of the WSDL generation, or by using the WSDL files with "rpc" as the 'style' attributes and "literal" as the 'use' attributes, take note of the following:

- Do not specify and send 'null' as the Web service application parameter.
- In the same communication, for the attachment file type parameter of 'javax.activation.DataHandler' of the Web service application, do not specify the same instance as that of another location. Instead, specify separate instances.

Using Sessions

Web service applications can use sessions that use HTTP cookies. However, the WS-I Basic Profile 1.0 standard recommends that the normal behavior of Web services should not rely on HTTP cookies. It is recommended that Web services be designed so that sessions are only used to assist processing and the Web service can run normally even if sessions do not continue.

Handling XML Data Encoded with UTF-8 using BOM

The WS-I Basic Profile 1.0 recognizes the use of UTF-8 with BOM (Byte Order Mark) for the encoding of the SOAP messages used for communication. If users create SOAP envelopes using XML data encoded with UTF-8 with an attached BOM, use classes that implement the following source interfaces.

- SAXSource
- StreamSource

Depending on the parser used by the user, there may be no problem even if classes that implement source interfaces other than the ones above are used. For more information, check the specification of the parser used.

Refer to the "Low-level Processing of SOAP Messages" appendix for more information about how to create SOAP envelopes using XML data.

Developing Applications that Conform to WS-I Attachments Profile 1.0

Take note of the following to develop applications that conform to WS-I Attachments Profile 1.0

- When executing the `iswsgen wsdl` subcommand, specifying "apache" in the `-attachmentsType` option causes a non-standard XML data type to be output to the WSDL files. When developing the applications that conform to the Attachments Profile, create the applications so that the `-attachmentsType` option is omitted or "swaref" is specified as the option value to enable the Java class to be mapped to the 'wsibp:swaRef' type or the 'mime:content' element.

Refer to other descriptions in this section to create the applications that conform to the Basic Profile.

15.6 Web Service Environment Definition Files (Deployment Descriptors)

The Web services environment definition file (`webservices.xml`) sets up the operating environment for Web services.

The Web application environment definition file (`web.xml`) is also required for Web services to run.

To look up Services using JNDI in the Web service client application, the service reference description must be added to the deployment descriptor.

This section explains how to mark up these definition files.

15.6.1 Format for webservicexml

Web services.xml

"webservicexml" is the deployment descriptor for Web service applications.

This definition file specifies the paths for Web service-related components for the application, and the implementations of the Web service being provided. This file also specifies the paths within the module used when Web service application files are packaged as a WAR file.

Mark up the deployment descriptor using XML format. The format for the deployment descriptor is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<webservicexml xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
        http://java.sun.com/xml/ns/j2ee/j2ee_web_services_1_1.xsd"
    version="1.1">
  <webservice-description>
    <webservice-description-name>name</webservice-description-name>
    <wsdl-file>file-path</wsdl-file>
    <jaxrpc-mapping-file>file-path</jaxrpc-mapping-file>
    <port-component>
      <port-component-name>name</port-component-name>
      <wsdl-port>qname</wsdl-port>
      <service-endpoint-interface>interface</service-endpoint-interface>
      <service-impl-bean>
        <servlet-link>servlet-name</servlet-link>
      </service-impl-bean>
    </port-component>
  </webservice-description>
</webservicexml>
```



Notes on marking up this file

- Only those definitions listed in this manual can be used.
- The initial <?xml...> tag specifies the XML declaration. Be sure to mark up this tag at the start of the deployment descriptor file.
- If Japanese double-byte characters are used in the deployment descriptor, specify UTF-8 for the encoding format (the "encoding=" section) in the <?xml...> tag. If Japanese double-byte characters are used, the Web service application cannot be deployed if an encoding other than UTF-8 is specified. This restriction also applies to comments.
- The <webservicexml...> and </webservicexml> tags are the root tags that indicate the start and end of the XML file. Be sure to specify the root tags.
- Follow the markup order above when marking up the file.
- Note that IJServer may start without outputting an error message even if definitions other than the ones in this manual are specified.

15.6.2 webservicexml Tag

The following tags can be specified in the Web services environment definition file (webservicexml).

Content Defined

| Tag name | Explanation | Optional? | Multiple specification? |
|-----------------------------|--|-----------|-------------------------|
| webservices | Defines the start and end of webservices.xml. | No | No |
| webservice-description | Specifies a definition for a single Web service. | No | Yes |
| webservice-description-name | Defines the Web service name. The values used as Web service names must be unique within the webservice.xml file. Note: The following characters cannot be used. ' , _ = ! , : * ; ' ? , ' " | No | No |
| wsdl-file | Specifies the path for the WSDL file in the module. | No | No |
| jaxrpc-mapping-file | Specifies the path for the <WSDL file name>_mapping.xml file in the module. | No | No |
| port-component | Specifies a definition for a single port. | No | Yes |
| port-component-name | Defines the port name. The values used as Web service names must be unique within the webservice.xml file. Note: The following characters cannot be used. ' , _ = ! , : * ; ' ? , ' " | No | No |
| wsdl-port | Defines the QName for the corresponding port in the WSDL. This must be a port defined in a WSDL file specified by the wsdl-file tag. The port 'Qname' is specified in the 'name' attribute of the 'port' element of the WSDL. The value belongs to the namespace specified in the 'targetNamespace' attribute of the definitions element of the WSDL. Note: The same port of WSDL cannot be specified with two or more wsdl-port tag. | No | No |
| service-endpoint-interface | Defines the fully qualified name for the service endpoint interface. | No | No |

| Tag name | Explanation | Optional? | Multiple specification? |
|-------------------|--|-----------|-------------------------|
| service-impl-bean | Specifies the definition for the Web service implementation class. | No | No |
| servlet-link | Defines the name of the servlet in web.xml that corresponds to the Web service. Note: The same servlet cannot be specified with two or more servlet-link tags. Either servlet-link or ejb-link (but not both) must be specified. | Yes | No |
| ejb-link | Defines the ejb-name of STATELESS Session Bean in ejb-jar.xml that corresponds to the Web service. Note: The same ejb application cannot be specified with two or more ejb-link tags. Either servlet-link or ejb-link (but not both) must be specified. | Yes | No |

Note

- The Web service application cannot be deployed if tags where the "Optional?" column is marked "No" are omitted.
- The Web service application cannot be deployed if tags where the "Multiple specification?" column is marked "No" are specified more than once.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<webservices xmlns=http://java.sun.com/xml/ns/j2ee
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/j2ee_web_services_1_1.xsd"
  version="1.1">
  <webservice-description>
    <webservice-description-name>StockQuoteProviderService
    </webservice-description-name>
    <wsdl-file>WEB-INF/wsdl/StockQuoteProviderPort.wsdl</wsdl-file>
    <jaxrpc-mapping-file>WEB-INF/StockQuoteProviderPort_mapping.xml
    </jaxrpc-mapping-file>
    <port-component>
      <port-component-name>StockQuoteProvider</port-component-name>
      <wsdl-port xmlns:pfx="http://example.com">
        pfx:StockQuoteProviderPort
      </wsdl-port>
      <service-endpoint-interface>com.example.StockQuoteProvider
      </service-endpoint-interface>
    <service-impl-bean>
```

```

        <servlet-link>StockQuoteServlet</servlet-link>
    </service-impl-bean>
</port-component>
</webservice-description>
</webservices>

```

15.6.3 web.xml

The Web application environment definition file (web.xml) is also required for Web services to run, in addition to the Web services environment definition file (webservices.xml).

web.xml format

Create web.xml using the format shown in "6.3 Web Application Environment Definition File (Deployment Descriptor)". There are no new tags to be added.

Refer to "6.3 Web Application Environment Definition File (Deployment Descriptor)" in the "Web Application Development" chapter for information about tags not shown in the following table.

| Tag name | Explanation |
|---------------|---|
| servlet-name | Defines the name of the servlet that is referred to from the servlet-link tag in webservices.xml. |
| servlet-class | Defines the fully qualified class name of the Web service implementation class. |
| url-pattern | Defines the URL that is mapped to the Web service. The default value is as follows: "/services/" + the value of the port-component-name tag in webservices.xml |



Example

```

<?xml version="1.0" encoding="UTF-8"?>
<!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>
  <servlet>
    <servlet-name>StockQuoteServlet</servlet-name>
    <servlet-class>com.example.StockQuoteProviderImpl</servlet-class>
  </servlet>
</web-app>

```

15.6.4 Service Reference Description

To look up a Service from a Web service client application using JNDI, the service reference description must be added to the following deployment descriptors, according to the client application configuration.

- Lookup of Service from Web applications: web.xml
- Lookup of Service from EJB applications: ejb-jar.xml
- Lookup of Service from J2EE application clients: application-client.xml

Add the <service-ref> tag to the target deployment descriptor and then describe the definition for the reference of the Web service.

For details about tags not shown in the table below, refer to:

- "Web Application Environment Definition File (Deployment Descriptor)" in the "Web Application Development" chapter
- "J2EE Application Client Deployment Descriptor File Detailed Set Up" in the "JNDI" chapter
- "Description in Deployment Descriptor File" in the "JNDI" chapter.

Content Defined

| Element name | Explanation | Optional? | Multiple specification? |
|---------------------|---|-----------|-------------------------|
| service-ref | Describes the definition for the reference of the Web service. | Yes | Yes |
| service-ref-name | Defines the name that is registered in JNDI. Define it so that it starts with "service/". | No | No |
| service-interface | Defines the fully-qualified class name of the Service interface. | No | No |
| wsdl-file | This element is omitted when stub and client applications are created using the iswsgen command or Interstage Studio. If this element is specified in a Web or EJB application, the stub/service implementation class used in Interstage for the application is generated when the deployment to the IJServer occurs. This element defines the WSDL file from the module root relative path. | Yes | No |
| jaxrpc-mapping-file | Defines <WSDL file name>_mapping.xml from the module root relative path. This element cannot be omitted if wsdl-file is specified. | Yes | No |

Note

- The class name that is created when <wsdl-file> is specified will be "the class name that excludes _isws package name" (i.e.:stock.server._isws_StockQuoteProviderPortSoapBindingStub). If a class with the same name exists in the module that is deployed, it is overwritten.
- "javax.xml.rpc.Service" cannot be specified in the <service-interface> element. Specify a Service interface (an interface that inherits javax.xml.rpc.Service) class name that is specific to the service that is generated from WSDL.
- <wsdl-file> cannot be specified if JDK is not installed.

Example

Lookup of Service from a Web application (web.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns=http://java.sun.com/xml/ns/j2ee
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xsi:schemaLocation=http://java.sun.com/xml/ns/j2ee
  http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd version="2.4">
  <display-name>StockQuote Client</display-name>
  ...
  <service-ref>
    <description>WSDL Service StockQuote </description>
    <service-ref-name>service/StockQuote</service-ref-name>
    <service-interface>stock.server.StockQuoteProviderService
    </service-interface>
  </service-ref>
</web-app>
```

15.7 Storage Location for the Sample Application

A sample application for a Web service is stored in the following directory.

Refer to the documentation in the same directory for information on the execution procedure.

Windows32/64

```
C:\Interstage\J2EE\sample\isws
```

Solaris32/64 Linux32/64

```
/opt/FJSVj2ee/sample/isws
```


Chapter 16 Interstage Web Service Operation

This chapter explains Interstage Web service operations.

16.1 How to Operate Web Services (the Server Function)

Web service applications are operated using either commands or the Interstage Management Console.

Creating an IJServer and Making Environment Settings

To use a Web service, create a "same VM" type IJServer.



- Only the IJServer type that runs on same VM can be used.
- Only XML parsers that support JAXP 1.2 or later can be used.

Deploying and Undeploying Applications

The WAR or EAR files that contain the Web service application can be deployed or undeployed in the same way as conventional WAR or EAR files.

Refer to "Deploying and Setting J2EE Applications" for more information about deploying and undeploying applications.

The default timeout period for read processing of WSDL files and resources referenced by WSDL files is 45 seconds.

To change the default timeout value, set the following value in the java process system property used by the Interstage JMX service.

```
-Dcom.fujitsu.interstage.isws.deploy.wsdl.timeout=length-of-timeout (unit: millisecond)
```

If "0" is specified, the timeout period will be unlimited.



To set 90 seconds as the timeout period:

```
-Dcom.fujitsu.interstage.isws.deploy.wsdl.timeout=90000
```

To obtain resources referenced by WSDL files by proxy, implement the required setting in the system property of the java process used by the Interstage JMX service.

Refer to "[16.2.3 Connection via Proxy Servers](#)" for details of the values to be set in the system property.

Web Service Public URL

In WAR files, Web service public URLs can be customized according to the Servlet features. For details, refer to "[7.1.1 Call that Requires Mapping](#)", "Calling by Specification in a URL" in the "How to Call Web Applications" chapter, and "Web Service Environment Definition Files (Deployment Descriptors)", "[15.6.3 web.xml](#)" in the "Developing Web Services" chapter.

Obtaining the Public WSDL

When a Web service application is deployed to an IJServer, the public WSDL for the Web service can be obtained.

If necessary, provide the WSDL to the users of the Web service by means of any method.

To obtain the public WSDL, select the Web module with the Interstage Management Console ("WorkUnit" -> IJServer name -> "Application Status/Undeploy" tab) and then obtain the public WSDL using the "Web Service Environment Definition" tab.

To obtain the public WSDL of a Web service application that is a STATELESS Session Bean, select the EAR module and ejb-jar module with the Interstage Management Console (select "WorkUnit" -> JServer name -> "Application Status/Undeploy" tab). Select the application using the "Web Service Environment Definition" tab.

Point

- The WSDL obtained in this way is organized in the standard format for J2EE application modules, as well as containing URL information for calling the Web service.
 - Some indentations may be smoothed out with the public WSDL.
 - The public WSDL also contains the files under "WEB-INF/wsdl" in the WAR file that was distributed, or files under "META-INF/wsdl" in the ejb-jar file.
 - The public WSDL cannot be obtained using the multi-server management batch operation function. If the multi-server management function is being used, use the integrated management function.
-

Web Service Environment Definitions

The Web service environment definition items are specified in the deployment descriptor.

Refer to "[15.6 Web Service Environment Definition Files \(Deployment Descriptors\)](#)" in the "Developing Web Services" chapter for more information about the Web service deployment descriptor.

Note

To change the settings, deploy the Web service application again. The settings cannot be changed once the Web service application has been deployed.

Monitoring Web Service Applications

The processing time for each method of a Web service application can be looked up using the Interstage Management Console.

The processing time including SOAP engine processing can also be looked up as the Servlet processing time, but this information is grouped by ports rather than methods, and therefore shows the totals and averages for all methods within each port.

Specifying the Upper Limit of the Request Message Size to be Received (Including Attachment Files and SOAP Messages)

The maximum size of a request to be received by a Web service (including attachment files and SOAP messages) can be specified.

If a request exceeding these conditions is received, the following occurs:

- The Web service application is not called.
- The following error is returned to the Web service client (depending on the request format):
 - When an HTTP error message is returned (normal)
An error message with the status code 413 is returned.
 - When a SOAP error message is returned ('Fault' message)
An error message for "Fault" is unfixed (FaultString).
The following is output to 'container.log'.
 - ISWS: ERROR: isws11201: Error occurred; nested exception is:
org.xml.sax.SAXParseException: Premature end of file. %s (detail information); or
 - Other

Note that a request message larger than 2GB (2147483647 bytes) cannot be received by a Web service. If a request message larger than 2GB is received, an HTTP error message is returned with the status code 400.

The method to limit the request size is shown below.

Setting Web Service Request Size Limits from the Interstage Management Console

Specify the upper limit for requests in the following setting items using the Interstage management console:

- [Service] > [Web server] > "Web-server-name" > [Environment settings] > [Detail settings] > [Maximum size of the request message main body]

This specification is valid not only for a Web service but for all requests received by Web servers. If any problems should occur, specify the limits using the following method.

Setting Request Size Limits by the Web service Web Application Unit

Specify the limits using the <Location> directive and the <LimitRequestBody> directive of the Interstage HTTP Server environment definition files.

After specifying the Web application path in the <Location> directive, specify the upper limit of the request size in the <LimitRequestBody> directive.



Example

Example of an entry (extracts) in the Interstage HTTP Server environment definition file to limit the request size of the Web service Web application (path: "myws")

```
<Location /myws>
  LimitRequestBody 1048576
</Location>
```

16.2 How to Operate Web Services (the Client Function)

For Web service clients, the operating method depends on the type of client application.

There are two types of client applications that run with Web services, as follows:

- Applications that run on IJServer
- J2EE application client
- Other applications

Applications that Run on IJServer

For applications that run on IJServer, package and deploy (or undeploy) the created application and stub in the same way as for servlet or EJB applications.

The following settings are also required, depending on the IJServer type:

For the "same VM" type:

Enable the Web service function for the container in the IJServer environment settings.

For other IJServer types:

Set the following jar file to the class path and Java VM option in the IJServer environment settings.

class path:

Windows32/64

```
C:\Interstage\J2EE\lib\isws.jar
```

Solaris32/64 Linux32/64

```
/opt/FJSVj2ee/lib/isws.jar
```

Java VM option:

Windows32/64

```
-XX:EndorsedClassPath=C:\Interstage\J2EE\lib\isws-saa-j-api.jar
```

Solaris32/64 Linux32/64

```
-XX:EndorsedClassPath=/opt/FJSVj2ee/lib/isws-saa-j-api.jar
```

J2EE Application Client

Refer to "JNDI Service Provider Environment Setup" and "Environment Setup in Client Environment" in the "JNDI" chapter, configure the settings that are required to use JNDI, and then execute the application.

The following must also be set in the class path and Java VM option:

class path:

- The created application, stub, etc.
- The following jar files:

Windows32/64

```
C:\Interstage\J2EE\lib\isws.jar  
C:\Interstage\J2EE\lib\isws-lib.jar  
C:\Interstage\J2EE\lib\isj2ee.jar
```

Solaris32/64 Linux32/64

```
/opt/FJSVj2ee/lib/isws.jar  
/opt/FJSVj2ee/lib/isws-lib.jar  
/opt/FJSVj2ee/lib/isj2ee.jar
```

Java VM option:

Windows32/64

```
-XX:EndorsedClassPath=C:\Interstage\J2EE\lib\isws-saa-j-api.jar
```

Solaris32/64 Linux32/64

```
-XX:EndorsedClassPath=/opt/FJSVj2ee/lib/isws-saa-j-api.jar
```

Other Applications

Set the following files in the class path and Java VM option and then execute the application.

class path:

- The created application, stub, etc.
- The following jar files:

Windows32/64

```
C:\Interstage\J2EE\lib\isws.jar  
C:\Interstage\J2EE\lib\isws-lib.jar
```

Solaris32/64 Linux32/64

```
/opt/FJSVj2ee/lib/isws.jar  
/opt/FJSVj2ee/lib/isws-lib.jar
```

Java VM option:

Windows32/64

```
-XX:EndorsedClassPath=C:\Interstage\J2EE\lib\isws-saa-j-api.jar
```

Solaris32/64 Linux32/64

```
-XX:EndorsedClassPath=/opt/FJSVj2ee/lib/isws-saa-j-api.jar
```

Note

- Cannot operate in a Java Version 1.3 or earlier environment.
- If you are running a Web service client on the IJServer, do not use the HotDeploy/auto reload functionality in the application.
- Only XML parsers that support JAXP 1.2 or later can be used.
- By setting "isws-lib.jar" above to the class path, the following jar files will also be automatically set to the class path.

Windows32/64

```
C:\Interstage\J2EE\lib\xerces\xercesImpl.jar  
C:\Interstage\J2EE\lib\xerces\xml-apis.jar  
C:\Interstage\J2EE\lib\isj2ee.jar
```

Solaris32/64 Linux32/64

```
/opt/FJSVj2ee/lib/xerces/xercesImpl.jar  
/opt/FJSVj2ee/lib/xerces/xml-apis.jar  
/opt/FJSVj2ee/lib/isj2ee.jar
```

- For the J2EE application client, set the class path so that "isj2ee.jar" is after "isws-lib.jar". For Other applications, do not set "isj2ee.jar" in the class path.

16.2.1 Client Function Logs

The Interstage Web service client function notifies error information to the user application by throwing exceptions. If exceptions cannot be thrown, it outputs the content of errors to the standard error output or to Web service client log files from outside the IJServer.

Note

Web services logs on IJServer are output to the container log. Refer to "Debugging Applications", "IJServer Log" in the "Operating J2EE Applications" chapter for information about container logs.

Web Service Client Log Files

Web service client log files are files where the Java client function for the Interstage Web service outputs error information (except for exceptions thrown) from outside the IJServer. These files are prepared separately for each process.

Note

- If Web service clients running on different processes have been set to output logs to the same file, logs may not be output correctly (file rotation may fail or output messages may become mixed up).
- If the initialization for these files fails (such as when there is no access permission to the path specified as the Web service client log file path, or when there is no such directory), error information will be output to the standard error output. In this case, no further logs will be output.

Specifying the Web Service Client Log File

Set the Web service client log file using the following item in the Web service settings file:

```
com.fujitsu.interstage.isws.log.file.path
```

Refer to "[16.3.1 Web Service Client Log File Path](#)" in "Web Service Settings File" for information about this setting.

Note

If multiple processes are using a common Web Service setting file, a common Web service client log file path will be set for multiple processes, and so errors may occur, such as failure to rotate the Web service client log file, as described in the note under "[16.3.1 Web Service Client Log File Path](#)". However, no problems will occur if a Web service client log file path is not specified, as errors will be output to the standard error output.

16.2.2 Stub Settings File

The properties of stub objects can be set using a stub settings file, as well as by using the stub's APIs.

The following section explains how to make settings using a stub settings file.

Point

- The properties that have been specified using the stub properties file are set when the stub object is obtained from the Service object. Accordingly, applications can overwrite these properties using the `_setProperty` method of the `javax.xml.rpc.Stub` interface.
- If, when obtaining the stub object from the Service objects, the URL to be connected to has been specified as the argument in the method, the value specified in that argument is given priority over the value specified in the stub setting files.

How to Specify the Stub Settings File

Specify the path to the stub settings file using the following system property for each Service interface:

System property name:

```
<Fully qualified class name (FQCN) for the Service interface>.configuration
```

Example

Settings example when the Service interface is `com.example.TestService`:

```
-Dcom.example.TestService.configuration=C:\temp\test.properties
```

Format for Stub Settings Files

Stub settings files are marked up using the property file format.

Property names and property values are marked up using the following format.

Settings for a particular port within a service

```
<Port name>.<Name of the property to be set in the stub>=<Value of the property to be set in the stub>
```

Settings for all port within a service (The settings above take precedence.)

```
default.<Name of the property to be set in the stub>=<Value of the property to be set in the stub>
```

Example

```
Test.javax.xml.rpc.service.endpoint.address=http://www.example.com/services/Test  
Test.javax.xml.rpc.session.maintain=true
```

Note

- If the property type is java.lang.Boolean, the property will be true if the value is equivalent to the string "true" (disregarding case) and false otherwise.
- If the property type is java.lang.Integer, a warning message will be output if an invalid value is specified for the integer value, and the property will not be set.
- Port names or property names that do not exist will be ignored.
- Authentication and other information for HTTP basic authentication can also be set as properties. Set access permissions to the file appropriately according to the information specified in the stub settings file.

Port Names

For the port names used in property names, use the same strings as the port name strings used in the get<port name> methods of the Service interface that is generated, rather than the port names specified in the WSDL. (Use port names that have been converted into names that can be used with Java.)

16.2.3 Connection via Proxy Servers

If Web service client applications connect to the Web service via a proxy server, set the values shown in the table below in the system properties or the String type properties in the stub object. The stub object settings have priority over the system properties. For details about the stub object settings, refer to "16.2.2 Stub Settings File" and "Developing Applications that Call Web Services (Client Function)", "15.2.5 Settings Relating to HTTP Connections" in the "Developing Web Services" chapter.

| KEY | Value to be set | Remarks |
|--------------------|--|---|
| http.proxyHost | Host Name | If no value is set, the Web service client application will connect without going through a proxy server. |
| http.proxyPort | Port Number | If no value is set, the Web service client application will connect without going through a proxy server. |
| http.nonProxyHosts | Name of the host connected without going through proxy | Multiple hosts can be specified by separating them with " ". |
| http.proxyUser | Use Name | Set this value if the proxy server performs basic authentication. |
| http.proxyPassword | Password | Set this value if the proxy server performs basic authentication. |

16.3 Web Service Settings File

The Web service settings file is the properties file where the Web service is set up. This properties file must be prepared for each process. However, if settings for concurrent processing have been made on an IJServer, then a common Web service settings file must be prepared for all of the processes on the same IJServer.

Installation Path Specification

Place the Web service settings file on the path that has been set in the following system property.

System Property Name

```
com.fujitsu.interstage.isws.configuration
```

Values that can be specified:

The path to the Web service settings file (specify either the full path or the relative path from the current directory)

If no value is specified:

The Web service settings file will not be loaded.

Example

System Properties Usage Example

Example where the Web service settings file (config.properties) is located in the current directory.

```
-Dcom.fujitsu.interstage.isws.configuration=config.properties
```

Windows32/64

Example where the Web service settings file (config.properties) is located in the C:\tmp.

```
-Dcom.fujitsu.interstage.isws.configuration=C:\tmp\config.properties
```

Solaris32/64 Linux32/64

Example where the Web service settings file (config.properties) is located in the /tmp.

```
-Dcom.fujitsu.interstage.isws.configuration=/tmp/config.properties
```

Note

If the Web service settings file does not exist at the specified path, or if this directory cannot be accessed, a warning message will be output and processing will continue using the default values for all settings items.

Settings Items

The settings items are as follows:

| No | Value to be set | Key Name | Optional? | Valid with IJServer? |
|----|--|--|-----------|----------------------|
| 1 | Web service client log file path | com.fujitsu.interstage.isws.log.file.path | Yes | No (*1) |
| 2 | Maximum size for Web service client log files | com.fujitsu.interstage.isws.log.file.maxfilesize | Yes | No (*1) |
| 3 | Maximum number of generations for Web service client log files | com.fujitsu.interstage.isws.log.file.maxbackupindex | Yes | No (*1) |
| 4 | SSL definitions used by Web service clients | com.fujitsu.interstage.isws.client.ssl.configname | Yes | Yes |
| 5 | Location of temporary files created for attachment files | com.fujitsu.interstage.isws.attachment.tmpdir | Yes | Yes |
| 6 | Maximum size for processing using memory only (not creating temporary files) when attachment files are received. | com.fujitsu.interstage.isws.attachment.memory.cachesize | Yes | Yes |
| 7 | Deletion of attachment file data received by the Web service application at the time of returning a response (releasing resources) | com.fujitsu.interstage.isws.attachment.tempfile.keepmode | Yes | Yes |
| 8 | Default character code of attachment files specified in 'text/plain' in WSDL | com.fujitsu.interstage.isws.attachment.plaintext.charset | Yes | Yes |

| No | Value to be set | Key Name | Optional? | Valid with IJServer? |
|----|---|--|-----------|----------------------|
| 9 | Non-ASCII character sending format | com.fujitsu.interstage.isws.utf8encoder.notscii.charref | Yes | Yes |
| 10 | Enabling authentication using the Directory Service | com.fujitsu.interstage.isws.enable.directory.authentication | Yes | Yes |
| 11 | Web service client connection timeout | com.fujitsu.interstage.isws.client.socket.connection.timeout | Yes | Yes |

*1) Settings relating to Web service client log files are not valid with IJServer.

Note

Enter the Web service setting files according to the specifications for the Java property files.

Example: Use the Unicode escape to enter characters, such as Japanese, that cannot be expressed directly using the ISO 8859-1 character encoding.

16.3.1 Web Service Client Log File Path

Key name

```
com.fujitsu.interstage.isws.log.file.path
```

Values that can be specified:

The path to the output file (specify either the full path or the path relative to the current directory)

If no value is specified:

Error information will be output to the standard error output.

Example

Outputting logs to "log.txt" in the current directory

```
com.fujitsu.interstage.isws.log.file.path=log.txt
```

Windows32/64

Outputting logs to "C:\tmp\log.txt"

```
com.fujitsu.interstage.isws.log.file.path=C:\\tmp\\log.txt
```

Solaris32/64 Linux32/64

Outputting logs to "/tmp/log.txt"

```
com.fujitsu.interstage.isws.log.file.path=/tmp/log.txt
```

Note

- Specify a different Web service client log file for each process. Refer to "[Web Service Client Log Files](#)" for more information.
- The directory where files are output must be prepared.
- If there are no access permissions to the path specified by this item, or if the directory does not exist, an error message will be output to the standard error output, and processing will continue. After the error message is output, logs will not be output to either the Web service client log file or the standard output.

16.3.2 Maximum Size for Web Service Client Log Files

Key name

```
com.fujitsu.interstage.isws.log.file.maxfilesize
```

Values that can be specified:

1 to 2048 (Unit: MB)

If no value is specified:

10

Example

Making the maximum size of Web service client log files 1024 MB (1 GB)

```
com.fujitsu.interstage.isws.log.file.maxfilesize=1024
```



- This item is only valid if com.fujitsu.interstage.isws.log.file has been specified.
- If an invalid value is specified, a warning message will be output to the standard error output, and the default value of 10 will be used.

16.3.3 Maximum Number of Generations for Web Service Client Log Files

Key name

```
com.fujitsu.interstage.isws.log.file.maxbackupindex
```

Values that can be specified:

1 to 100

If no value is specified:

5

Example

Setting the maximum number of generations of the Web service client log file to 10

```
com.fujitsu.interstage.isws.log.file.maxbackupindex=10
```



- This item is only valid if com.fujitsu.interstage.isws.log.file has been specified.
- If an invalid value is specified, a warning message will be output to the standard error output, and the default value of 5 will be used.

16.3.4 SSL Definitions Used by Web Service Clients

Key name

```
com.fujitsu.interstage.isws.client.ssl.configname
```

Values that can be specified:

SSL definition name defined in Interstage Certificate Environment

If no value is specified:

None

Example

Using SSL definitions with the definition name "WSClientSSL" for Web service client SSL communications

```
com.fujitsu.interstage.isws.client.ssl.configname=WSClientSSL
```

16.3.5 Location of Temporary Files Created for Attachment Files

Key name

```
com.fujitsu.interstage.isws.attachment.tmpdir
```

Explanation

To receive an attachment file larger than a certain size, a temporary file is generated internally to save memory. Temporary files may also be generated when API is used for the attachment files.

Values that can be specified:

Absolute path of the directory

(The JavaVM process must have the authority to write in the local system and sufficient disk space.)

If no value is specified:

An application that operates on IJServer: JavaVM current (system property:user.dir)

An application that operates on non-IJServer: JavaVM TEMP folder (system property: java.io.tmpdir)

When the system property value is not a valid path, receipt of attachment files fails.

Example

Windows32/64

A directory is created for each process in D:\tmp\wsdata, and temporary files are created under it.

```
com.fujitsu.interstage.isws.attachment.tmpdir=D:\\tmp\\wsdata
```

Solaris32/64 Linux32/64

A directory is created for each process in /var/tmp/wsdata, and temporary files are created under it.

```
com.fujitsu.interstage.isws.attachment.tmpdir=/var/tmp/wsdata
```

Note

- Allocate sufficient disk space according to the upper limit of the attachment file size and the amount called to the location of file creation.
- Allocation of an independent drive or partitioning of the file creation location is recommended so that other processing is not affected by insufficient disk space when a large attachment or a large number of attachments are received.
- When a Java process has abended, temporary files may remain in the created location. In this instance, delete the files manually. If the IJServer has not customized the file creation location, in the Web service application, the files are deleted automatically after a certain number of activations as long as the current directory is under work unit function control (default).
- The file creation location must be a directory that can be written to by the Java process. It must also be a directory in the local system (it cannot be a remote shared directory, for example).
- The "iswsatttmp*" directory is created under the specified directory in which temporary files will be located. Do not locate files other than temporary files under the "iswsatttmp*" directory.

16.3.6 Maximum Size for Processing Using Memory Only (not Creating Temporary Files) when Attachment Files are Received.

Key name

```
com.fujitsu.interstage.isws.attachment.memory.cachesize
```

Values that can be specified:

An integer between 0 and 2147483647 (by Kbyte unit)

0 if a negative value is specified. A default value if any other invalid value is specified

If no value is specified:

16 (Kbytes)

Example

Attachment files of up to 64K bytes in size are processed using memory only (without generating temporary files).

```
com.fujitsu.interstage.isws.attachment.memory.cachesize=64
```

16.3.7 Deletion of Attachment File Data Received by the Web Service Application at the Time of Returning a Response (Releasing Resources)

Key name

```
com.fujitsu.interstage.isws.attachment.tempfile.keepmode
```

Explanation

Specify whether or not the Web service application is to retain the attachment file data received from the Web service client after the response is returned.

Values that can be specified:

'true' or 'false' (not case sensitive)

'false' is specified when a value other than 'true' is specified.

When 'false' (or a default value) is specified, the 'DataHandler' object that the Web service application received from the Web service client and the 'DataSource' object with its data contents, may be deleted when a response is returned (releasing resources). In this instance, these objects can no longer be used.

If 'true' is selected, the objects of the attachment file data are retained after a response has been returned.

If no value is specified:

false

Example

The objects of the attachment file data are not released after a response has been returned.

```
com.fujitsu.interstage.isws.attachment.tempfile.keepmode=true
```



- If 'true' is set for this item, the temporary files are not deleted until the GC collects the objects of the attachment file data. Therefore, it is important to allocate sufficient disk space to the temporary file creation location and to not retain references to the objects of the attachment file data in the application unnecessarily.
- Data read by the 'getInputStream' method of the 'DataHandler' object before the 'DataSource' objects were released is not subject to resource release. It can be used and released arbitrarily by the application.

- In the Web service client, deletion of the 'DataSource' object (resource release) is left with the GC.

16.3.8 Default Character Code of Attachment Files Specified in 'text/plain' in WSDL

Key name

```
com.fujitsu.interstage.isws.attachment.plaintext.charset
```

Explanation

Specify which character code to use when sending attachment files with 'text/plain' in the 'type' attributes of the "mime:content" element in the WSDL or when receiving if the character code is unknown.

Values that can be specified:

US-ASCII

ISO-8859-1

UTF-16

Other character codes supported by the version of JDK in use

If no value is specified:

UTF-8

Example

The files are sent in ISO-8859-1 and interpreted as ISO-8859-1 if the character code is unknown when received.

```
com.fujitsu.interstage.isws.attachment.plaintext.charset=ISO-8859-1
```

16.3.9 Non-ASCII Character Sending Format

Key name

```
com.fujitsu.interstage.isws.utf8encoder.notascii.charref
```

Explanation

Specify whether to send non-ASCII characters in string data used for SOAP communication as they are, or whether to convert them to XML character reference format before sending them as ASCII characters.

According to the XML specification, no differences in the data occur as a result of this change on the recipient side. Usually, there is no need to change this setting.

Values that can be specified:

True or false (names are not case-sensitive).

Values other than true are treated as "false".

If no value is specified:

false

Example

Converting non-ASCII characters in string data used for communication to XML character reference format before sending them as ASCII characters:

```
com.fujitsu.interstage.isws.utf8encoder.notascii.charref=true
```

Note

- If true is specified for this setting, characters that are not Unicode basic multilingual plane characters (characters expressed according to the surrogate pair in the Java char type), for example some characters in JIS X 0213:2004 that are added from JIS X 0208, cannot be sent correctly if they are assigned.
- This setting does not affect the sending format of the string data in the attachment file data of the attachment file type.

16.3.10 Enabling Authentication Using the Directory Service

Key name

```
com.fujitsu.interstage.isws.enable.directory.authentication
```

Explanation

Specify whether the Web service client application that looks up Service using JNDI sends HTTP requests using information authenticated by the Directory Service. If true is set for this item, use the following as the Basic authentication user name/password (instead of the settings in "How to set user names and passwords for Web services" of "[Settings Relating to HTTP Connections](#)").

- J2EE application client:

The user name/password set in the JNDI environment properties and authenticated by the JNDI service provider

- Applications on the IJServer:

The user name/password authenticated by the Web application.

Values that can be specified:

True or false (names are not case-sensitive).

Values other than true are treated as "false".

If no value is specified:

false

Example

Using the user name/password authenticated by the Directory Service

```
com.fujitsu.interstage.isws.enable.directory.authentication=true
```

Note

This setting is not valid for obtaining Service objects using ServiceFactory.

16.3.11 Web Service Client Connection Timeout

Key name

```
com.fujitsu.interstage.isws.client.socket.connection.timeout
```

Explanation

Specify the Web service client connection timeout.

Values that can be specified:

An integer between -2147483648 and 2147483647 (the unit is milliseconds).

If an invalid value is specified, the default value will be used.

Note

- If the specified value exceeds the timeout set in the system, the timeout set in the system takes priority.
- If 0 is specified, the timeout set in the system will be used.
- If a negative value is specified, the value that was specified for the response timeout will be used. However, if the response timeout exceeds the timeout set in the system, the timeout set in the system takes priority.
- To change the timeout set in the system, tune the TCP/IP parameter.
- For details on the value that is specified for the response timeout, refer to "Developing Applications that Call Web Services (Client Function)", "[15.2.5 Settings Relating to HTTP Connections](#)", "How to Specify the Response Timeout" in the "Developing Web Services" chapter.

If no value is specified:

If SSL is not used, the value that was specified for the response timeout is used.

If SSL is used, the timeout set in the system is used.

Example

When the connection timeout is set as 30 seconds

```
com.fujitsu.interstage.isws.client.socket.connection.timeout=30000
```

Part 5 JTS/JTA Edition Windows32/64 Solaris32 Linux32/64

| | |
|---|-----|
| Chapter 17 Using Java Transaction API (JTA) | 372 |
|---|-----|

Chapter 17 Using Java Transaction API (JTA) Windows32/64 Solaris32

Linux32/64

JTA is a distributed transaction operation API supplied by a transaction service. The Java transaction service (JTS) is the transaction service supplied by Interstage.

17.1 JTA Windows32/64 Solaris32 Linux32/64

JTA Interfaces

The JTA specifications were proposed as a standard Java interface by the corporation formerly known as Sun (now Oracle Corporation).

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.

| 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.

17.2 User Transaction Interface Windows32/64 Solaris32 Linux32/64

The User Transaction interface is included in JTA.

17.2.1 Functions Provided by the User Transaction Interface Windows32/64 Solaris32

Linux32/64

The user transaction interface has the functions listed in Table.

| Function name | Description |
|---------------|---|
| begin | Starts a transaction, and associates the thread with the transaction. |

| Function name | Description |
|-----------------------|---|
| 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

- The setTransactionTimeout method is only valid if a distributed transaction is used. If this method is used in a local transaction, the value that was set is invalid.
- 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.

17.2.2 Environment Setup for the User Transaction Interface Windows32/64 Solaris32

Linux32/64

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

Windows32/64

```
C:\INTERSTAGE\ots\lib\fjtsclient.jar
```

Solaris32

```
/opt/FSUNots/lib/fjtsclient.jar
```

Linux32/64

```
/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.

17.2.3 Acquiring the User Transaction Interface Windows32/64 Solaris32 Linux32/64

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, refer to "JNDI Service Provider Environment Setup", "J2EE application client", in the "JNDI" chapter.

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

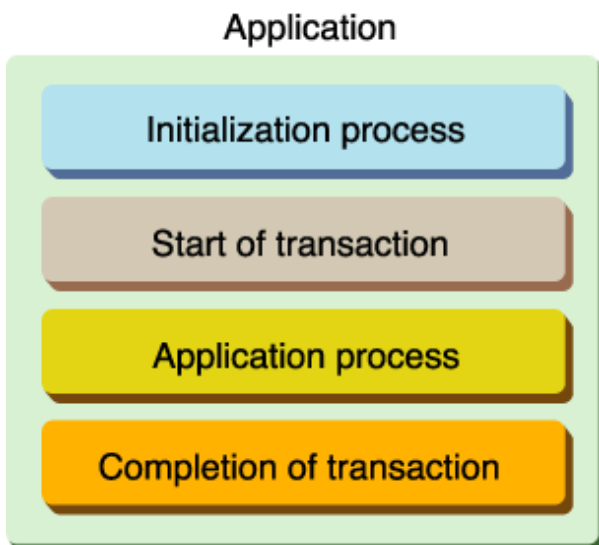
If JNDI environment properties are not correctly defined, lookup processing fails. In this case, confirm the setting of the JNDI environment properties.

17.3 Generating a JTA Application Windows32/64 Solaris32 Linux32/64

This section explains how to generate a JTA application.

17.3.1 Application Configuration Windows32/64 Solaris32 Linux32/64

Figure shows an example of a client application configuration that takes the following processes into account.



Initialization Process

Use JNDI to acquire the UserTransaction object.

Start a Transaction

Start a transaction.

Application Process Section

Describe the business logic.

Completion of Transaction

Commit or roll back a transaction.

17.3.2 Performing Initialization Process and Acquiring the UserTransaction Object Windows32/64 Solaris32 Linux32/64

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);
}
```

17.3.3 From Transaction Start to Transaction Stop Windows32/64 Solaris32 Linux32/64

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.");
    }
}
}

```

17.3.4 JTA Application Example Windows32/64 Solaris32 Linux32/64

```

/* 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.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.");
    }
}

```

1. To use JNDI, generate InitialContext.
2. Acquire the javax.transaction.UserTransaction object from JNDI.
3. Start a transaction using the javax.transaction.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.

Point

For details on JNDI, refer to the "JNDI" chapter.

17.3.5 Precautions Windows32/64 Solaris32 Linux32/64

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 the "JNDI" chapter.

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 6 JMS Edition

| | |
|---|-----|
| Chapter 18 Basic Functions of Interstage JMS..... | 380 |
| Chapter 19 Environment Settings for Interstage JMS..... | 386 |
| Chapter 20 Developing a JMS Application..... | 398 |

Chapter 18 Basic Functions of Interstage JMS

This chapter describes the basic functions of Interstage JMS.

The following lists some key terms used for Interstage JMS.

Message

A message refers to a piece of user data to be sent and received using JMS.

Publish/Subscribe messaging model

Publish/Subscribe is a 1-to-n messaging model in which the same message is distributed to multiple receivers.

Publisher

For JMS, a publisher refers to the sender of a message in the Publish/Subscriber messaging model.

Subscriber

For JMS, a subscriber refers to the receiver of a message in the Publish/Subscriber messaging model.

Durable Subscription function

The durable subscription function allows a message sent while an application is not active to be received after the application becomes active.

Durable Subscriber

A durable subscriber refers to a subscriber with the durable subscription function.

Message Listener

The message listener function automatically handles a message when it arrives at the receiver.

Point-To-Point messaging model

Point-To-Point is 1-to-1 messaging model in which the message is distributed to a specific receiver.

Sender

For JMS, a sender refers to the sender of a message in the Point-To-Point messaging model.

Receiver

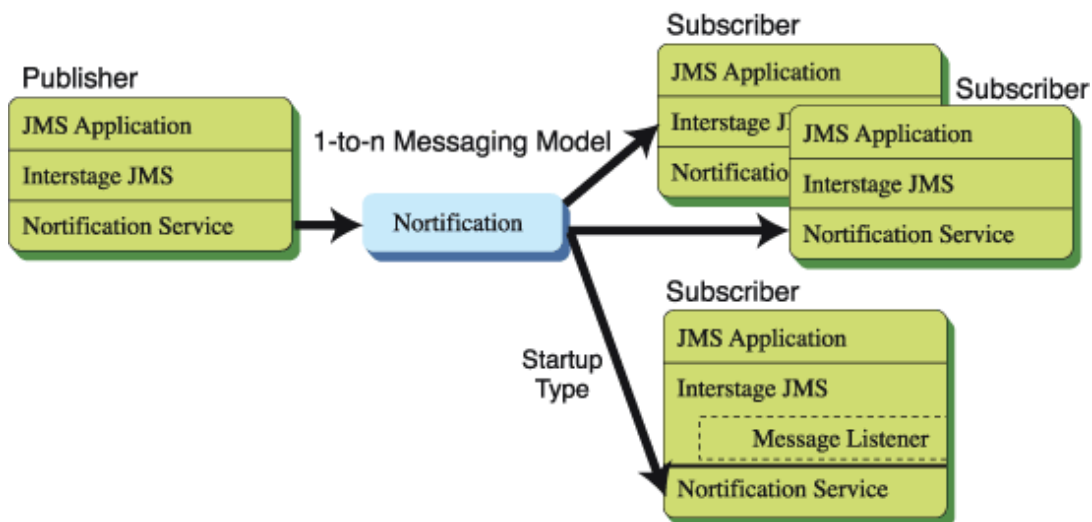
For JMS, a receiver refers to the receiver of a message in the Point-To-Point messaging model.

18.1 Publish/Subscribe Messaging Model (1-to-n Messaging Model)

Interstage JMS provides the following functions in the Publish/Subscribe messaging model.

- This model provides a model that delivers the same messages to multiple receivers.
The sending application (publisher) and the receiving application (subscriber) are linked via the Notification Service.
- As the type of receiving application, both the standby type and startup type using Message Listener are supported.
- The priority of messages and the control of lifetime are provided by the Notification Service.

The Publish/Subscribe messaging model or Point-To-Point messaging model can be specified during creation of a channel (*esmchnl*).



The following describes the advantages obtained when using the Publish/Subscribe messaging model.

- Delivering to multiple subscribers

A publisher can deliver the same message to multiple subscribers by sending it to one destination without being aware of the number of connecting subscribers.

- Changing the operation type with little effect on applications

No publisher's change is required even when the number of subscribers changes.

18.2 Point-To-Point Messaging Model (1-to-1 Messaging Model)

Interstage JMS provides the following functions in the Point-To-Point messaging model.

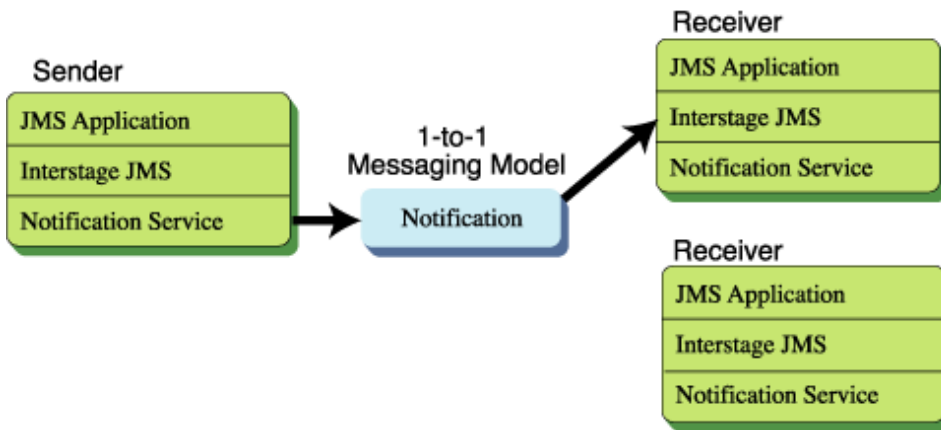
- Point-To-Point asynchronous communication

The sending application (sender) and the receiving application (receiver) are linked via the Notification Service.

Even when multiple receivers are connected, only one receiver can receive a message. The message is automatically routed.

- As the type of receiving application, both the standby type and startup type using Message Listener are supported.
- The priority of messages and the control of lifetime are provided by the Notification Service.

The Publish/Subscribe messaging model or Point-To-Point messaging model can be specified during creation of a channel (*esmkchnl*).



The following describes the advantages obtained when using the Point-To-Point messaging model.

- **Improving system throughput**

The system throughput of a business application can be improved by increasing the concurrency of a receiver with process concurrency or thread concurrency. The system throughput is improved by performing database processing concurrently after a receiver received a message.

- **Improving operability**

For the Publish/Subscribe messaging model, the messages delivered before subscribers were created with createSubscriber are discarded.

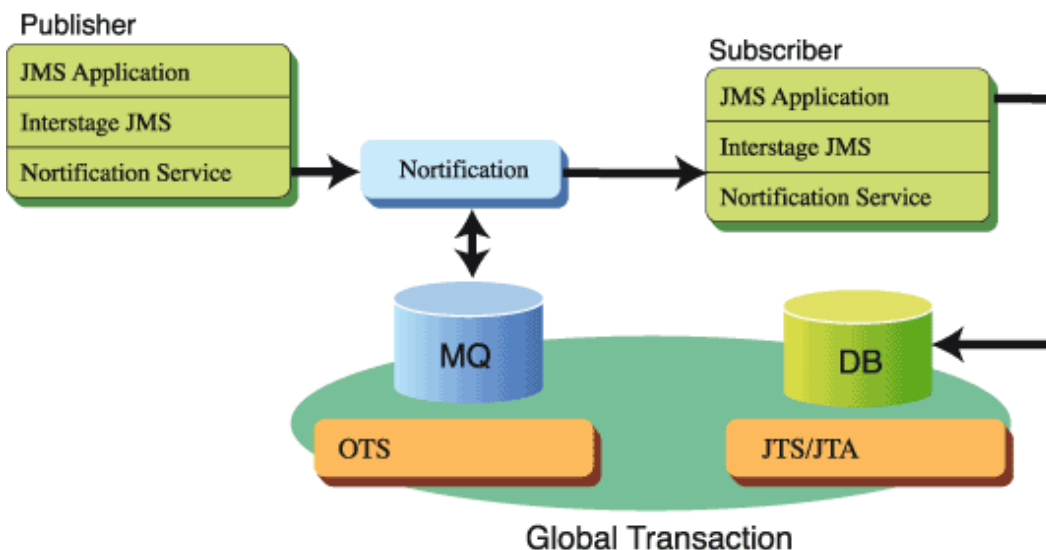
For the Point-To-Point messaging model, delivered messages are stored regardless of the status of a receiver and can be received after receivers are created with createSubscriber.

18.3 Message Guarantee

Message duplication and losses are prevented by the persistent function of the event channel and local transaction function.

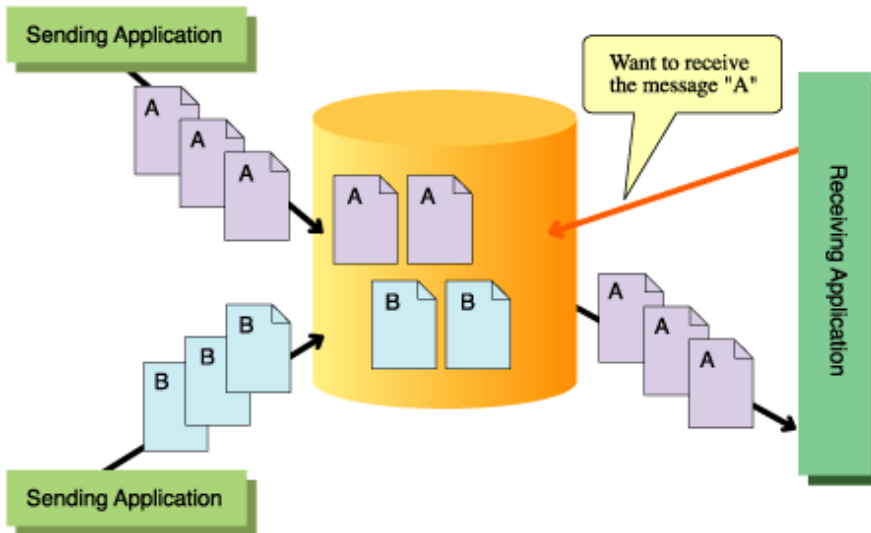
Windows32/64 Solaris32 Linux32/64

The global transaction function guarantees message sending/receiving and consistency of DB update processing.



18.4 Message Selector Function

The message selector function can be used by a receiver application to selectively receive the desired messages among the messages sent by a sender application.

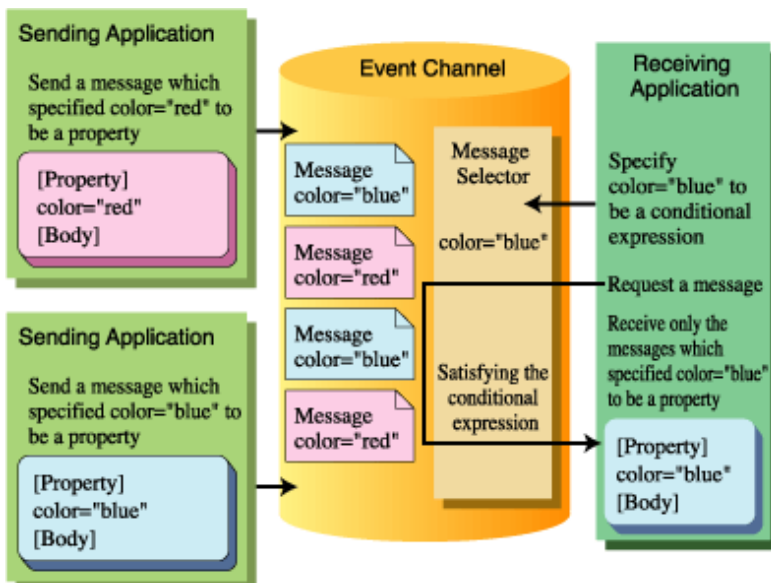


The following describes how the message selector function works.

- A sender application sets information for identifying and classifying a message in the Properties field of the message and then sends the message.
- A receiver application specifies the conditional expression for identifying desired messages at the beginning of the reception. From then on, a receiver application receives only the messages that have a property satisfying the conditional expression.

For the conditional expression, refer to '[Message Selector Conditional Expression](#)' under 'Developing a JMS Application'.

- For the Publish/Subscribe messaging model, the messages that do not satisfy the conditional expression are discarded. For the Point-To-Point messaging model, the messages that do not satisfy the conditional expression are skipped.



The following describes the advantages obtained when the message selector function is used.

- Only interested information can be obtained by a receiver application.

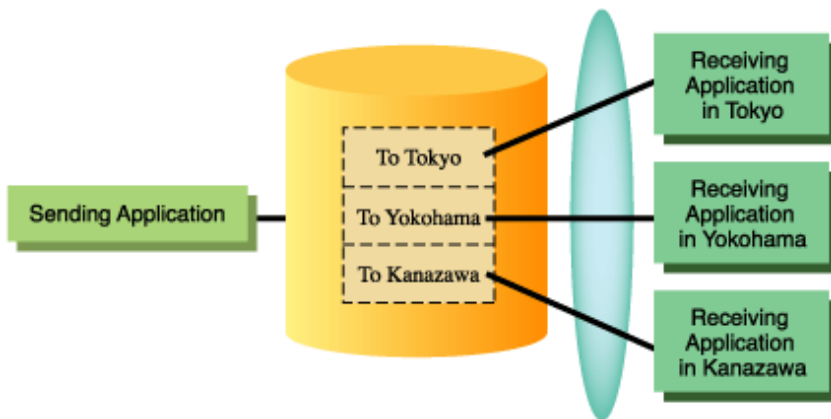
Since a receiver application can receive only the desired information instead of all messages sent by a sender application, operability is improved.

For example, when a sender application sends news about international problems, national problems, politics, economics, weather forecast, sports, and show business, a receiver application can selectively receive news about weather forecast and sports.

- The number of event channels and the amount of system resources can therefore be reduced.

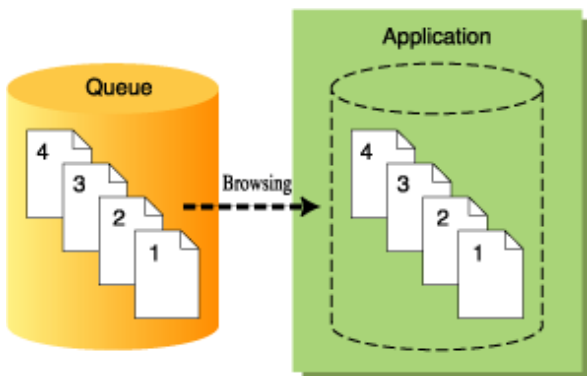
Conventionally, a channel must have been created for each site when a message is sent to specific sites. In this case, the workload of the system is high because system resources are consumed as the number of sites increases.

Since the message selector function allows a receiver application on each site to selectively receive messages, the number of channels can be reduced to 1 as shown below. This saves the system resources even when the number of sites increases.



18.5 Queue Browser Function

The queue browser function can be used by an application for browsing messages stored in the queue.

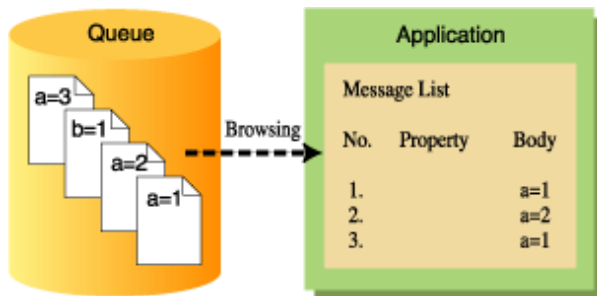


The following describes how the queue browser function works.

- An application creates a browser for browsing messages stored in the queue.

- Then, it can browse the contents of the queue by sequentially retrieving the messages.
- During browsing, retrieving a message from the queue does not delete it.

The development of the application, which has the feature displaying the contents of the queue by using queue browser, becomes easy.



Chapter 19 Environment Settings for Interstage JMS

This chapter describes the environment settings for using Interstage JMS.

Point

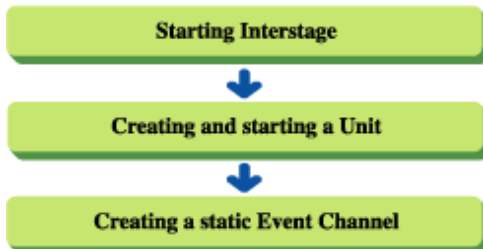
The following table shows the correspondence between terms used in the Event Service and terms used in Interstage JMS.

| Event Service | Interstage JMS |
|--------------------------------|----------------------------------|
| Event Channel | Topic or Queue |
| Statically generated channels | Topic or Queue |
| Dynamically generated channels | TemporaryTopic or TemporaryQueue |
| Consumer | Subscriber, Receiver or Consumer |
| Supplier | Publisher, Sender or Producer |
| Event Data | Message |

19.1 Environment Settings for the Event Channel Operation Machine before Operation

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 figure shows the procedure for environment setting before the operation.



19.1.1 Starting Interstage

Start the Event Service by starting Interstage with the Interstage Management Console or the *isstart* command.




Using the *isstart* command

Execute the *isstart* command to start Interstage and then start the Event Service.

```
isstart
```

19.1.2 Creating and Starting a Unit

To use the following functions, create a unit with the Interstage Management Console or the *esmkunit* command.

- Durable Subscription function
- Event channel persistent function (message assurance function)
- Local transaction function (message assurance function)
- Global transaction function (message assurance function)   

Using the *esmkunit* command

The unit can be created using the *esmkunit* command.

Example

Creating a unit using the unit definition file 'unit1.def'.

```
esmkunit -uf unit1.def
```

The prototype of a unit definition file is in the following directory (default installation path):

Edit the unit definition file settings items as necessary to create the unit.

Windows32/64

```
C:\Interstage\eswin\etc\def
```

Solaris32/64 Linux32/64

```
/opt/FJSVes/etc/def
```

Note

Set the unit definition file settings items below using values correctly estimated:

- **unitmode**
Specify "std" (standard unit) or "ext" (extended unit).
- **tranmax**
Specify the number of transactions that can be concurrently performed for Event Channels in the unit.
- **tranunitmax** **Solaris32/64 Linux32/64**
Specify the maximum message size (in blocks) that can be operated in a transaction (1 block: 16 KB (fixed)).
- **syssize**
Specify the capacity of system (for unit control) file.
- **sysqnum**
Specify the number of system (for unit control) data storage areas.
- **usersize**
Specify the capacity of event data (message) file.
- **userqnum**
Specify the number of event data (message) storage areas.
- **shmmax**
Specify the size of shared memory to be used by unit.

19.1.3 Creating a Static Event Channel

Create an event channel used for sending/receiving messages by the JMS application by using the Interstage Management Console or the *esmkchnl* command.

Note

- **Windows32/64 Solaris32 Linux32/64**
Check that the database linkage service has been started before global transaction operation.
- **Windows32/64 Solaris32 Linux32/64**
If the database linkage service is used, SSL linkage cannot be used.
- Automatic start settings are configured as the default when the Event Channel is created, so the Event Channel starts automatically when Interstage starts (the Event Service starts). The Event Channel automatic start settings can be changed using the Interstage Management Console.
- If the SSL Accelerator is used for applications on the machine using the Event Channel, and the SSL environment has not been set up on this machine, the Event Channel cannot be created using the Interstage Management Console. In this case, create the Event Channel using the *esmkchnl* command.

Using the *esmkchnl* command

The Event Channel can be created using the *esmkchnl* command.

Example

Create the event channel 'mychannel' for operation using the persistent function of the event channel and local transaction feature in the group 'mygroup':

For Publish/Subscribe messaging model:

```
esmkchnl -g mygroup -c mychannel -notify -persist all -tran
```

For Point-To-Point messaging model:

```
esmkchnl -g mygroup -c mychannel -notify -ptp -persist all -tran
```

Specify the following options and arguments as required:

- **-g group**
Specify the group name.
- **-c channel**
Specify the name of the Event Channels within the group.
- **-m number**
Specifies the total number (maximum number) of consumers and suppliers that can connect to the mixed model Event Channels contained in the group. To use Message-driven Bean, estimate the maximum number using the formula shown below.
$$\langle \text{max number} \rangle = (\langle \text{initial instance num} \rangle * 2) * 1.2 + \langle \text{client application concurrency level} \rangle$$
- **-notify**
Generates an Event Channel of the JMS. 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 following functions are used. Event data (messages) and connection information will be persistence targets.

- Durable Subscription function
- Persistent function of the event channel
- Local transaction function
- Global transaction function **Windows32/64** **Solaris32** **Linux32/64**

- -tran

Specify -tran when the global transaction function is not used.

- **-ots** **Windows32/64** **Solaris32** **Linux32/64**

Specify -ots when the global transaction function is used.

- -autodiscon

Specify whether remaining connection information in the EventChannel is automatically collected when the Producer/Consumer disconnects, without issuing the Connection or Session 'close' method for a static generation EventChannel when the application ends abnormally or IJServer is shut down, for example.

Local transactions still incomplete are rolled back.

Note: Do not use this option with the Durable Subscription function.

- -ssl

Specify this to use SSL communication.

19.1.4 Changing the Event Channel Operating Environment

The following information in the event channel operating environment can be set using the Interstage Management Console or the *essetcnf* *essetcnfchnl* command:

- Event Service configuration information
- Event channel environment information



Note

When the environment of an Event Service in persistent channel operation is modified, only the following configuration information and environment information can be modified.

- Event Data Waiting Time
- Error Log File Size
- Local Transaction Timeout
- 2-Phase Commit Transaction Timeout **Windows32/64** **Solaris32** **Linux32/64**
- Error return when consumer is disconnected
- Event Channels Auto Start

If you try to change other information, the operating environment of the event channel in persistent channel operation may be modified. This means the integrity of the persistent information may not be retained, and the event channel in persistent channel operation must be recreated.

Using the *essetcnf*/*essetcnfchnl* command

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 *esetcnfchnl* command

The following describes the command options.

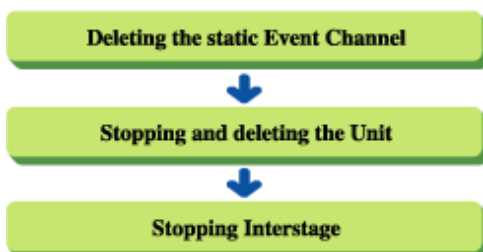
| Option | Description |
|---|---|
| -schmax | Maximum number of activations of the static event channel used by a topic or queue |
| -dchmax | Maximum number of activations of the dynamic event channel used by a temporary topic or temporary queue |
| -edmax | Maximum number of messages that can be stored in an event channel |
| -wtime | Message waiting time (second) (*1) |
| -logsize | Size of the log file (in Kbytes) which stores error information from the Event Service |
| Windows32/64 Solaris32 Linux32/64 -gtrnmax | Number of global transactions that can be executed simultaneously |
| -ltrntime | Local transaction timeout period (in seconds) |
| Windows32/64 Solaris32 Linux32/64 -2pctime | Two-phase commit timeout monitoring period (in seconds) |
| Windows32/64 Solaris32 Linux32/64 -retrytime | Retry interval during recovery (in seconds) |
| -retrymax | Recovery retry count |
| -chkcon | Error return mode during disconnection of subscribers |
| -autostart | Activate automatically the Event Channel when the Event Service starts |

*1) A value of ten or more is recommended. If setting to a lower value, configure operations that do not wait for message using the `receiveNoWait()` method.

Specify a value that is at least 20 seconds less than the `period_receive_timeout` value of the CORBA service operating environment file (`config`).

19.2 Environment Deletion for the Event Channel Operation Machine after Operation

The following figure shows the procedure for environment deletion after the operation:



19.2.1 Deleting the Static Event Channel

Delete the event channel used for sending/receiving messages by the JMS application by using the Interstage Management Console or the *esrmchnl* command.

Using the *esrmchnl* command

The Static Event Channel can be deleted using the *esmkchnl* command.

Example

Specify as shown below to delete all event channels belonging to the group 'mygroup'.

```
esrmchnl -g mygroup
```

19.2.2 Stopping and Deleting a Unit

Delete the unit using the Interstage Management Console or the *esrmunit* command.

Using the *esrmunit* command

The unit can be deleted using the *esrmunit* command after the *esstopunit* command has been used to forcibly stop the unit.

Example

Stopping the unit name 'unit1' forcibly by using the *esstopunit* command.

```
esstopunit -unit unit1 -o off
```

Deleting the unit name 'unit1' by using the *esrmunit* command.

```
esrmunit -unit unit1
```

19.2.3 Stopping Interstage

Stop the Event Service forcibly by stopping Interstage using the Interstage Management Console or the *isstop* command.

Using the *isstop* command

Interstage can be stopped by using the *isstop* command.

```
isstop -f
```

19.3 Environment Settings for the JMS Application Operation Machine before Operation

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

Note

Check that the paths and class files required for the operation of JMS applications are defined in the following environment variables (default installation path).

Windows32/64

- Environment variable PATH

JDK path (*1)

C:\Interstage\J2EE\bin

- Environment variable CLASSPATH

C:\Interstage\ODWIN\etc\Class\ODjava4.jar

C:\Interstage\eswin\lib\esnotifyjava4.jar (*2)

C:\Interstage\J2EE\lib\isj2ee.jar

C:\Interstage\jms\lib\fjmsprovider.jar

C:\Interstage\ots\lib\fjtsclient.jar (*3) (*4)

Solaris32/64

- Environment variable PATH

JDK path (*1)

/opt/FJSVj2ee/bin

/opt/FJSVjms/bin

- Environment variable CLASSPATH

/opt/FSUNod/etc/class/ODjava4.jar

/opt/FJSVes/lib/esnotifyjava4.jar

/opt/FJSVj2ee/lib/isj2ee.jar

/opt/FJSVjms/lib/fjmsprovider.jar

/opt/FSUNots/lib/fjtsclient.jar (*3) Solaris32

- Environment variable LD_LIBRARY_PATH

/opt/FSUNod/lib (*5)

/opt/FJSVjms/lib

Linux32/64

- Environment variable PATH

JDK path (*1)

/opt/FJSVj2ee/bin

/opt/FJSVjms/bin

- Environment variable CLASSPATH

/opt/FJSVod/etc/class/ODjava4.jar

/opt/FJSVes/lib/esnotifyjava4.jar

/opt/FJSVj2ee/lib/isj2ee.jar

/opt/FJSVjms/lib/fjmsprovider.jar

/opt/FJSVots/lib/fjtsclient.jar (*3)

- **Environment variable LD_LIBRARY_PATH**

/opt/FJSVod/lib (*6)

/opt/FJSVjms/lib

*1) If multiple JDKs are installed, make a setting so that the JDK to be used has a valid path.

*2) If the Interstage client function is installed, set the following class file:

```
C:\Interstage\ODWIN\etc\Class\esnotifyjava4.jar
```

*3) Required to use the global transaction function.

*4) Must be retrieved by the database linkage service from the host in which the function is installed.

*5) Do not specify /opt/FSUNod/lib/nt for LD_LIBRARY_PATH, as the JMS does not operate when this is specified.

*6) Do not specify /opt/FJSVod/lib/nt for LD_LIBRARY_PATH as the JMS does not operate when this is specified.



Specifying ORB

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'.

Example settings for the orb.properties file:

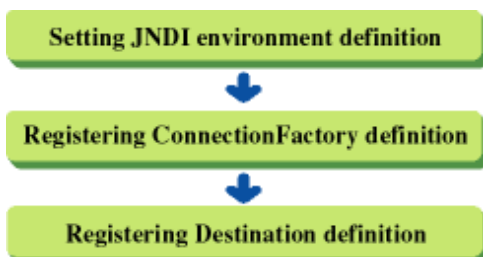
```
org.omg.CORBA.ORBClass=com.fujitsu.ObjectDirector.CORBA.ORB  
org.omg.CORBA.ORBSingletonClass=com.fujitsu.ObjectDirector.CORBA.SingletonORB
```

ConnectionFactory definitions

If a large number of ConnectionFactory definitions have been registered, then login processing for the Interstage Management Console might take longer.

Using the *jmsinfact* command, check whether unnecessary ConnectionFactory definitions have been registered. If that is the case, then delete them (if necessary) using the *jmsrmfact* command.

The following figure shows the procedure for environment setting before operation:



19.3.1 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.

Refer to "JNDI Service Provider Environment Setup" in the "JNDI" chapter for information on environment property.

Refer to "Description in Deployment Descriptor File" in the "JNDI" chapter for information on deployment descriptor.

19.3.2 Registering ConnectionFactory Definition

Register the ConnectionFactory definitions to be referenced by the JMS application by using the Interstage Management Console.

The ConnectionFactory definition is required in order to connect the JMS application with the Interstage JMS provider.

This definition contains the connection parameters, including the client identifier and the transaction type.

If the Interstage client function has been installed, use the JMS operation command to register the ConnectionFactory definition. For details about JMS Operation Commands, refer to the "JMS Operation Commands" chapter in the Reference Manual (Command Edition).

Specify the following option when using the *jmsmkfact* command to register the ConnectionFactory definition:

- **-t**
Specify when ConnectionFactory is of TopicConnectionFactory type.
- **-q**
Specify when ConnectionFactory is of QueueConnectionFactory type.
- **-x** Windows32/64 Solaris32 Linux32/64
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 of the JNDI name "TopicCF001" is registered as the default definition of the TopicConnectionFactory type.

The ConnectionFactory definition of the JNDI name "QueueCF001" is registered as the default definition of the QueueConnectionFactory type.

In the default definition, the Update ConnectionFactory Configuration Settings page, *isj2eeadmin* command, or *jmsmkfact* command can be specified to change the following settings:

- Client Identifier (specified in the -i option)
 - Global Transaction (specified in the -x option) Windows32/64 Solaris32 Linux32/64
-

19.3.3 Registering Destination Definition

Register the Destination definitions to be referenced by the JMS application by using the Interstage Management Console.

The Destination definition is addressing information that the JMS application uses to send and receive messages.

This definition includes the event channel name and the group name of the Event Service (Notification Service).

If the Interstage client function has been installed, use the JMS operation command to register the ConnectionFactory definition. For details about JMS Operation Commands, refer to the "JMS Operation Commands" chapter in the Reference Manual (Command Edition).

Specify the following option when using the *jmsmkdst* command to register the Destination definition.

- **-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

- To define a Destination that is associated with an Event Channel of another server, follow the method detailed below to specify the "host name or IP address" and "port number" registered for the Event Channel on the server that operates JMS applications, and then create the Destination.

If the information set for "host definition" and "port number" in the Naming Service of the server that operates JMS applications is the same as the information on the server containing the Event Channel, there is no need to specify the "host name or IP address" and "port number" when creating the Destination.

- Interstage Management Console

- On the Standalone Server

[System] > [Resources] > [JMS] > [Destination] > [Create the new Destination]

- On the Operate in Batch of the Admin Server **Windows32/64** **Solaris32** **Linux32/64**

[Operate in Batch] > [Interstage] > [Interstage Application Server] > [Resources] > [JMS] > [Destination] > [Modify Definition]

*) In server group applications, it is not possible to register a Destination definition for an Event Channel created in a different server group.

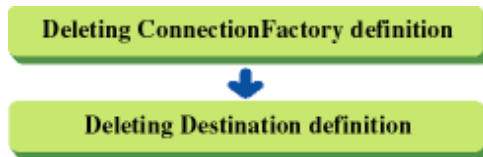
- isj2eadmin command

- *jmsmkdst* command

- Check if name resolution (IP address resolution) is possible by checking the contents of the host file or DNS settings when "Host name or IP address" is specified.

19.4 Environment Deletion for the JMS Application Operation Machine after Operation

The following figure shows the procedure for environment deletion after operation:



19.4.1 Deleting ConnectionFactory Definition

Delete the ConnectionFactory definitions by using the Interstage Management Console.

If the Interstage client function has been installed, use the JMS operation command to register the ConnectionFactory definition. For details about JMS Operation Commands, refer to the "JMS Operation Commands" chapter in the Reference Manual (Command Edition).

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 of the JNDI name "TopicCF001" is registered as the default definition of the TopicConnectionFactory type.

The ConnectionFactory definition of the JNDI name "QueueCF001" is registered as the default definition of the QueueConnectionFactory type.

The default definition cannot be deleted.

19.4.2 Deleting Destination Definition

Delete the Destination definitions by using the Interstage Management Console.

If the Interstage client function has been installed, use the JMS operation command to register the ConnectionFactory definition. For details about JMS Operation Commands, refer to the "JMS Operation Commands" chapter in the Reference Manual (Command Edition).

Example

Specify as shown below to delete the Destination definition whose JNDI name is 'java:comp/env/jms/TestTopic'.

```
jmsrmdst TestTopic
```

19.4.3 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
```

Note

If a durable Subscriber is deleted, you must activate the event channel by using the *esstartchnl* command.

Point

Durable Subscription name and client ID can be checked with the *jmsinfods* command.

Chapter 20 Developing a JMS Application

This chapter describes the development procedure of a JMS application.

20.1 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.
- Improving throughput when storing messages in the queue
Increase the send application (Sender) concurrency level



.....
Create an application that conforms to the Java Message Service 1.1 terms published by Oracle Corporation. The behavior of applications that conform to the Java Message Service 1.1 terms is also guaranteed for applications that conform to the Java Message Service 1.0.2 terms.
.....

20.2 Creating a JMS Application

Create an application in accordance with the Java Message Service 1.1 specification released by the corporation formerly known as Sun (now Oracle Corporation).

Applications created in accordance with the Java Message Service 1.0.2 specification are also reliable when run in accordance with the Java Message Service 1.1 specification.

20.2.1 Publish/Subscribe Messaging Model and Point-To-Point Messaging Model

The same API can be used for the creation of Publish/Subscribe and Point-to-Point messaging model applications.

MessageProducer (for sent applications) and MessageConsumer (for received applications) are used.

MessageProducer sends messages to the Event Channel, and MessageConsumer makes a request for messages from the Event Channel.

Creating a MessageProducer

MessageProducer sends messages to the Event Channel. Examples showing the procedure for the sending of messages to the Event Channel by MessageProducer and the processing flow are described below.

MessageProducer

```
import javax.jms.*;
import javax.naming.*;

public class Producer {
    public static void main( String[] args) {
        Connection connection = null;
        ...
        try {
            java.util.Hashtable env = new java.util.Hashtable( );
            env.put ( javax.naming.Context.INITIAL_CONTEXT_FACTORY ,
                "com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient" );
            InitialContext initialContext = new InitialContext(env);          /* 1 */
            ConnectionFactory connectionFactory = (ConnectionFactory)
                initialContext.lookup( "java:comp/env/jms/ConnectionFactory" ); /* 2 */
            Destination destination = (Destination)
                initialContext.lookup( "java:comp/env/jms/Destination" );     /* 3 */
            connection = connectionFactory.createConnection();               /* 4 */
            Session session = connection.createSession(false,
                Session.AUTO_ACKNOWLEDGE);                                    /* 5 */
            MessageProducer messageProducer =
                session.createProducer(destination);                          /* 6 */
            TextMessage message = session.createTextMessage( );
            message.setText( "Message" );
            messageProducer.send(message);                                    /* 7 */
        } catch( Exception e ) {
            ...
        }
        finally {
            if( null != connection ) {
                try {
                    connection.close();                                       /* 8 */
                }
                catch( Exception e ) {
                    ...
                }
            }
        }
    }
}
```

1. Construct a JNDI start context.
2. Acquire a ConnectionFactory object (if the JNDI name is 'ConnectionFactory').
3. Acquire a Destination object (if the JNDI name is 'Destination').
4. Create Connection.
5. Create Session.
6. Create MessageProducer.
7. Send a message. (The Message type is "TextMessage", and the sent message is "Message")
8. Close Connection.



Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to "J2EE Application Client" in the "JNDI" chapter.

Creating a MessageConsumer

MessageConsumer makes a request for messages from the Event Channel. Examples detailing how to request messages from the Event Channel using MessageConsumer, and the processing flow are described below.

MessageConsumer

```
import javax.jms.*;
import javax.naming.*;

public class Consumer {
    public static void main( String[] args) {
        Connection connection = null;
        ...
        try {
            java.util.Hashtable env = new java.util.Hashtable( );
            env.put ( javax.naming.Context.INITIAL_CONTEXT_FACTORY ,
                "com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient" );
            InitialContext initialContext = new InitialContext(env);          /* 1 */
            ConnectionFactory connectionFactory = (ConnectionFactory)
                initialContext.lookup("java:comp/env/jms/ConnectionFactory"); /* 2 */
            Destination destination = (Destination)
                initialContext.lookup("java:comp/env/jms/Destination");      /* 3 */
            connection = connectionFactory.createConnection();                /* 4 */
            Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE); /* 5 */
            MessageConsumer messageConsumer = session.createConsumer(destination); /* 6 */
            connection.start();                                               /* 7 */
            Message message = messageConsumer.receive();                       /* 8 */
        } catch( Exception e ) {
            ...
        }
        finally {
            if( null != connection ) {
                try {
                    connection.close();                                       /* 9 */
                }
                catch( Exception e ) {
                    ...
                }
            }
        }
    }
}
```

1. Construct a JNDI start context.
2. Acquire a ConnectionFactory object (if the JNDI name is 'ConnectionFactory').
3. Acquire a Destination object (if the JNDI name is 'Destination').
4. Create Connection.
5. Create Session.
6. Create MessageConsumer.
7. Start the delivery of a message upon connection.
8. Receive a message.

9. Close Connection.



For information on the specification of an environment property when a JNDI start context is constructed, refer to "J2EE Application Client" in the "JNDI" chapter.

20.2.2 Create a Message Listener

This section describes how to develop a receiving application using Message Listener.

Message Listener is a function of automatically handling a message when it arrives at a receiver.

Creating a MessageConsumer using Message Listener

To handle a received message, register a Message Listener object. When a message arrives, Message Listener is invoked. The following procedure example describes the processing flow for receiving a message using Message Listener.

MessageConsumer using Message Listener

```
import javax.jms.*;
import javax.naming.*;

public class ConsumerA implements MessageListener,ExceptionListener { /* 1 */
    public static void main( String[] args ) {
        ...
        Connection connection = null;
        ConsumerA consumerA = new ConsumerA (); /* 2 */
        ...
        try {
            java.util.Hashtable env = new java.util.Hashtable( );
            env.put ( javax.naming.Context.INITIAL_CONTEXT_FACTORY ,
                "com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient" );
            InitialContext initialContext = new InitialContext(env); /* 3 */
            ConnectionFactory connectionFactory = (ConnectionFactory)
                initialContext.lookup("java:comp/env/jms/ConnectionFactory"); /* 4 */
            Destination destination = (Destination)
                initialContext.lookup("java:comp/env/jms/Destination"); /* 5 */
            connection = connectionFactory.createConnection(); /* 6 */
            connection.setExceptionListener( consumerA ); /* 7 */
            Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
            /* 8 */
            MessageConsumer consumer = session.createConsumer(destination); /* 9 */
            consumer.setMessageListener(consumerA); /* 10 */
            connection.start(); /* 11 */
            /* Wait processing*/
        } catch( Exception e ) {
            ...
        }
        finally {
            if( null != connection ) {
                try {
                    connection.close(); /* 12 */
                }
                catch( Exception e ) {
                    ...
                }
            }
        }
    }
    public void onMessage(Message message) {
        ...
    }
}
```

```

    try {
    } catch( JMSEException e ) {
        ...
    }
    ...
}

public void onException( JMSEException jmsEx ) {
    ...
}
}

```

1. Implement a MessageListener interface and ExceptionListener interface.
2. Create a class instance.
3. Construct a JNDI start context.
4. Acquire a ConnectionFactory object (if the JNDI name is 'ConnectionFactory').
5. Acquire a Destination object (if the JNDI name is 'Destination').
6. Create Connection.
7. Register the ExceptionListener object.
8. Create Session.
9. Create MessageConsumer.
10. Register the MessageListener object.
11. Start the delivery of a message upon connection.
12. Close Connection.



Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to "J2EE Application Client" in the "JNDI" chapter.

20.2.3 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 MessageConsumer 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.

MessageConsumer using the Durable Subscription Function

```

import javax.jms.*;
import javax.naming.*;

public class SubscriberD {
    public static void main( String[] args ) {
        Connection connection = null;
        ...
        try {
            java.util.Hashtable env = new java.util.Hashtable( );
            env.put ( javax.naming.Context.INITIAL_CONTEXT_FACTORY ,
                "com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient" );

```


20.2.4 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 send (Message message, int deliveryMode, int priority, long timeToLive) method for the MessageProducer 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.



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.

20.2.5 Message Persistent Function

To use the message persistent function:

1. Use the *esmkchnl* command to create a persistent channel.
2. Specify `javax.jms.DeliveryMode.PERSISTENT` in argument `deliveryMode` of the send method of the MessageProducer interface.

20.2.6 Local Transaction

Creating a MessageProducer using a Local Transaction

MessageProducer in a Local Transaction

```
import javax.jms.*;
import javax.naming.*;

public class Producer {
    public static void main( String[] args ) {
        Connection connection = null;
        ...
        try {
            java.util.Hashtable env = new java.util.Hashtable( );
            env.put ( javax.naming.Context.INITIAL_CONTEXT_FACTORY ,
                    "com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient" );
            InitialContext initialContext = new InitialContext(env);          /* 1 */
            ConnectionFactory connectionFactory = (ConnectionFactory)
                initialContext.lookup("java:comp/env/jms/ConnectionFactory"); /* 2 */
            Destination destination = (Destination)
                initialContext.lookup("java:comp/env/jms/Destination");      /* 3 */
            connection = connectionFactory.createConnection();                /* 4 */
            Session session = connection.createSession(true, 0);             /* 5 */
            MessageProducer producer = session.createProducer(destination); /* 6 */
            TextMessage message = session.createTextMessage( );
            message.setText( "Message" );
            producer.send(message);                                          /* 7 */
            session.commit();                                                /* 8 */
        } catch( Exception e ) {
            ...
        }
        finally {
            if( null != connection ) {
```

```

        try {
            connection.close();           /* 9 */
        }
        catch( Exception e ) {
            ...
        }
    }
}
}
}

```

1. Construct a JNDI start context.
2. Acquire a ConnectionFactory object (if the JNDI name is 'ConnectionFactory').
3. Acquire a Destination object (if the JNDI name is 'Destination').
4. Create Connection.
5. Create Session.
6. Create MessageProducer.
7. Send a message. (The Message type is "TextMessage", and the sent message is "Message")
8. Make commitment.
9. Close Connection.

Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to "J2EE Application Client" in the "JNDI" chapter.

Creating a MessageConsumer using a Local Transaction

MessageConsumer in a Local Transaction

```

import javax.jms.*;
import javax.naming.*;

public class Consumer {
    public static void main( String[] args ) {
        Connection connection = null;
        ...
        try {
            java.util.Hashtable env = new java.util.Hashtable( );
            env.put ( javax.naming.Context.INITIAL_CONTEXT_FACTORY ,
                    "com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient" );
            InitialContext initialContext = new InitialContext(env);           /* 1 */
            ConnectionFactory connectionFactory = (ConnectionFactory)
                initialContext.lookup("java:comp/env/jms/ConnectionFactory"); /* 2 */
            Destination destination = (Destination)
                initialContext.lookup("java:comp/env/jms/Destination");       /* 3 */
            connection = connectionFactory.createConnection();                 /* 4 */
            Session session = connection.createSession(true, 0);              /* 5 */
            MessageConsumer consumer = session.createConsumer(destination);    /* 6 */
            connection.start();                                                 /* 7 */
            Message message = consumer.receive();                               /* 8 */
            session.commit();                                                   /* 9 */
        } catch( Exception e ) {
            ...
        }
    }
}

```

```

        if( null != connection ) {
            try {
                connection.close();
            }
            catch( Exception e ) {
                ...
            }
        }
    }
}
}
}

```

1. Construct a JNDI start context.
2. Acquire a ConnectionFactory object (if the JNDI name is 'ConnectionFactory').
3. Acquire a Destination object (if the JNDI name is 'Destination').
4. Create Connection.
5. Create Session.
6. Create MessageConsumer.
7. Start the delivery of a message upon connection.
8. Receive a message.
9. Make commitment.
10. Close Connection.

Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to "J2EE Application Client" in the "JNDI" chapter.

20.2.7 Global Transaction Windows32/64 Solaris32 Linux32/64

Creating a MessageProducer using a Global Transaction Windows32/64 Solaris32 Linux32/64

MessageProducer in a Global Transaction

```

import javax.jms.*;
import javax.naming.*;

public class Producer {
    public static void main( String[] args ) {
        Connection connection = null;
        ...
        try {
            java.util.Hashtable env = new java.util.Hashtable( );
            env.put ( javax.naming.Context.INITIAL_CONTEXT_FACTORY ,
                "com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient" );
            InitialContext initialContext = new InitialContext(env);
            ConnectionFactory connectionFactory = (ConnectionFactory)
                initialContext.lookup("java:comp/env/jms/ConnectionFactory");
            Destination destination = (Destination)
                initialContext.lookup("java:comp/env/jms/Destination");
            javax.transaction.UserTransaction ut =(javax.transaction.UserTransaction)
                initialContext.lookup("java:comp/UserTransaction");
            connection = connectionFactory.createConnection();
            Session session = connection.createSession(true, 0);
            MessageProducer producer = session.createProducer(destination);

```

```

        TextMessage Message = session.createTextMessage( );
        Message.setText( "Message" );
        ut.begin();                               /* 8 */
        producer.send(Message);                  /* 9 */
        ut.commit();                             /* 10 */
    } catch( Exception e ) {
        ...
    }
    finally {
        if( null != connection ) {
            try {
                connection.close();             /* 11 */
            }
            catch( Exception e ) {
                ...
            }
        }
    }
}
}
}
}
}

```

1. Construct a JNDI start context.
2. Acquire a ConnectionFactory object (if the JNDI name is 'ConnectionFactory').
3. Acquire a Destination object (if the JNDI name is 'Destination').
4. Acquire a UserTransaction object.
5. Create Connection.
6. Create Session.
7. Create MessageProducer.
8. Start a global transaction.
9. Send a message. (The Message type is "TextMessage", and the sent message is "Message")
10. Complete the global transaction.
11. Close Connection.



Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to "J2EE Application Client" in the "JNDI" chapter.

Creating a MessageConsumer using a Global Transaction Windows32/64 Solaris32 Linux32/64

MessageConsumer in a Global Transaction

```

import javax.jms.*;
import javax.naming.*;

public class Consumer {
    public static void main( String[] args ) {
        Connection connection = null;
        ...
        try {
            java.util.Hashtable env = new java.util.Hashtable( );
            env.put ( javax.naming.Context.INITIAL_CONTEXT_FACTORY ,
                "com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient" );
            InitialContext initialContext = new InitialContext(env);           /* 1 */
            ConnectionFactory connectionFactory = (ConnectionFactory)

```



```
java -Dcom.fujitsu.ObjectDirector.CORBA.GlobalTransactionMode=True Publisher
```

20.2.8 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.

| Location | JMS Message | StructuredEvent |
|----------|------------------------------|---|
| Header | JMSPriority JMSExpiration | QoS property - Priority - Timeout FixedHeader: Automatically added by JMS-Provider - domain_name='JMS' - type_name='v6.2' - event_name="" |
| Property | Not converted | |
| 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.

| StructuredEvent | JMS Message |
|-----------------------------|--------------|
| string-type | TextMessage |
| wstring-type | |
| array of octet-type | BytesMessage |
| sequence-type of octet-type | |

20.2.9 Message Selector Function

To use the message selector function, 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 Consumer, Durable Subscriber, 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' when a Consumer is created.

```
String selector = "NAME = 'FUJITSU'";
session.createConsumer(destination, 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'
```

JMS Messages where the property with property name NUMBER has a property value of 1000 or more can be received.

```
NUMBER >= 1000
```

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
```

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 you to perform searches using a pattern search.

Specify '%' and '_' in a character string literal to perform a 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.

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.

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.

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 you to perform searches depending on whether a property is present.

 **Example**

JMS Messages without property name PROPERTY can be received.


```
PROPERTY IS NULL
```

JMS Messages with property name PROPERTY can be received.

```
PROPERTY IS NOT NULL
```

- IN conditional expression

The IN conditional expression allows you 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' )
```

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')
```

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')
```

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'))
```



Note

- 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.

20.2.10 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

```
import javax.jms.*;
import javax.naming.*;

public class Browser {
    public static void main( String[] args ) {
        Connection connection = null;
        ...
        try {
            java.util.Hashtable env = new java.util.Hashtable( );
            env.put ( javax.naming.Context.INITIAL_CONTEXT_FACTORY ,
                    "com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient" );
            InitialContext initialContext = new InitialContext(env);          /* 1 */
            ConnectionFactory connectionFactory = (ConnectionFactory)
                initialContext.lookup("java:comp/env/jms/ConnectionFactory"); /* 2 */
            Queue queue = (Queue)initialContext.lookup("java:comp/env/jms/Queue"); /* 3 */
            connection = connectionFactory.createConnection();                /* 4 */
            Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE); /* 5 */
            QueueBrowser queueBrowser = session.createBrowser(queue);        /* 6 */
            connection.start();                                              /* 7 */
            java.util.Enumeration e = queueBrowser.getEnumeration();         /* 8 */
            Message message;
            while ( e.hasMoreElements() ) {
                message = (Message)e.nextElement();                          /* 9 */
                ...
            }
            queueBrowser.close();                                           /* 10 */
        } catch( Exception e ) {
            ...
        }
        finally {
            if( null != connection ) {
                try {
                    connection.close();                                     /* 11 */
                }
                catch( Exception e ) {
                    ...
                }
            }
        }
    }
}
```

1. Construct a JNDI start context.
2. Acquire a ConnectionFactory object (if the JNDI name is 'ConnectionFactory').
3. Acquire a Queue object (if the JNDI name is 'Queue').
4. Create Connection.
5. Create Session.
6. Create QueueBrowser.
7. Start the delivery of a message upon connection.
8. Start browsing.
9. Acquire one message.
10. Close QueueBrowser.
11. Close Connection.



Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to "J2EE Application Client" in the "JNDI" chapter.

Only one queue browser can be used per channel.

20.2.11 Notes on Executing an Application

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
```



Point

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.

Preventing Message Loss

If the upper limit for data that can be stored for Event Channel messages is exceeded, messages may be lost.

To avoid loss of messages, modify the system properties to post an error message.

System property name

`com.fujitsu.interstage.jms.queue_max_err`

Specification

Set "yes".



Example

```
-Dcom.fujitsu.interstage.jms.queue_max_err=yes
```

JMS1.0.2 Specification API

Applications created in accordance with the Java Message Service 1.0.2 specification in version 8.0 or earlier of this product can be run the same way as version 8.0 or earlier by using the system properties.

System property name

com.fujitsu.interstage.jms.exception_version_lower

Specification

Set "yes".



Example

```
-Dcom.fujitsu.interstage.jms.exception_version_lower=yes
```

Received Message Timeouts

If a message cannot be received, the receiver timeout may exceed the timeout specified for the receive() method. For details on received message timeouts, refer to receive() method - (*1) of ["20.3.4 API List of the Package javax.jms \(Part 4\)"](#).

20.3 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.

20.3.1 API List of the Package javax.jms (Part 1)

| Interface name/Class name | Method | Support |
|--|-------------------------------------|----------------------------------|
| BytesMessage | getBodyLength() | Yes (refer to Note below) |
| | readBoolean() | Yes |
| | readByte() | Yes |
| | readBytes(byte[] value) | Yes |
| | readBytes(byte[] value, int length) | Yes |
| | readChar() | Yes |
| | readDouble() | Yes |
| | readFloat() | Yes |
| | readInt() | Yes |
| | readLong() | Yes |
| | readShort() | Yes |
| | readUnsignedByte() | Yes |
| | readUnsignedShort() | Yes |
| | readUTF() | Yes |
| | reset() | Yes |
| | writeBoolean(boolean value) | Yes |
| | writeByte(byte value) | Yes |
| | writeBytes(byte[] value) | Yes |
| writeBytes(byte[] value, int offset, int length) | Yes | |
| writeChar(char value) | Yes | |

| Interface name/Class name | Method | Support |
|---------------------------|---|----------------------------------|
| | writeDouble(double value) | Yes |
| | writeFloat(float value) | Yes |
| | writeInt(int value) | Yes |
| | writeLong(long value) | Yes |
| | writeObject(java.lang.Object value) | Yes |
| | writeShort(short value) | Yes |
| | writeUTF(java.lang.String value) | Yes |
| Connection | close() | Yes |
| | createConnectionConsumer(Destination destination, java.lang.String messageSelector, ServerSessionPool sessionPool, int maxMessages) | Yes (refer to Note below) |
| | createDurableConnectionConsumer(Topic topic, java.lang.String subscriptionName, java.lang.String messageSelector, ServerSessionPool sessionPool, int maxMessages) | Yes (refer to Note below) |
| | createSession(boolean transacted, int acknowledgeMode) | Yes (refer to Note below) |
| | getClientID() | Yes |
| | getExceptionListener() | Yes |
| | getMetaData() | Yes |
| | setClientID(java.lang.String clientID) | Yes |
| | setExceptionListener(ExceptionListener listener) | Yes |
| | start() | Yes |
| | stop() | Yes |
| ConnectionConsumer | close() | Yes |
| | getServerSessionPool() | Yes |
| ConnectionFactory | createConnection() | Yes (refer to Note below) |
| | createConnection(java.lang.String userName, java.lang.String password) | Yes (refer to Note below) |

Yes: Supported

No: Not supported

Note: This was added in the JMS 1.1 specification.

20.3.2 API List of the Package javax.jms (Part 2)

| Interface name/Class name | Method | Support |
|--|---|----------------------------------|
| ConnectionMetaData | getJMSMajorVersion() | Yes |
| | getJMSMinorVersion() | Yes |
| | getJMSProviderName() | Yes (refer to Note below) |
| | getJMSVersion() | Yes |
| | getJMSXPropertyNames() | Yes |
| | getProviderMajorVersion() | Yes |
| | getProviderMinorVersion() | Yes |
| | getProviderVersion() | Yes |
| DeliveryMode | No method | |
| Destination | No method | |
| ExceptionListener | onException(JMSException exception) | Yes |
| MapMessage | getBoolean(java.lang.String name) | Yes |
| | getByte(java.lang.String name) | Yes |
| | getBytes(java.lang.String name) | Yes |
| | getChar(java.lang.String name) | Yes |
| | getDouble(java.lang.String name) | Yes |
| | getFloat(java.lang.String name) | Yes |
| | getInt(java.lang.String name) | Yes |
| | getLong(java.lang.String name) | Yes |
| | getMapNames() | Yes |
| | getObject(java.lang.String name) | Yes |
| | getShort(java.lang.String name) | Yes |
| | getString(java.lang.String name) | Yes |
| | itemExists(java.lang.String name) | Yes |
| | setBoolean(java.lang.String name, boolean value) | Yes |
| | setByte(java.lang.String name, byte value) | Yes |
| | setBytes(java.lang.String name, byte[] value) | Yes |
| | setBytes(java.lang.String name, byte[] value, int offset, int length) | Yes |
| | setChar(java.lang.String name, char value) | Yes |
| | setDouble(java.lang.String name, double value) | Yes |
| | setFloat(java.lang.String name, float value) | Yes |
| | setInt(java.lang.String name, int value) | Yes |
| | setLong(java.lang.String name, long value) | Yes |
| | setObject(java.lang.String name, java.lang.Object value) | Yes |
| setShort(java.lang.String name, short value) | Yes | |
| setString(java.lang.String name, java.lang.String value) | Yes | |

Yes: Supported

No: Not supported

Note: 'Interstage Application Server' is returned.

20.3.3 API List of the Package javax.jms (Part 3)

| Interface name/Class name | Method | Support |
|---|--|---------|
| Message | acknowledge() | Yes |
| | clearBody() | Yes |
| | clearProperties() | Yes |
| | getBooleanProperty(java.lang.String name) | Yes |
| | getByteProperty(java.lang.String name) | Yes |
| | getDoubleProperty(java.lang.String name) | Yes |
| | getFloatProperty(java.lang.String name) | Yes |
| | getIntProperty(java.lang.String name) | Yes |
| | getJMSCorrelationID() | Yes |
| | getJMSCorrelationIDAsBytes() | No |
| | getJMSDeliveryMode() | Yes |
| | getJMSDestination() | Yes |
| | getJMSExpiration() | Yes |
| | getJMSMessageID() | Yes |
| | getJMSPriority() | Yes |
| | getJMSRedelivered() | Yes |
| | getJMSReplyTo() | Yes |
| | getJMSTimestamp() | Yes |
| | getJMSType() | Yes |
| | getLongProperty(java.lang.String name) | Yes |
| | getObjectProperty(java.lang.String name) | Yes |
| | getPropertyNames() | Yes |
| | getShortProperty(java.lang.String name) | Yes |
| | getStringProperty(java.lang.String name) | Yes |
| | propertyExists(java.lang.String name) | Yes |
| | setBooleanProperty(java.lang.String name, boolean value) | Yes |
| | setByteProperty(java.lang.String name, byte value) | Yes |
| | setDoubleProperty(java.lang.String name, double value) | Yes |
| | setFloatProperty(java.lang.String name, float value) | Yes |
| | setIntProperty(java.lang.String name, int value) | Yes |
| setJMSCorrelationID(java.lang.String correlationID) | Yes | |
| setJMSCorrelationIDAsBytes(byte[] correlationID) | No | |
| setJMSDeliveryMode(int deliveryMode) | Yes | |
| setJMSDestination(Destination destination) | Yes | |
| setJMSExpiration(long expiration) | Yes | |

| Interface name/Class name | Method | Support |
|---------------------------|--|---------|
| | setJMSMessageID(java.lang.String id) | Yes |
| | setJMSPriority(int priority) | Yes |
| | setJMSRedelivered(boolean redelivered) | Yes |
| | setJMSReplyTo(Destination replyTo) | Yes |
| | setJMSTimestamp(long timestamp) | Yes |
| | setJMSType(java.lang.String type) | Yes |
| | setLongProperty(java.lang.String name, long value) | Yes |
| | setObjectProperty(java.lang.String name, java.lang.Object value) | Yes |
| | setShortProperty(java.lang.String name, short value) | Yes |
| | setStringProperty(java.lang.String name, java.lang.String value) | Yes |

Yes: Supported

No: Not supported

20.3.4 API List of the Package javax.jms (Part 4)

| Interface name/Class name | Method | Support |
|---|---|----------|
| MessageConsumer | close() | Yes |
| | getMessageListener() | Yes |
| | getMessageSelector() | Yes |
| | receive() | Yes |
| | receive(long timeout) | Yes (*1) |
| | receiveNoWait() | Yes (*2) |
| | setMessageListener(MessageListener listener) | Yes |
| MessageListener | onMessage(Message message) | Yes |
| MessageProducer | close() | Yes |
| | getDeliveryMode() | Yes |
| | getDestination() | Yes (*3) |
| | getDisableMessageID() | Yes |
| | getDisableMessageTimestamp() | Yes |
| | getPriority() | Yes |
| | getTimeToLive() | Yes |
| | send(Destination destination, Message message) | Yes (*3) |
| | send(Destination destination, Message message, int deliveryMode, int priority, long timeToLive) | Yes (*3) |
| | send(Message message) | Yes (*3) |
| | send(Message message, int deliveryMode, int priority, long timeToLive) | Yes (*3) |
| | setDeliveryMode(int deliveryMode) | Yes (*4) |
| | setDisableMessageID(boolean value) | Yes |
| setDisableMessageTimestamp(boolean value) | Yes | |

| Interface name/Class name | Method | Support |
|---------------------------|--|----------|
| | setPriority(int defaultPriority) | Yes |
| | setTimeToLive(long timeToLive) | Yes (*5) |
| ObjectMessage | getObject() | Yes |
| | setObject(java.io.Serializable object) | Yes |

Yes: Supported

No: Not supported

*1 When there is no need to wait for a message, execute the no wait operation using the receiveNoWait() method where there is no wait for messages, rather than the receive() method. If the receive() method is used, there will be a wait for messages to be received, and a null value will be returned when the receiver timeout is reached. The receiver timeout is calculated using the value specified for the receive() method and the value set for the "Message waiting time" event channel operating environment, as shown below. For details on how to set the event channel operating environment, refer to "19.1.4 Changing the Event Channel Operating Environment".

```
Timeout for receiving messages = Message waiting time x n
n = Value of receive() / (Message waiting time x 1000)
n: Fractional digits, truncated to an integer
```



Example

When Value of receive() is "50000 milliseconds" and Message waiting time is "40 seconds"

```
n = 50000 / (40 x 1000) 2 (Fractional digits, truncated to an integer)
Timeout for receiving messages = 40 x 2 = 80(seconds)
```

*2 This method immediately returns null when there is no message which can be received.

*3 This was added in the JMS1.1 specification.

*4 Only the same mode as the delivery mode of the event channel is supported.

*5 timeToLive supports in milliseconds. However, since the message timeout of the event channel is in seconds, timeToLive is rounded off to the nearest value.

20.3.5 API List of the Package javax.jms (Part 5)

| Interface name/Class name | Method | Support |
|---------------------------|---|----------|
| Queue | getQueueName() | Yes |
| | toString() | Yes (*1) |
| QueueBrowser | close() | Yes |
| | getEnumeration() | Yes |
| | getMessageSelector() | Yes |
| | getQueue() | Yes |
| QueueConnection | createConnectionConsumer(Queue queue, java.lang.String messageSelector, ServerSessionPool sessionPool, int maxMessages) | Yes |
| | createQueueSession(boolean transacted, int acknowledgeMode) | Yes (*2) |
| QueueConnectionFactory | createQueueConnection() | Yes |
| | createQueueConnection(java.lang.String userName, java.lang.String password) | Yes |

| Interface name/Class name | Method | Support |
|---------------------------|---|----------|
| QueueReceiver | getQueue() | Yes |
| QueueRequestor | close() | Yes |
| | request(Message message) | Yes |
| QueueSender | getQueue() | Yes |
| | send(Message message) | Yes |
| | send(Message message, int deliveryMode, int priority, long timeToLive) | Yes (*3) |
| | send(Queue queue, Message message) | Yes |
| | send(Queue queue, Message message, int deliveryMode, int priority, long timeToLive) | Yes (*3) |
| QueueSession | createBrowser(Queue queue) | Yes |
| | createBrowser(Queue queue, java.lang.String messageSelector) | Yes |
| | createQueue(java.lang.String queueName) | Yes |
| | createReceiver(Queue queue) | Yes |
| | createReceiver(Queue queue, java.lang.String messageSelector) | Yes |
| | createSender(Queue queue) | Yes |
| | createTemporaryQueue() | Yes |

Yes: Supported

No: Not supported

*1 'com.fujitsu.interstage.jms:<Queue name>::<the group name of the event channel>::<the channel name of the event channel>' is returned.

*2 userName and password are ignored.

*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.

20.3.6 API List of the Package javax.jms (Part 6)

| Interface name/Class name | Method | Support |
|---|---|----------|
| ServerSession | getSession() | Yes |
| | start() | Yes |
| ServerSessionPool | getServerSession() | Yes |
| Session | close() | Yes |
| | commit() | Yes |
| | createBrowser(Queue queue) | Yes (*1) |
| | createBrowser(Queue queue, java.lang.String messageSelector) | Yes (*1) |
| | createBytesMessage() | Yes |
| | createConsumer(Destination destination) | Yes (*1) |
| | createConsumer(Destination destination, java.lang.String messageSelector) | Yes (*1) |
| createConsumer(Destination destination, | Yes (*1) | |

| Interface name/Class name | Method | Support |
|--|--|----------|
| | java.lang.String messageSelector, boolean NoLocal) | |
| | createDurableSubscriber(Topic topic, java.lang.String name) | Yes (*1) |
| | createDurableSubscriber (Topic topic, java.lang.String name, java.lang.String messageSelector, boolean noLocal) | Yes (*1) |
| | createMapMessage() | Yes |
| | createMessage() | Yes |
| | createObjectMessage() | Yes |
| | createObjectMessage(java.io.Serializable object) | Yes |
| | createProducer(Destination destination) | Yes (*1) |
| | createQueue(java.lang.String queueName) | Yes (*1) |
| | createStreamMessage() | Yes |
| | createTemporaryQueue() | Yes (*1) |
| | createTemporaryTopic() | Yes (*1) |
| | createTextMessage() | Yes |
| | createTextMessage(java.lang.String text) | Yes |
| | createTopic(java.lang.String topicName) | Yes (*1) |
| | getAcknowledgeMode() | Yes (*1) |
| | getMessageListener() | Yes |
| | getTransacted() | Yes |
| | recover() | Yes |
| | rollback() | Yes |
| | run() | Yes |
| setMessageListener(MessageListener listener) | Yes (*2) | |
| unsubscribe() | Yes (*1) | |
| unsubscribe(java.lang.String name) | Yes (*1) | |
| StreamMessage | readBoolean() | Yes |
| | readByte() | Yes |
| | readBytes(byte[] value) | Yes |
| | readChar() | Yes |
| | readDouble() | Yes |
| | readFloat() | Yes |
| | readInt() | Yes |
| | readLong() | Yes |
| | readObject() | Yes |
| | readShort() | Yes |

| Interface name/Class name | Method | Support |
|-------------------------------------|--|---------|
| | readString() | Yes |
| | reset() | Yes |
| | writeBoolean(boolean value) | Yes |
| | writeByte(byte value) | Yes |
| | writeBytes(byte[] value) | Yes |
| | writeBytes(byte[] value, int offset, int length) | Yes |
| | writeChar(char value) | Yes |
| | writeDouble(double value) | Yes |
| | writeFloat(float value) | Yes |
| | writeInt(int value) | Yes |
| | writeLong(long value) | Yes |
| | writeObject(java.lang.Object value) | Yes |
| | writeShort(short value) | Yes |
| writeString(java.lang.String value) | Yes | |
| TemporaryQueue | delete() | Yes |
| TemporaryTopic | delete() | Yes |

Yes: Supported

No: Not supported

*1 This was added in the JMS1.1 specification.

*2 Synchronous and asynchronous messages cannot be received simultaneously by one MessageConsumer.

20.3.7 API List of the Package javax.jms (Part 7)

| Interface name/class name | Method | Support |
|---------------------------|---|----------|
| TextMessage | getText() | Yes |
| | setText(java.lang.String string) | Yes |
| Topic | getTopicName() | Yes |
| | toString() | Yes (*1) |
| TopicConnection | createConnectionConsumer(Topic topic, java.lang.String messageSelector, ServerSessionPool sessionPool, int maxMessages) | Yes |
| | createDurableConnectionConsumer(Topic topic, java.lang.String subscriptionName, java.lang.String messageSelector, ServerSessionPool sessionPool, int maxMessages) | Yes |
| | createTopicSession(boolean transacted, int acknowledgeMode) | Yes |
| TopicConnectionFactory | createTopicConnection() | Yes |
| | createTopicConnection(java.lang.String userName, java.lang.String password) | Yes (*2) |
| TopicPublisher | getTopic() | Yes |
| | publish(Message message) | Yes |
| | publish(Message message, int deliveryMode, int priority, long timeToLive) | Yes (*3) |

| Interface name/class name | Method | Support |
|---------------------------|--|----------|
| | publish(Topic topic, Message message) | Yes |
| | publish(Topic topic, Message message, int deliveryMode, int priority, long timeToLive) | Yes (*3) |
| TopicRequestor | close() | Yes |
| | request(Message message) | Yes |
| TopicSession | createDurableSubscriber(Topic topic, java.lang.String name) | Yes |
| | createDurableSubscriber(Topic topic, java.lang.String name, java.lang.String messageSelector, boolean noLocal) | Yes |
| | createPublisher(Topic topic) | Yes |
| | createSubscriber(Topic topic) | Yes |
| | createSubscriber(Topic topic, java.lang.String messageSelector, boolean noLocal) | Yes |
| | createTemporaryTopic() | Yes |
| | createTopic(java.lang.String topicName) | Yes |
| TopicSubscriber | unsubscribe(java.lang.String name) | Yes |
| | getNoLocal() | Yes |
| | getTopic() | Yes |

Yes: Supported

No: Not supported

*1 'com.fujitsu.interstage.jms:<Topic name>::<the group name of the event channel>::<the channel name of the event channel>' is returned.

*2 userName and password are ignored.

*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.

20.3.8 API List of the Package javax.jms (Part 8)

| Interface name/class name | Method | Support |
|---------------------------|---|---------|
| XAConnection | createSession(boolean transacted, int acknowledgeMode) | No |
| | createXASession() | No |
| XAConnectionFactory | createXAConnection() | No |
| | createXAConnection(java.lang.String userName, java.lang.String password) | No |
| XAQueueConnection | createQueueSession(boolean transacted, int acknowledgeMode) | No |
| | createXAQueueSession() | No |
| XAQueueConnectionFactory | createXAQueueConnection() | No |
| | createXAQueueConnection(java.lang.String userName, java.lang.String password) | No |
| XAQueueSession | getQueueSession() | No |
| XASession | commit() | No |
| | getSession() | No |

| Interface name/class name | Method | Support |
|---------------------------|---|---------|
| | getTransacted() | No |
| | getXAResource() | No |
| | rollback() | No |
| XATopicConnection | createTopicSession(boolean transacted, int acknowledgeMode) | No |
| | createXATopicSession() | No |
| XATopicConnection | createXATopicConnection() | No |
| Factory | createXATopicConnection(java.lang.String userName, java.lang.String password) | No |
| XATopicSession | getTopicSession() | No |

Yes: Supported

No: Not supported

Note: The XA interface is not supported.

Part 7 Connector Edition

| | |
|---|-----|
| Chapter 21 Basic Functions of the Interstage Connector..... | 427 |
|---|-----|

Chapter 21 Basic Functions of the Interstage Connector

This chapter explains the basic function of Interstage Connector.

Note

- The connector1.5 specification-compliant resource adapter can only be used when the IJServer type is as shown below. An error will occur if you try to deploy the connector1.5 specification-compliant resource adapter to an IJServer that is not of the following type:
 - Run Web applications and EJB applications on the same JavaVM
- There are two methods of deploying the connector1.5 specification-compliant resource adapter: Deployment to the IJServer and Deployment of resources without specifying the IJServer. For the connector1.5 specification-compliant resource adapter, however, use the first method (deployment to the IJServer). An error will occur if you try to use the second method (deployment of resources without specifying the IJServer).

21.1 Resource Adapter Types

There are two types of resource adapter: a synchronous type resource adapter that notifies requests from J2EE applications to EIS (called an "**outbound resource adapter**"), and a resource adapter that can process messages notified from EIS asynchronously in the J2EE application (called an "**inbound resource adapter**"). Depending on the resource adapter type, the **management target object** that manages resource adapter-specific information may be defined.

Each type is explained below.

Outbound Resource Adapter

The outbound resource adapter is a resource adapter that runs synchronously with the application. It accepts processing requests from the application and executes EIS processing.

To use the outbound resource adapter, specify the ConnectionFactory definition name using the Interstage Management Console. By specifying the definition name in the JNDI lookup method and executing it, the application obtains ConnectionFactory and executes processing requests to EIS. By using JNDI, processing requests for connection, transaction and security are managed by the IJServer.

Inbound Resource Adapter

This resource adapter was added from the connector1.5 specification. The inbound resource adapter runs asynchronously to the application, and makes requests for processing to the application when necessary. It accepts processing requests from EIS, calls Message-driven Bean at the necessary time, and delivers processing requests to the application server. Messages that are delivered are managed in the EJB container according to the Message-driven Bean definition.

Federation with Message-driven Bean

Applications received by the inbound resource adapter can be created in Message-driven Bean. In the EJB 2.1 specification and higher Message-driven Bean, "JMS" or "resource adapter" can be selected as the configuration for the message that is received (called a "**target type for receipt**"). If "resource adapter" is selected, the name of the resource adapter that is the target of receipt is selected. Message-driven Bean must be implemented according to the MessageListener interface of the resource adapter that is selected.

Management target Object

Management target objects may be defined in the resource adapter that follows the connector1.5 (and higher) specification. Resource adapter-specific information may need to be managed separately to the ConnectionFactory. Objects that manage this specific information can be defined in the resource adapter as management target objects. To use management target objects, specify the management target object definition name using the Interstage Management Console. By specifying the definition name in the JNDI lookup method and executing it, the application obtains the management target object and executes processing.



Note

Notes about setting the definition name

When the definition name is set, the same name, which is unique, must be entered for all of the following:

- connector1.0 specification resource adapter definition name
- connector1.5 specification ConnectionFactory definition name
- connector1.5 specification management target object definition name

You should set names that are unique, so that the connector1.0-specification resource adapter definition name and the connector1.5 specification resource adapter resource adapter name are not the same. For details about the character types that can be used, refer to the Interstage Management Console Help.

21.2 Stopping/Starting the Resource Adapter

Depending on the resource adapter, resource adapter start and stop processing that is synchronized with the IJServer may be executed.

In the connector1.5 specification resource adapter, the ResourceAdapter class can be defined in the deployment descriptor file (ra.xml). The start, stop, endpointActivation, and endpointDeactivation methods are implemented in the ResourceAdapter class.

If the resource adapter defined in the ResourceAdapter class is deployed, each method is called and processing for the receiving of asynchronous messages starts or stops when the IJServer is started or stopped (when deploy, undeploy, or reactivate is executed when the HotDeploy functionality is used).

The timing for the execution of the ResourceAdapter class method is shown in the table below. For details about the timing of the activation and deactivation of each module, refer to "HotDeploy Function of J2EE".

| Method name | Call timing |
|----------------------|--|
| start | When the IJServer starts up When classes contained in RAR files are auto-reloaded When one of the following applies, use the HotDeploy functionality When the RAR file (or EAR file that contains the RAR file) is deployed to the IJServer that starts up When the RAR file (or EAR file that contains the RAR file) is activated in the IJServer that starts up |
| stop | When the IJServer stops When classes contained in RAR files are auto-reloaded When one of the following applies, use the HotDeploy functionality When the RAR file (or EAR file that contains the RAR file) is undeployed from the IJServer that starts up When the RAR file (or EAR file that contains the RAR file) is deactivated in the IJServer that starts up |
| endpointActivation | When the IJServer starts up When the Message-driven Bean classes in the ejb-jar file that receive the resource adapter are auto-reloaded When one of the following applies, use the HotDeploy functionality When the Message-driven Bean in the ejb-jar file (or EAR file that contains the ejb-jar file) that receives the resource adapter is deployed to the IJServer that starts up When the Message-driven Bean in the ejb-jar file (or EAR file that contains the ejb-jar file) that receives the resource adapter is activated in the IJServer that starts up |
| endpointDeactivation | When the IJServer stops When the Message-driven Bean classes in the ejb-jar file that receive the resource adapter are auto-reloaded When one of the following applies, use the HotDeploy functionality |

| Method name | Call timing |
|-------------|---|
| | When the Message-driven Bean in the ejb-jar file (or EAR file that contains the ejb-jar file) that receives the resource adapter is undeployed from the IIServer that starts up |
| | When the Message-driven Bean in the ejb-jar file (or EAR file that contains the ejb-jar file) that receives the resource adapter is deactivated in the IIServer that starts up |



Note

Notes about redeployment

To perform redeployment, redefine the ConnectionFactory and management target object definition names.

21.3 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.

Windows32/64

```
C:\Interstage\J2EE\etc\JCA\jca.properties
```

Solaris32/64 Linux32/64

```
/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 |



Example

To set the time-out period to 300 seconds, change it as follows.

```
idle.resource.threshold=300
```


21.4 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.

| 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

Note when Transaction Function is Used

To use XA transactions, do as follows: From the Interstage Management Console, select [WorkUnit] > [IJSERVER name] > [Settings] > [EJB Container Settings] > [Use Distributed Transaction for EJB applications] and select "Yes." However, this cannot be selected when the IJSERVER type is "Run Web applications and EJB applications on the same JavaVM".

21.5 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.

| 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 authorized 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 |

21.6 Work Management

Interstage connector supports Work management functionality for the execution of Work (such as the calling of applications, network endpoint monitoring, and input data processing) synchronously or asynchronously. If the Work management functionality is used, applications that use the thread control mechanism provided in Interstage can be set up. This functionality can be used in the connector1.5 specification-compliant resource adapter.

The resource adapter registers the Work for the implementation of the WorkManager class provided in IJServer using a method such as scheduleWork. For details, refer to the connector1.5 specification.

Number of Threads Executed Simultaneously that can be Allocated to Work

One thread is allocated to one Work. The Minimum, the Maximum and the Idle Timeout can be specified as the tuning parameter for the number of threads that can be executed simultaneously.

Set the value in the following file to modify the number of threads executed simultaneously that can be allocated to Work.

Windows32/64

```
C:\Interstage\J2EE\etc\JCA\jca.properties
```

Solaris32/64 Linux32/64

```
/opt/FJSVj2ee/etc/jca/jca.properties
```

Set the following properties in the above file. If a parameter is not specified, and the value that is set is outside the range that can be specified, the default value is used.

| Tuning parameter | Property name | Meaning |
|------------------|---------------------|---|
| Minimum | min-thread-size | This is the minimum number of threads being executed simultaneously that can be allocated to Work. Specify a number from 1 to 2147483647. The default is '1'. |
| Maximum | max-thread-size | This is the maximum number of threads being executed simultaneously that can be allocated to Work. Specify a number from 1 to 2147483647. The default is '64'. If a value less than Minimum is specified, the Minimum value is set as Maximum. |
| Idle Timeout | thread-idle-timeout | This is the idle timeout (in seconds). Threads returned to the pool are destroyed if they are not used before the specified time is exceeded. The number for the initial start threads is not destroyed, however. Specify a number from 0 to 2147483647. The default is '600'. If '0' is specified, there is no timeout. |

Example

In the example shown below, the minimum number of threads being executed simultaneously that can be allocated to Work is 2, the maximum number of threads being executed simultaneously that can be allocated to Work is 10, and the idle timeout is 1000.

```
min-thread-size=2
max-thread-size=10
thread-idle-timeout=1000
```

Part 8 Appendixes

| |
|--|
| |
|--|

Appendix A JDK, JRE and FJVM

Refer to "JDK/JRE Tuning" in the *Tuning Guide* for a detailed explanation of JDK/JRE and Java VM, including the FJVM.

Appendix B Linkage with Oracle Real Application Clusters

This product supports linkage with the Oracle option 'Oracle Real Application Clusters' (hereafter referred to as Oracle RAC).

Environment Settings

Configure the following settings to enable linkage with Oracle RAC.

- Oracle Settings
- Interstage Settings

Refer to the following notes to enable linkage with Oracle RAC:

- Notes on Linkage with Oracle RAC

B.1 Oracle Settings

In order to create an environment in which Oracle RAC can be used, settings must be made on the Oracle side.

The Oracle settings depend on whether or not load balancing is used.

- [Without Load Balancing](#)
- [With Load Balancing](#)

Without Load Balancing

Determine a default connection node for each application (JDBC data source) such that applications only connect to a different node if an error occurs in the default connection. Make settings in the listener.ora file of the Oracle server based on the example below.



Example

For example, in listener.ora of Oracle server 1:

```
LISTENER =
  (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = TCP)(HOST = server1)(PORT = 1521))
  )
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = smp1)
      (SID_NAME = smp11)
      (ORACLE_HOME = /opt/app/oracle/product/9.2.0)
    )
  )
```

When using an oci driver

When using an oci driver, make settings in the tnsnames.ora file of the Oracle client (Interstage) based on the example below. This is not required when using a thin driver.



Example

Example in tnsnames.ora of the client (Interstage):

```
SAMPLE.WORLD =
  (DESCRIPTION = (ENABLE = BROKEN)
    (FAILOVER = ON)
    (ADDRESS = (PROTOCOL = TCP)(HOST = server1)(PORT = 1521))
    (ADDRESS = (PROTOCOL = TCP)(HOST = server2)(PORT = 1521))
```

```
(ADDRESS = (PROTOCOL = TCP)(HOST = server3)(PORT = 1521))
(CONNECT_DATA = (SERVICE_NAME = smp1))
)
```

With Load Balancing

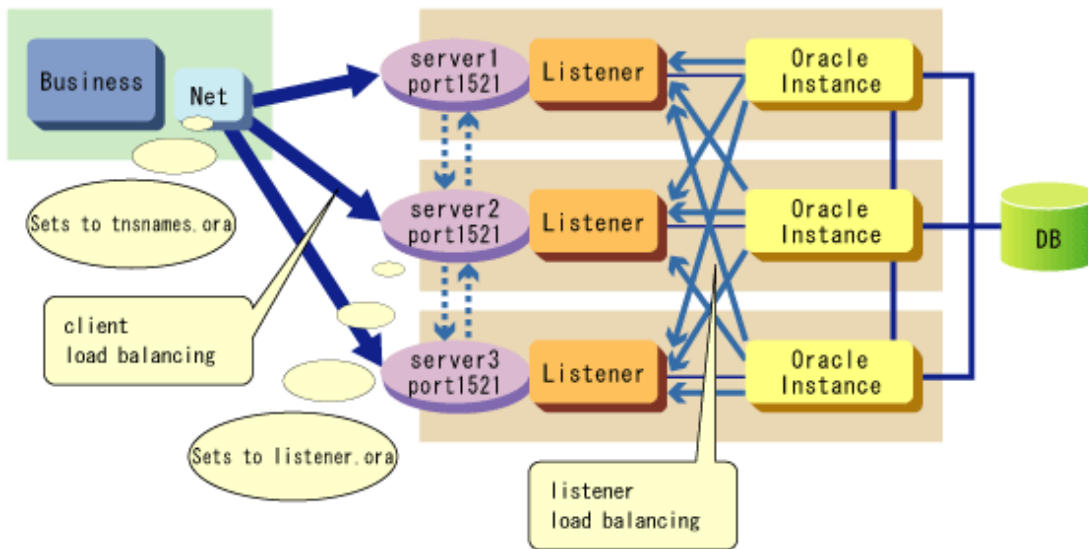
The load balancing function distributes the load from normal operations. There are two types of load balancing: "client load balancing" and "listener load balancing":

client load balancing

Distributes the load from the application to the Oracle listener.

listener load balancing

Distributes the load from the listener to the Oracle instance.



Example

Examples of settings for client load balancing and listener load balancing are given below. Base the settings you make on these examples.

Example of Oracle Server 1 initialization parameter (excerpt):

```
local_listener = "(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(Host = server1)(Port = 1521)))"
remote_listener = "(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(Host = server2)
(Port = 1521))(ADDRESS = (PROTOCOL = TCP)(Host = server3)(Port = 1521)))"
```

Example in listener.ora of Oracle server 1:

```
LISTENER =
  (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = TCP)(Host = server1)(Port = 1521))
  )
```

When Using an OCI Driver

When using an oci driver, make settings in the tnsnames.ora file of the Oracle client (Interstage) based on the example below. This is not required when using a thin driver.

Example

Example in tnsnames.ora of the client (Interstage):

```
SAMPLE.WORLD =
  (DESCRIPTION = (ENABLE = BROKEN)
    (LOAD_BALANCE = ON)
    (FAILOVER = ON)
    (ADDRESS = (PROTOCOL = TCP)(HOST = server1)(PORT = 1521))
    (ADDRESS = (PROTOCOL = TCP)(HOST = server2)(PORT = 1521))
    (ADDRESS = (PROTOCOL = TCP)(HOST = server3)(PORT = 1521))
    (CONNECT_DATA = (SERVICE_NAME = smp1))
  )
```

B.2 Interstage Settings

The settings differ depending on whether a thin driver or OCI driver is used.

Configuration

Using a thin driver

Register the JDBC data sources by specifying "Use" in the "Use RAC" setting. On the Interstage management console, click [Resources[> [JDBC]> [Create new]. Alternatively, register the JDBC data resources in the same way using the resource subcommand of the *isj2eadmin* command.

Specify the server URL of Oracle RAC in the Server URL. An example of server URL specification is shown as follows:

Example

Example of server URL specification:

```
jdbc:oracle:thin:@(DESCRIPTION=
  (ENABLE=BROKEN)
  (ADDRESS_LIST=
    (FAILOVER=ON)
    (LOAD_BALANCE=ON)
    (ADDRESS=
      (PROTOCOL=tcp)
      (HOST=host1)
      (PORT=1521))
    (ADDRESS=
      (PROTOCOL=tcp)
      (HOST=host2)
      (PORT=1521)))
  (CONNECT_DATA=
    (SERVICE_NAME=service_name)))
```

Using an OCI driver

Register the JDBC data sources by specifying "Use" in the "Use RAC" setting. On the Interstage management console, click [Resources[> [JDBC]> [Create new]. Alternatively, register the JDBC data resources in the same way using the resource subcommand of the *isj2eadmin* command.

Specify the server URL of Oracle RAC in the Server URL. An example of the server URL specification is shown as follows:

Example

Example of server URL specification

```
jdbc:oracle:oci:@(DESCRIPTION=
  (ENABLE=BROKEN)
  (ADDRESS=
    (PROTOCOL=tcp)
    (HOST=cluster_alias)
    (PORT=1521))
  (CONNECT_DATA=
    (SERVICE_NAME=service_name)))
```

Configuring the server URL

When the Interstage management console is used, a prototype corresponding to the thin driver or the OCI driver is output when you click the [Create template] button, located on [Server URL], at the time of data source definition. Replace the italicized characters in the Item column of the following table with values appropriate for the environment they are to be used in.

Refer to the Oracle manual on how to set the server URL for other servers:

| Item | Contents |
|--|---|
| FAILOVER= <i>ON</i> | Set "OFF" if the failover function is not to be used. Example: FAILOVER=OFF |
| LOAD_BALANCE= <i>ON</i> | Set "OFF" if the load balance function is not to be used. Example: LOAD_BALANCE=OFF |
| PROTOCOL= <i>tcp</i> | Only 'tcp' is supported by the thin driver. Specify 'ipc' if the 'ipc' protocol is used with the OCI driver. Example: PROTOCOL=ipc |
| HOST= <i>host</i> or HOST= <i>cluster_alias</i> | Enter the host name of the DB server according to the environment it is to be used in (cluster name for the OCI driver). Example: HOST=shop001 |
| PORT= <i>1521</i> | Change this value if the Oracle port number is something other than 1521. Normally, there is no need to change the value. |
| SERVICE_NAME= <i>service_name</i> | Set the service name according to the environment being used on the Interstage management console. When the OCI driver is selected, set the net service name registered in the 'tnsnames.ora' files. Example: SERVICE_NAME=banking |

Example

Example: Configuring the server URL

An example of how to configure the server URL is shown as follows:

When OracleDB1 is the operation DB server and OracleDB2 is the standby DB server, and only the failover function is used:

```
jdbc:oracle:thin:@(DESCRIPTION=
  (ENABLE=BROKEN)
  (ADDRESS_LIST=
    (FAILOVER=ON)
    (LOAD_BALANCE=OFF)
    (ADDRESS=
      (PROTOCOL=tcp)(HOST=OracleDB1)(PORT=1521))
```

```
(ADDRESS=
  (PROTOCOL=tcp)(HOST=OracleDB2)(PORT=1521))
(CONNECT_DATA=
  (SERVICE_NAME=service_name))
```

The following example illustrates the use of the load balance function on OracleDB1, OracleDB2, and OracleDB3:

```
jdbc:oracle:thin:@(DESCRIPTION=
  (ENABLE=BROKEN)
  (ADDRESS_LIST=
    (FAILOVER=ON)
    (LOAD_BALANCE=ON)
    (ADDRESS=
      (PROTOCOL=tcp)(HOST=OracleDB1)(PORT=1521))
    (ADDRESS=
      (PROTOCOL=tcp)(HOST=OracleDB2)(PORT=1521))
    (ADDRESS=
      (PROTOCOL=tcp)(HOST=OracleDB3)(PORT=1521)))
  (CONNECT_DATA=
    (SERVICE_NAME=service_name)))
```



Note

- When the OCI driver is selected, the 'listener.ora' files on the Oracle side and the 'tnsnames.ora' files on the client side (Interstage) must be edited. For details, refer to Oracle manuals.
- Any carriage returns in the [Server URL] text area is ignored at the time of data source creation.

Other Settings

The high-speed connection failover function is used when "Use the Oracle connection pooling" is set for the data source type in the data source definitions for using RAC in the Oracle 10g or later RAC environment. The function is not used if "Use the Interstage connection pooling" is selected as the data source type. For details on the high-speed connection failover function, refer to the Oracle manuals.

The 'ons.jar' files must be set in the class path when "Use the Oracle connection pooling" is configured as the data source type in the data source definitions for using RAC. The 'ons.jar' files are located under the directory where Oracle is installed (ORACLE_HOME) as follows:

- ORACLE_HOME\opmn\lib\ons.jar

Configure the value of ORACLE_HOME (the directory where Oracle is installed) on the server installing Interstage as follows in the JavaVM option:

- -Doracle.ons.oraclehome=ORACLE_HOME

From the Interstage management console, the JavaVM option can be set in the J2EE property specified in the system environment settings and the IJServer environment settings.

When the environment variables are set for both the system item and the IJServer item, the value set for The IJServer becomes valid. If the above setting is not made property in J2EE, the DB connection test cannot be performed correctly.

| Definition item | Definition contents |
|-----------------|---|
| JavaVM option | Set the option to be specified in the Java command when the DB connection test function is used or The IJServer is operating. Specify a character string of 4096 bytes or less. <ul style="list-style-type: none"> - When a null is included in an option, enclose the option with "" (double quotation marks) in the specification. |

| Definition item | Definition contents |
|-----------------|---|
| | <ul style="list-style-type: none"> - To specify multiple options, use a null as a separator and enter options consecutively. If the entire section is enclosed by "" (double quotation marks), the entire section is seen as one option. Use "" (double quotation marks) only when options containing nulls are specified. - No carriage returns may be included. |

Refer to the Oracle manuals to configure the Oracle Notification Service on the Interstage side and activate the ONS demon.

B.3 Notes on Linkage with Oracle RAC

- When a database server failover occurs

When Oracle RAC or the server fails and failover occurs operations degenerate for continued Oracle RAC operation. In such cases, all connections to the failed server are distributed to a functioning Oracle system.

At this time, a higher number of connections than normal are made to the functioning servers and connections are pooled. When the failed server is restored, operations may not be returned to the restored server because the required number of connections are already pooled. If this happens, sever the connection using the *ijstune jdbc* command or restart the IJServer to re-establish connections correctly.

- When making Interstage settings

It is recommended to select the Use option for the Interstage Automatic Re-connect function. By default, the 'Use' option is selected. For more details, refer to Tuning of IJServer in 'JNDI', Tuning J2EE Applications of the Tuning Guide.

- Compatibility with non-Java applications

No special definitions are required in Interstage to ensure compatibility with non-Java applications. Settings are required in Oracle RAC only.

- When using the Database Linkage Service Windows32/64 Solaris32 Linux32/64

Oracle RAC cannot be used with the Database Linkage Service (for distributed transactions).

- Setting the Interstage Management Console

- The syntax of the entered server URL can be checked during the execution of [DB connection test] at the time of data source creation, or by pressing the [DB connection test] button on the environment setting screen.

Note that when the load balance/failover function is set, in order to access and obtain a connection with a DB server on the address list, if at least one DB server defined is connected properly, no error is issued. Therefore, not all the host names can be confirmed this way.

- When the data sources are created by selecting [Use Oracle connection pooling], reconnection at the time of an error generation by selecting [work unit] > [Environment settings] > [DB connection settings] on the Interstage management console is not valid.
- When the Oracle Notification Service setting is incorrect, an error is issued from ONS to the container log at the time of the IJServer activation. However, the IJServer will still activate successfully. Refer to the Oracle manuals and "Exception Information Output When J2EE is Used" in the "Messages". After reviewing the settings, reactivate the IJServer.

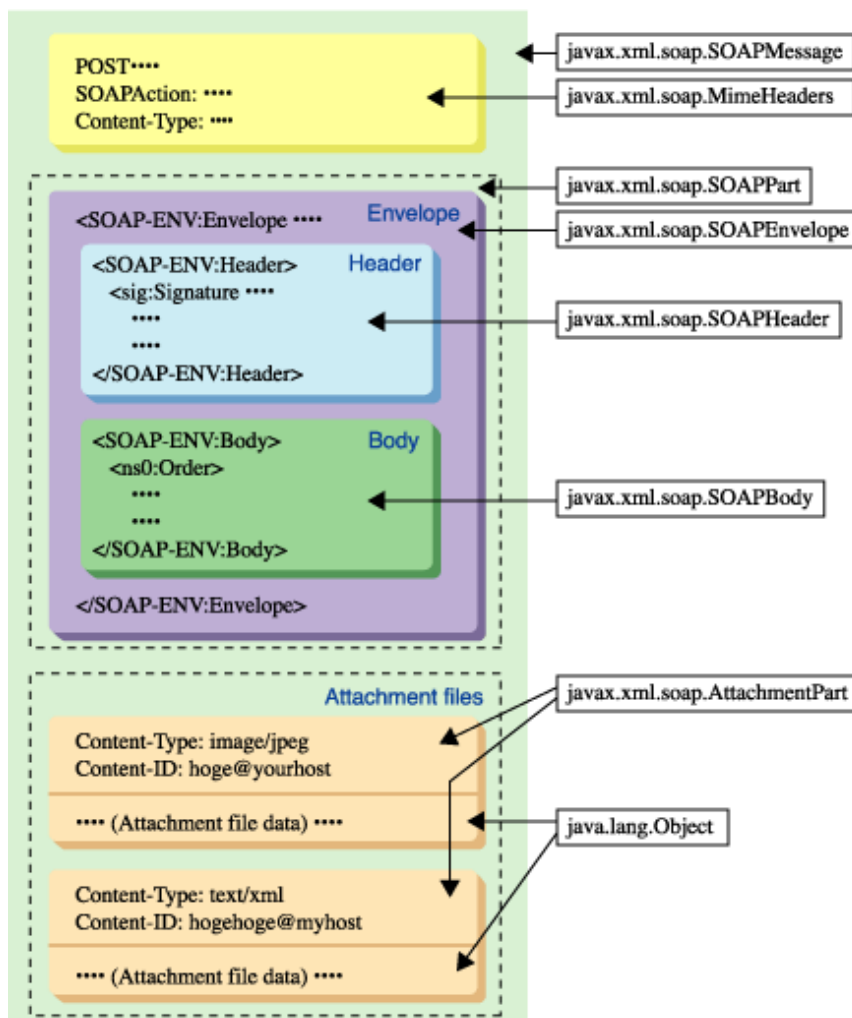
Appendix C Low-level Processing of SOAP Messages

This appendix explains the low-level processing for the SOAP messages that use advanced APIs to access the SOAP messages directly.

Note

- If the SOAP packet data size is large, a large amount of memory will be consumed when messages are received, bringing about the possibility of insufficient memory or an excessive processing time.
- As an estimate, investigate the use of an attachment file if the communication data size exceeds several 100Kbytes (including XML tags).
- However, the amount of memory consumed will differ greatly depending on the content and format of the data, regardless of its size, and the amount of high volume processing that can be done simultaneously as well as performance requirements, will also differ depending on the system. It is recommended that you consider the content and format of the actual business data, and the conditions under which it will be communicated, using the above size estimation as a guide.

C.1 Structure of SOAP Message



The above figure shows the structure of a SOAP message having attachment files. SAAJ-API maps the individual elements of the SOAP message to the Java classes shown at right in the figure.

C.2 Creating a SOAP Message

To create new components for the SOAP envelope, use a SOAPFactory object.

Get SOAPFactory objects by using the newInstance class method of the SOAPFactory class.

```
import javax.xml.soap.*;
.....
MessageFactory mf = MessageFactory.newInstance();
SOAPMessage msg = mf.createMessage();
SOAPPart part = msg.getSOAPPart();
SOAPEnvelope env = part.getEnvelope();
.....
```

An application that uses the reliable messaging function can generate a MessageFactory object from the ProviderConnection object as shown below. See Chapter 7 Installing the Reliable Messaging Function for information on the application that uses the reliable messaging function.

```
import javax.xml.soap.*;
.....
MessageFactory mf = MessageFactory.newInstance();
SOAPMessage msg = mf.createMessage();
SOAPPart part = msg.getSOAPPart();
SOAPEnvelope env = part.getEnvelope();
.....
```

C.3 SOAP Envelope Processing

The SOAPEnvelope object is an object that represents a SOAP envelope. One SOAP message always includes one SOAP envelope.

A SOAPEnvelope object can be obtained using the SOAPPart.getEnvelope method. The SOAPEnvelope object API is used to operate the SOAP header or SOAP body directly. The SOAP envelope includes zero or one SOAP header and one SOAP body.

```
import javax.xml.soap.*;
.....
MessageFactory mf = MessageFactory.newInstance();
SOAPMessage msg = mf.createMessage();
SOAPPart part = msg.getSOAPPart();
SOAPEnvelope env = part.getEnvelope();
// Processing when SOAPHeader is not used
SOAPHeader header = env.getHeader();
header.detachNode();
// Processing when SOAPHeader is used
// SOAPHeader header = env.getHeader();
// SOAPHeaderElement helm = header.addHeaderElement(
//     env.createName( "Header1", "ns1", "urn:Sample" ));
// helm.addNamespaceDeclaration("ns1", "urn:Sample");
// SOAPBody setting
SOAPBody body = env.getBody();
SOAPBodyElement belm = body.addBodyElement(
    env.createName( "Body1", "ns1", "urn:Sample" ));
belm.addNamespaceDeclaration("ns1", "urn:Sample");
.....
```

If the SOAP header is not to be used, call the SOAPHeader.detachNode method explicitly or do nothing. If nothing is done, a null header (<xxx:Header/>) is output.

The SOAPHeader, SOAPHeaderElement, SOAPBody, and SOAPBodyElement interfaces each inherit the SOAPElement interface and therefore can continue to set or obtain subelements by using the API of the SOAPElement interface.

Note

- If a required name space declaration is not performed in or under the SOAPEnvelope object, a required name space declaration is automatically added to the SOAP envelope data to be sent or sent back when it is sent or sent back.
- When adding a SOAP digital signature or using XML encryption, be sure to perform a required name space declaration in or under the SOAPEnvelope object.

For processing the SOAP envelope, the API of the SOAPEnvelope interface can be used to directly manipulate the SOAPHeader and SOAPBody objects. In addition, XML data can be used to set the XML object as is in the SOAP envelope or the contents of the SOAP envelope can be extracted as XML data. For this operation, use the setContent/getContent method of the SOAPPart object.

```
import javax.xml.soap.*;
import javax.xml.transform.Source;
import javax.xml.transform.sax.SAXSource;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamSource;
.....
MessageFactory mf = MessageFactory.newInstance();
SOAPMessage msg = mf.createMessage();
SOAPPart part = msg.getSOAPPart();
// Create a SAXSource object.
SAXSource source = .....;
// Create a DOMSource object.
DOMSource source = .....;
// Create a StreamSource object.
StreamSource source = .....;
// Create SOAPMessage from the Source object.
part.setContent(source);
.....
// Obtain SOAPMessage as a Source object.
Source source = part.getContent();
.....
```

SAXSource, DOMSource, and StreamSource each are classes in which a Source interface is installed. An object in which a Source interface is installed can be defined as a SOAP envelope by the SOAPPart.setContent method. Similarly, the contents of a SOAP envelope can be obtained as a Source object by the SOAPPart getContent method. See the sample for a usage example.

C.4 Fault Processing

The application that sends a SOAP message may receive a fault message as return information from the server system. SAAJ-API represents a fault as a SOAPFault object. The SOAPFault object is set as a subelement of the SOAPBody object in a SOAPMessage object.

The application that has caused a fault generates a SOAPFault object using the SOAPBody.addFault method. The application that receives a fault can take out the SOAPFault object using the SOAPBody.getFault method. The sender application can use the SOAPBody.hasFault method to check whether a fault is included in the SOAPMessage object received.

The SOAPFault object retains the following types of information, which can be set or obtained by applications:

To manipulate the Fault code, Fault description and Fault actor, be sure to use the set methods provided by the SOAPFault class (setFaultCode, setFaultActor or setFaultString). Information cannot be set using the SOAPElement.addTextNode method.

- Fault code (java.lang.String)
- Fault description (java.lang.String)
- Fault actor (java.lang.String)
- Fault detail (java.xml.soap.Detail)

```
import javax.xml.soap.*;
import java.util.Iterator;
.....
```

```

MessageFactory mf = MessageFactory.newInstance();
SOAPMessage msg = mf.createMessage();
SOAPPart part = msg.getSOAPPart();
SOAPEnvelope env = part.getEnvelope();
env.getHeader().detachNode();
SOAPBody body = env.getBody();
.....
// Set SOAPFault.
SOAPFault fault = body.addFault();
fault.setFaultActor(Fault);
fault.setFaultCode(Fault);
fault.setFaultString(Fault);
Detail detail = fault.addDetail();
DetailEntry entry = detail.addDetailEntry( ..... );
entry.addChildElement(...);
.....
SOAPMessage reply = .....;
// Check whether a fault occurred.
env = reply.getSOAPPart().getEnvelope();
body = env.getBody();
if( body.hasFault() ) {
// Fault occurrence
    fault = body.getFault();
    String faultActor = fault.getFaultActor();
    String faultCode = fault.getFaultCode();
    String faultString = fault.getFaultString();
    detail = fault.getDetail();
    Iterator it = detail.getDetailEntries();
    while( it.hasNext() ){
        entry = (DetailEntry)it.next();
        .....
    }
}
else{
// Normal processing
    .....
}

```

The SOAPFault interface inherits the SOAPBodyElement interface that represents a subelement of SOAPBody. SOAPFault, SOAPFaultElement, Detail, and DetailEntry interfaces each inherit the SOAPElement interface and therefore permit fault information to be assembled the same as for assembling normal SOAP messages.

C.4.1 Analyzing Fault Information

If an error occurs during invocation processing of the receiver application (application in the server system), fault information is set in the reply message received by the sender application (application in the client system). The SOAPBody.hasFault method is used to check whether fault information is set in the reply message received. If the return value from the method is 'true,' fault information is set in the reply message.

Fault information is set as a sub-element of the SOAP body the same as with a normal SOAP message. Fault information is represented as a SOAPFault object and can be extracted using the SOAPBody.getFault method. The following shows a program for receiving a reply message and extracting fault information.

```

import javax.xml.soap.*;
.....
// Receive response data.
SOAPMessage response = conn.call( msg, endPoint );
SOAPEnvelope env = response.getSOAPPart().getEnvelope();
// Extract the SOAPBody
SOAPBody body = env.getBody();
// Check whether fault information is set.
if( body.hasFault() ) {

```



```

// Extract the fault information.
SOAPFault fault = body.getFault();
// Extract the fault code.
String faultCode = fault.getFaultCode();
// Extract the fault description.
String faultString = fault.getFaultString();
// Extract the fault detail.
Detail detail = fault.getDetail();
Iterator it = detail.getDetailEntries();
.....
}
else{
// Fault information is not set (normal).
.....
}

```

Fault information includes the following items:

- Fault code
- Fault description
- Fault actor
- Fault detail

| Information type | Description |
|-------------------|--|
| Fault code | Classification of fault. One of the following character strings is set: <ul style="list-style-type: none"> - "Server" - "Client" See the 'Fault code classification' shown below for the meanings of these character strings. |
| Fault description | Character strings that describe the fault |
| Fault actor | URI that indicates the error location (such as the receiver application) |
| Fault detail | Error information that is specific to the Web service application is set. |

Extracting the Fault Code

Use the following method of the SOAPFault object to extract the fault code:

```
java.lang.String getFaultCode()
```

Fault information includes the following items:

- Fault code
- Fault description

| OtherFault code | Description |
|-----------------|--|
| Server | Indicates that a problem was detected in Web Service Container or during processing by the receiver application. Returned to indicate that, for instance, Web Service Container contains a setting error. |
| Client | Indicates that the sender application detected an error in the message it sent. If this fault code is returned to the sender application, check the message sent to the receiver application. |
| VersionMismatch | This code indicates that the Web service does not support the SOAP version for the received message. |
| MustUnderstand | This code indicates that the Web service failed to process a mandatory child element of the SOAP header in the received message. |

| OtherFault code | Description |
|------------------------|---|
| Other fault code value | A fault code other than the above may be returned if: <ul style="list-style-type: none"> - The Web service has defined its own fault code. |

Extracting the Fault Description

To extract the fault description, use the following method of the SOAPFault object:

```
java.lang.String getFaultString()
```

Extracting the Fault Actor

To extract the fault actor, use the following method of the SOAPFault object:

```
java.lang.String getFaultActor()
```

Extracting the Fault Detail

The fault detail represents the information items specific to a Web service consisting of an arbitrary number of fault detail items.

To extract the fault detail, use the following method of the SOAPFault object:

```
javax.xml.soap.Detail getDetail()
```

The individual detail items included in the fault detail are represented as javax.xml.soap.DetailEntry objects and can be extracted by using the following method of the Detail object:

```
java.util.Iterator getDetailEntries()
```

The individual fault detail items are retained as javax.xml.soap.DetailEntry objects in the java.util.Iterator object. The following shows a program for extracting fault detail items.

```
import javax.xml.soap.*;
.....
if( body.hasFault() ) {
    SOAPFault fault = body.getFault();
    String faultCode = fault.getFaultCode();
    String faultString = fault.getFaultString();
    // Extract the fault information
    Detail detail = fault.getDetail();
    // Extract the fault detail.
    Iterator it = detail.getDetailEntries();
    while(it.hasNext() ) {
        DetailEntry entry = (DetailEntry)it.next();
        .....
    }
}
else{
    .....
}
```

The javax.xml.soap.DetailEntry interface inherits the javax.xml.soap.SOAPElement interface. Therefore, the individual detail items included in the fault detail can be operated as pure SOAPElement objects.



Note

When using the SOAPElement.getChildElements(Name name) method, be sure to get the child elements by specifying both a prefix and a URI for the Name object parameter.

Appendix D Executing Sample Applications

This appendix explains the sample applications that can be executed in Interstage Application Server.

D.1 The Sample Environment

This section explains the following topics:

- Overview of the Sample Environment and Explanation of Functions
- Advance Preparation
- Starting the Sample Environment
- Sample Applications that are Offered

D.1.1 Overview of the Sample Environment and Explanation of Functions

The sample environment is a GUI tool for running sample applications that use J2EE functions and Framework sample applications. Use the sample environment in a standalone environment.

This tool can also be used to display a guide containing an overview of each sample application, the environment setup method, and the execution method.

D.1.2 Advance Preparation

Before starting the sample environment, it is recommended that the following preparation tasks are executed.

Enterprise Edition

- Running the Web server so that it is linked with Interstage

After Interstage is installed, the settings are such that the Web server does not start up and link with Interstage. If the machine is restarted immediately after the installation, the Web server is not started automatically. For this reason, if the sample environment explained below is started immediately after the restart, the sample environment window is not displayed in the browser.

To run the Web server so that it is linked with Interstage, configure the settings according to the following procedure.

1. Select [System] from the Interstage Management Console.
2. Enter the following information in the [Update System Settings] page, and then click [Update].

Detailed Settings

| Item name | Input value |
|------------|------------------------|
| Web server | Select [Synchronized]. |

- Setting up Event Service

Before JMS can be used, "Event Service" must be registered as the Interstage configuration service.

Using the Interstage Management Console, select [System]. In the [View System Status] window, click details [Show]. If "Event Service" has not been registered in [Interstage Component Services], register it according to the following procedure:

1. Using the Interstage Management Console, select [System].
2. Enter the following information in Detailed Settings of the [Update System Settings] page, and then click [Update].

Detailed Settings

| Item name | Input value |
|------------------------|--------------|
| Event Service Settings | Select [Yes] |

Standard-J Edition

- Setting up Event Service

Before JMS can be used, "Event Service" must be registered as the Interstage configuration service.

Using the Interstage Management Console, select [System]. In the [View System Status] window, click details [Show]. If "Event Service" has not been registered in [Interstage Component Services], register it according to the following procedure:

1. Using the Interstage Management Console, select [System].
2. Enter the following information in Detailed Settings of the [Update System Settings] page, and then click [Update].

Detailed Settings

| Item name | Input value |
|--------------|--------------|
| JMS Settings | Select [Yes] |

- Setting the class path for the JDBC driver in J2EE properties

To use a sample application that uses an Oracle or SQL Server database, the class path for the JDBC driver must be set in the J2EE properties.

Set the J2EE properties according to the following procedure.

1. Using the Interstage Management Console, select [System].
2. Enter the following information in the [Update System Settings] page, and then click [Update].

Detailed Settings

| Item name | Input value |
|-----------|--|
| Classpath | Class path of the JDBC driver that is used |



Note

Set the class path for the JDBC driver as the class path. For detailed information about the environment settings, refer to the appropriate JDBC driver manual.

D.1.3 Starting the Sample Environment

Start the sample environment according to the following operation. If it has already been set up, the Web browser starts up, and the sample environment initial window is displayed.

Windows32/64

Click the [Start] menu, and then click [Programs] > [Interstage Application Server] > [Deploy and run Sample Applications] to start the sample environment.

After Interstage is started according to the above procedure, the following query is output. Respond to the query to launch the setup of the sample environment.

The following query is output immediately after the installation. Any time after that, the query contents are different to those shown below, depending on the environment status.

```
Set up Sample Integrated Environment Guide and sample applications.
1. Sample Integrated Environment Guide preparation
Do you want to deploy the Sample Integrated Environment Guide to the
SampleGuideServer? (default:y) [y,n]
```

If "y" is entered for the above query, an IJServer WorkUnit with the name "SampleGuideServer" is created, the application of the Sample Integrated Environment Guide is deployed and started, and the following query is displayed:

```
Create the IJServer WorkUnit. (IJServer WorkUnit name: SampleGuideServer)
Deployment information
Web application name:SampleGuideServer
Sample Integrated Environment Guide is being deployed. Deployed
C:\Interstage\sample\integrate\SampleGuideServer.war
Starting the IJServer WorkUnit because the Sample Integrated Environment Guide
was deployed for a stopped IJServer WorkUnit.
```

```
2. Sample application preparation
Specify the name of the IJServer WorkUnit used for sample application
deployment. (default:SampleServer)[?,q]
```

If nothing is entered for the above query (the "default" value is selected) and the Enter key is pressed, an IJServer WorkUnit with the name "SampleGuideServer" is created, the sample application is deployed and started, and the sample environment initial window is displayed in the Web browser. The message that is output is shown below.

To change the IJServer WorkUnit name, specify any IJServer WorkUnit name when the above query is made.

```
Refer to the Sample Integrated Environment Guide for details of sample applications.
Deployment information
IJServer WorkUnit name: SampleServer
Web application name: HelloServlet
Web application name: meetingroom
Sample Application is being deployed. Deployed:
C:\Interstage\sample\integrate\ja\j2ee\servlet\helloservlet\HelloServlet.war
Sample Application is being deployed. Deployed:
C:\Interstage\sample\integrate\ja\framework\meetingroom\meetingroom.war
Starting the IJServer WorkUnit because the Sample Integrated Environment
Guide was deployed for a stopped IJServer WorkUnit.
```

If "n" is entered for the first query, setup processing is interrupted. The Sample Integrated Environment Guide and application deployment are not executed. The message that is output is shown below:

```
Processing was terminated because the Sample Integrated Environment Guide
was not deployed. Execute again, and deploy the Sample Integrated
Environment Guide.
Press any key to continue . . .
```

Solaris32/64 **Linux32/64**

Execute the following.

```
/opt/FJSVisspl/integrate/sampleexec.sh
```

If the above is executed, the following query is output. Respond to the query to launch the setup of the sample environment. Only the superuser can execute the sampleexec.sh.

```
Set up Sample Integrated Environment Guide and sample applications.
Enter the user name that starts the Sample Integrated Environment
Guide and sample applications.(default:root)[?,q]
Sample Integrated Environment Guide preparation
Do you want to deploy the Sample Integrated Environment Guide to the
SampleGuideServer? (default:y) [y,n]
```

After entering the user name that starts the Sample Integrated Environment Guide and sample applications, if "y" is entered for the above query, an IJServer WorkUnit with the name "SampleGuideServer" is created, the application of the Sample Integrated Environment Guide is deployed and started, and the following query is displayed. The message that is output is shown below:

```
Create the IJServer WorkUnit. (IJServer WorkUnit name: SampleGuideServer)
Deployment information
Web application name:SampleGuideServer
Sample Application is being deployed. Deployed: /opt/FJSVisspl/integrate/SampleGuideServer.war
Starting the IJServer WorkUnit because the Sample Integrated Environment
```

Guide was deployed for a stopped IJServer WorkUnit.

2. Sample application preparation

Specify the name of the IJServer WorkUnit used for sample application deployment.

```
(default:SampleServer)[?,q]
```

If nothing is entered for the above query (the "default" value is selected) and the Enter key is pressed, an IJServer WorkUnit with the name "SampleGuideServer" is created, the sample application is deployed and started, and the sample environment initial window is displayed in the Web browser. The message that is output is shown below.

To change the IJServer WorkUnit name, specify any IJServer WorkUnit name when the above query is made.

Refer to the Sample Integrated Environment Guide for details of sample applications.

Deployment information

IJServer WorkUnit name:SampleServer

Web application name:HelloServlet

Web application name:meetingroom

Sample Application is being deployed. Deployed

/opt/FJSVisspl/integrate/ja/j2ee/servlet/helloservlet/HelloServlet.war

Sample Application is being deployed. Deployed: /opt/FJSVisspl/integrate/ja/framework/meetingroom/meetingroom.war

Starting the IJServer WorkUnit because the Sample Application was deployed for a stopped IJServer WorkUnit.

To access the deployed Sample Integrated Environment Guide, specify the following URL on a Web browser.

(hostname : Host Name,port : Port No)

http://hostname:port/SampleGuideServer/

If "n" is entered for the first query, setup processing is interrupted. The Sample Integrated Environment Guide and application deployment are not executed. The message that is output is shown below:

Processing was terminated because the Sample Integrated Environment Guide was not deployed.

Execute again, and deploy the Sample Integrated Environment Guide.



Note

- By executing the above, "IJServer name: SampleGuideServer" (the default for deployment of the sample environment) and "IJServer name: SampleServer" (the first time default for deployment of the sample application) are created, the following applications are deployed automatically, and IJServer and the Web browser are started.

Deployed in "IJServer name: SampleGuideServer"

- Sample environment applications

Deployed in "IJServer name: SampleServer"

- HelloServlet sample application
- Meeting Room Reservation System (Framework sample application)

However, the Meeting Room Reservation System is not deployed automatically if Framework is not installed.

The IJServer types that are created are as follows:

- When EJB is installed

Web and EJB applications are operated in the same JavaVM

- When EJB is not installed

Only the Web application is operated

Additionally, the IJServer name for the sample application can be changed when the sample environment is executed (initialized). If the name for the sample application is changed, it becomes the default value for the IJServer name next time. If an IJServer name that

has already been created is specified, it may cause the execution (initialization) of the sample environment to fail because of differences in the IJServer name, IJServer type, and applications that have already been deployed.

- When the sample environment is executed (initialized), if an IJServer name is specified for an IJServer that has already been created using the Interstage Management Console, it may cause the following message to be output in the sample environment window.
 - Could not display the window because of defects in the environment.

If this message is output, check that the following items have been set in [Environment Settings] of the IJServer to which the sample environment has been deployed. If these items have not been set, set them using the Interstage Management Console.

Windows32/64

| Item name | Input value |
|-----------|---|
| Path | Interstage installation folder\sample\integrate\bin |

Solaris32/64 Linux32/64

| Item name | Input value |
|--------------|------------------------------|
| Library path | /opt/FJSVisspl/integrate/lib |

- An application that has been deployed automatically cannot be undeployed automatically. Use the Interstage Management Console to undeploy the application.

Additionally, an IJServer that has been created cannot be deleted automatically. Use the Interstage Management Console to delete the IJServer.

- The sample environments mentioned in the above procedure cannot be executed in the following environments.
 - A multiserver environment (Admin Server, Managed Server, reserve server)
 - A system (Interstage) or Web server (Interstage HTTP Server) that has stopped
 - JRE is selected for a custom install
 - A WorkUnit with the default value WorkUnit name ("SampleGuideServer" or "SampleServer"), or a WorkUnit for which the name has been changed exists as mentioned in the above procedure, and only the EJB application is operated for this WorkUnit
 - A WorkUnit called "SampleGuideServer" exists, and the sample environment.

If the following message is output, the sample environment can be set up according to the procedure shown below.

Although the IJServer WorkUnit called "SampleGuideServer" exists, the Sample Integrated Environment Guide is not deployed. For this reason, "SampleGuideServer" is treated as a user resource and processing stops. Check the list of IJServer WorkUnit with Interstage Management Console.

When IJServer WorkUnit SampleGuideServer is not a user resource, retry after deleting SampleGuideServer.

When IJServer WorkUnit SampleGuideServer is a user resource and unable to deploy Sample Integrated Environment Guide, refer to "Appendix A of Installation Guide", create appropriate IJServer WorkUnit from Interstage Management Console and deploy SampleGuideServer.war.

Set up the sample environment according to the procedure shown below.

1. Using the Interstage Management Console, select [System]>[WorkUnit].
2. In the [Create New] page, click details [Show].
3. Click WorkUnit settings [Show], enter the following information and then click [Create].

Simple settings

| Item name | Input value |
|---------------|----------------------|
| WorkUnit name | This can be any name |

WorkUnit settings

Windows32/64

| Item name | Input value |
|-----------|---|
| Path | Interstage installation folder\sample\integrate\bin |

Solaris32/64 Linux32/64

| Item name | Input value |
|--------------|-------------------------------|
| Library path | /opt/FJSSVisspl/integrate/lib |

- Using the Interstage Management Console, select [System]>[WorkUnit]>[WorkUnit name entered above].
- In the [Deploy] window, enter the following information and then click [Deploy].

Deployment settings

| Item name | Input value |
|-----------------|--|
| Deployment file | <p>Windows32/64</p> <p>Interstage installation folder\sample\integrate\SampleGuideServer.war</p> <p>Solaris32/64 Linux32/64</p> <p>/opt/FJSSVisspl/integrate/SampleGuideServer.war</p> |

- After the deployment is complete, click [Start] in the [Operate] window.
- Specify the following URL in the Web browser:

`http://localhost:port number/SampleGuideServer/`

"port number": the port number for the Web server

Create the IJServer WorkUnit for deployment of the sample application according to the Sample Integrated Environment Guide.

- If the Interstage Application Server configuration settings are as shown below, the sample environment start window cannot be displayed.

Using the Interstage Management Console, click [System]. In the [Update System Settings] window, click [Detailed Settings] > [Servlet Service Settings]. [Run Web server and WorkUnit on the same machine?] is not selected.

If [Run Web server and WorkUnit on the same machine?] is not selected, select it.

The settings sometimes revert to those shown above when a multiserver environment is changed to a standalone environment. If [Run Web server and WorkUnit on the same machine?] is not selected, select it.

- If the Web server port number is changed after the installation, the sample environment initial window cannot be displayed. In this case, specify the following in the Web browser:

`http://localhost:port number/SampleGuideServer/`

"port number": the port number for the Web server

If the installation is executed by overwriting the previous version, and the Web server port number is a number other than "80", the procedure shown above is the same.

- If the Web server settings are changed to use SSL after the installation, the sample environment initial window cannot be displayed. In this case, specify the following in the Web browser:

```
https://localhost:port number/SampleGuideServer/
```

"port number": the port number for the Web server

D.1.4 Sample Applications that are Offered

The following sample applications are offered in the sample environment.

- Servlet/JSP
 - HelloServlet application
 - Applications that use Session management
 - Applications that use Listener
 - Applications that use Filter
 - Applications that use the Tag library
 - Applications that use JDBC
- EJB
 - Application that uses the Session Bean
 - Applications that use the BMP that conforms to EJB 2.0 specifications
 - Applications that use the CMP (CMP 1.1) that conforms to EJB 2.0 specifications
 - Application that uses the Message-driven Bean that conforms to EJB 2.0 specifications
 - Applications that use the CMP (CMP 2.0) that conforms to EJB 2.0 specifications
- JMS
 - Application that uses the JMS Publish/Subscribe model
 - Application that uses the JMS Point-To-Point model
 - Application that uses the JMS message selector function
 - Application that uses the JMS queue browse function
- JavaMail
 - Application that does sends and receives that uses JavaMail
- Business model
 - Expense management system
This sample can be executed in 32-bit products only.
- Framework
 - Meetingroom booking system
 - Office management system

For details about the method to import the above sample applications to the sample environment, refer to the appropriate application guide for the sample environment.

Depending on the installation status of Interstage Application Server (the components selected for custom install), it might not be possible to use an application.

Additionally, for JMS to be used in the following applications an environment in which JMS can be used is required. For details about the environment setup method, refer to "[D.1.2 Advance Preparation](#)".

- Application that uses the Message-driven Bean that conforms to EJB 2.0 specifications
- Application that uses the JMS Publish/Subscribe model
- Application that uses the JMS Point-To-Point model
- Application that uses the JMS message selector function
- Application that uses the JMS queue browse function
- Expense management system

 Note

.....
For details about trouble that might occur in the sample environment, refer to the sample environment Help.
.....