HITACHI
Inspire the Next

uCosminexus Application Server

Compatibility Guide

3021-3-J12-10(E)

Notices

■ Relevant program products

See the *Release Notes*.

■ Export restrictions

If you export this product, please check all restrictions (for example, Japan's Foreign Exchange and Foreign Trade Law, and USA export control laws and regulations), and carry out all required procedures.

If you require more information or clarification, please contact your Hitachi sales representative.

■ Trademarks

■ Issued

Aug. 2022: 3021-3-J12-10(E)

■ Copyright

Preface

For details on the prerequisites before reading this manual, see the *Release Notes*.

■ Non-supported functionality

Some functionality described in this manual is not supported. Non-supported functionality includes:

- Audit log functionality
- Compatibility functionality
- Cosminexus Component Transaction Monitor
- Cosminexus Reliable Messaging
- Cosminexus TPBroker and VisiBroker
- Cosminexus Web Service - Security
- Cosminexus XML Security - Core functionality
- JP1 linkage functionality
- Management Server management portal
- Remote installation functionality for the UNIX edition
- SOAP applications complying with specifications other than JAX-WS 2.1
- uCosminexus OpenTP1 linkage functionality
- Virtualized system functionality
- XML Processor high-speed parse support functionality

■ Non-supported compatibility functionality

"Compatibility functionality" in the above list refers to the following functionality:

- Basic mode
- Check of JSP source compliance (cjjsp2java) with JSP1.1 and JSP1.2 specifications
- Database connection using Cosminexus DABroker Library
- EJB client application log subdirectory exclusive mode
- J2EE application test functionality
- Memory session failover functionality
- Servlet engine mode
- Simple Web server functionality
- Switching multiple existing execution environments
- Using EJB 2.1 and Servlet 2.4 annotation

Contents

1 Application Server Functionality

This chapter describes the classifications and purpose of the Application Server functionality, and the manuals corresponding to the functionality. This chapter also describes the functionality that changed in this version.

1.1 Classifications of functionality

The *Application Server* product is used to build an environment for executing applications mainly on a J2EE server compliant with Java EE 7 and for developing applications that run in an execution environment. You can use a variety of functionality, such as functionality compliant with Java EE standard specifications and functionality independently extended on Application Server. By selecting and using functionality according to your goals and intended use, you can build and operate highly reliable systems that have excellent processing performance.

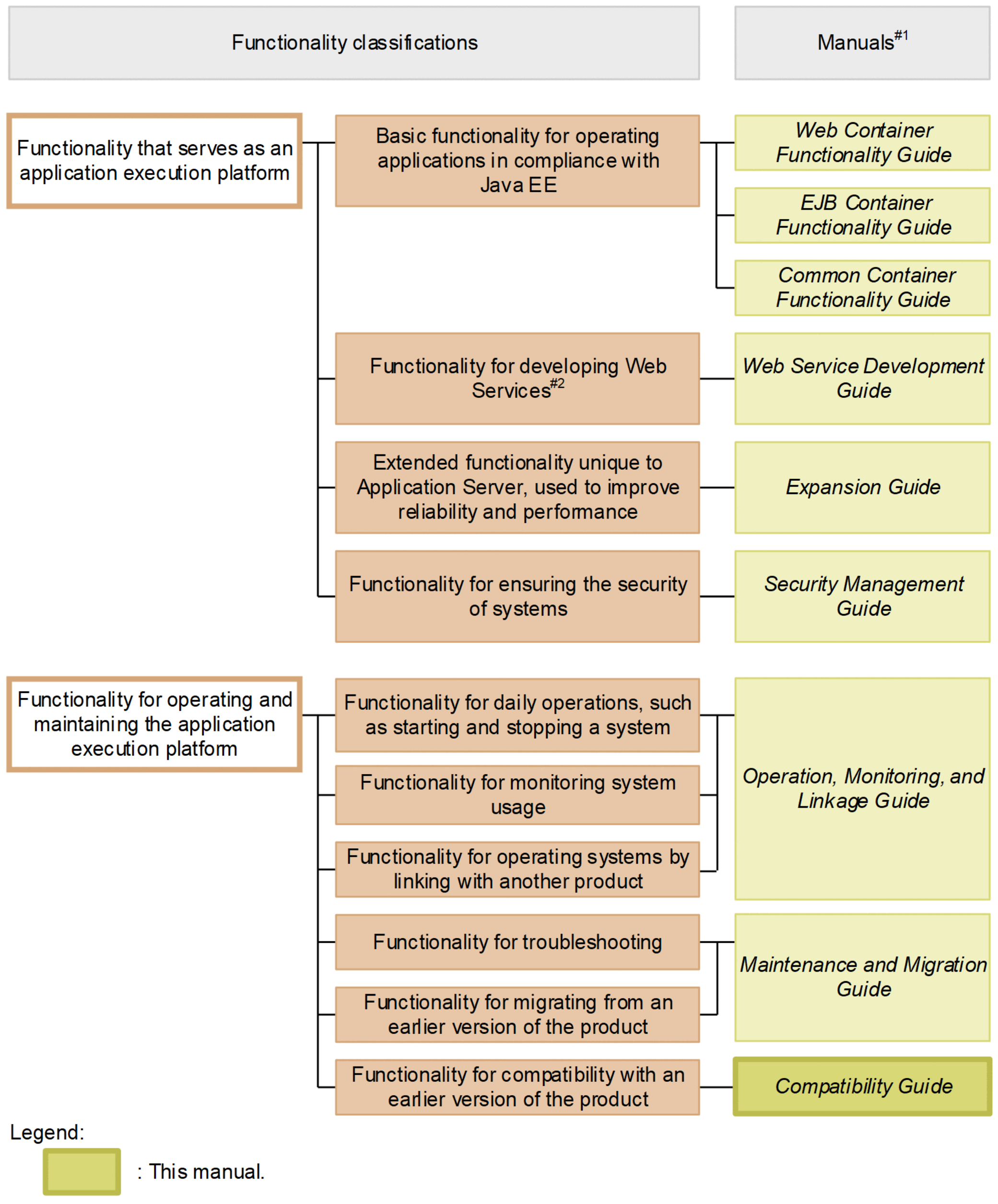
We can classify the Application Server functionality into the following two general categories:

- Functionality that serves as an application execution platform
- Functionality for operating and maintaining the application execution platform

The two categories can be subdivided further according to the positioning and intended use of the functionality. Manuals for the Application Server are provided according to the functionality classifications.

The following figure shows the classifications of Application Server functionality and the corresponding manuals.

Figure 1–1: Classifications of Application Server functionality and corresponding manuals



Functionality classifications
Manuals#1
Functionality that serves as an application execution platform
Basic functionality for operating applications in compliance with Java EE
Web Container Functionality Guide
EJB Container Functionality Guide
Common Container Functionality Guide
Functionality for developing Web Services#2
Web Service Development Guide
Extended functionality unique to Application Server, used to improve reliability and performance
Expansion Guide
Functionality for ensuring the security of systems
Security Management Guide
Functionality for operating and maintaining the application execution platform
Functionality for daily operations, such as starting and stopping a system
Functionality for monitoring system usage
Functionality for operating systems by linking with another product
Operation, Monitoring, and Linkage Guide
Functionality for troubleshooting
Functionality for migrating from an earlier version of the product
Maintenance and Migration Guide
Functionality for compatibility with an earlier version of the product
Compatibility Guide
Legend:
: This manual.

#1

uCosminexus Application Server is omitted from the manual names.

#2

On the Application Server, you can execute SOAP Web Services and RESTful Web Services. Depending on what you want to do, see the following manuals in addition to the *uCosminexus Application Server Web Service Development Guide*.

When developing and executing a SOAP application:

- *uCosminexus Application Server SOAP Application Development Guide*

When ensuring security of SOAP Web Services and SOAP applications:

- *uCosminexus Application Server XML Security - Core User Guide*
- *uCosminexus Application Server Web Service Security Users Guide*

For details about XML processing, see the following:

- *uCosminexus Application Server XML Processor User Guide*

The following subsections describe the functionality classifications and corresponding manuals.

1.1.1 Functionality that serves as an application execution platform

This functionality serves as a platform for executing online businesses and batch businesses implemented as applications. Choose the functionality that you want to use according to the intended use of the system and requirements.

Before building a system or developing an application, you need to determine whether you want to use the functionality that serves as an application execution platform.

The following describes each classification of the functionality that serves as the application execution platform:

(1) Basic functionality for operating applications (basic and development functionality)

This includes the basic functionality for operating applications (J2EE applications). This main functionality is the J2EE server functionality.

Application Server provides Java EE 7-compliant J2EE servers. In addition to providing functionality that complies with the standard specifications, a J2EE server also provides original functionality for Application Server.

The basic and development functionality can be further subdivided into three categories according to the status of J2EE applications that use the functionality. Manuals for the Application Server functionality are also separated according to this classification.

The following is an overview of each type:

- **Functionality for executing web applications (web containers)**
  This category includes the web container functionality that serves as the execution platform for web applications, and the functionality achieved by linking with a web container and web server.
- **Functionality for executing Enterprise Bean (EJB containers)**
  This category includes the EJB container functionality that serves as the execution platform for Enterprise Bean. This category also includes the EJB client functionality for invoking Enterprise Bean.
- **Functionality used in both web applications and Enterprise Beans (Common container functionality)**
  This category includes functionality that can be used for both web applications operating in web containers and Enterprise Bean operating in an EJB container.

(2) Functionality for developing Web Services

This includes the functionality that serves as the execution and development environment of Web Services.

Application Server provides the following engines:

- The JAX-WS engine for binding SOAP messages according to the JAX-WS specifications
- The JAX-RS engine for binding RESTful HTTP messages according to the JAX-RS specifications

(3) Extended functionality unique to Application Server, used to improve reliability and performance (extended functionality)

This includes extended functionality unique to Application Server. This also includes the functionality implemented by using processes other than J2EE server processes, such as processes for a batch server, CTM, and a database.

On Application Server, a variety of functionality has been extended to improve the reliability of the system and to achieve stable operation. Functionality has also been extended to operate applications other than J2EE applications (batch applications) in a Java environment.

(4) Functionality for ensuring the security of systems (security management functionality)

This functionality is used to ensure the security of Application Server-based systems. The functionality includes functionality such as authentication functionality for preventing unauthorized access and encryption functionality for preventing information leakage in communication channels.

1.1.2 Functionality for operating and maintaining the application execution platform

This functionality is used to effectively operate and maintain the application execution platform. After system operation starts, use this functionality as necessary. However, depending on the functionality, you might need to specify settings and implement applications in advance.

Based on the functionality classifications, this section describes the functionality used to operate and maintain the application execution platform.

(1) Functionality for daily operations, such as starting and stopping a system (operation functionality)

This includes the functionality used in daily operations, such as starting and stopping systems or applications, and replacing applications.

(2) Functionality for monitoring system usage (watch functionality)

This includes the functionality used for monitoring system usage and resource depletion. This classification also includes the functionality that outputs information used for auditing, such as the system operation history.

(3) Functionality for operating systems by linking with another product (linkage functionality)

This includes the functionality that is implemented by linking with another product, such as a JP1 product or cluster software.

(4) Functionality for troubleshooting (maintenance functionality)

This includes the functionality for troubleshooting. This classification also includes the functionality for displaying information to be referenced during troubleshooting.

(5) Functionality for migrating from an earlier version of the product (migration functionality)

This includes the functionality for migrating from an earlier version of Application Server to a later version.

(6) Functionality for compatibility with an earlier version of the product (compatibility functionality)

This includes the functionality for compatibility with an earlier version of Application Server. For compatible functionality, we recommend that you migrate the product to the corresponding recommended functionality.

1.1.3 Correspondence between functionality and manuals

Manuals for Application Server functionality are separated according to the functionality classifications.

The following table shows the functionality classifications and corresponding manuals.

Table 1–1: Functionality classifications and the corresponding functionality guides

Category	Functionality	Manuals[#1]
Basic and development functionality	Web container	*Web Container Functionality Guide*
	Using JSF and JSTL	
	Using JAX-RS 2.0	
	WebSocket	
	NIO HTTP server	
	Servlet and JSP implementation	
	EJB container	*EJB Container Functionality Guide*
	EJB client	
	Precautions when implementing Enterprise Bean	
	Naming management	*Common Container Functionality Guide*
	Resource connections and transaction management	
	Invoking Application Server from OpenTP1 (TP1 inbound integrated function)	
	Using JPA 2.1	
	CJMS provider	
	Using JavaMail	
	Using CDI on Application Server	
	Using Bean Validation on Application Server	
	Java Batch	
	JSON-P	
	Concurrency Utilities	
	Application property management	

Category	Functionality	Manuals#1
	Using annotations	
	Formatting and deploying J2EE applications	
	Container extension library	
Extended functionality	Executing applications using the batch server	*Expansion Guide*
	Scheduling and load balancing of requests by using CTM	
	Scheduling batch applications	
	Inheriting the session information between the J2EE servers (Session failover functionality)	
	Database session failover functionality	
	Controlling Full GC using the Explicit Memory Management functionality	
	Output of the application user log	
	CORBA/OTM gateway functionality	
Security management functionality	Authentication using the integrated user management	*Security Management Guide*
	Authentication using application settings	
	Using TLSv1.2 for SSL/TLS communication	
	Controlling by using the management functionality of the load balancers that uses API-based direct connections	
Operation functionality	Starting and stopping the system	*Operation, Monitoring, and Linkage Guide*
	Operation of J2EE applications	
Watch functionality	Monitoring operation information (Statistics collection functionality)	
	Monitoring resource depletion	
	Audit log output functionality	
	Database audit trail linkage functionality	
	Output of operation information using management commands	
	Automatic execution of processing based on management event notifications and management actions	
	Collecting operation statistics of CTM	
	Output of the console log	
Linkage functionality	Operating a system linked with JP1	
	Centralized system monitoring (Linking with JP1/IM)	
	Automatic operation of systems by using jobs (Linking with JP1/AJS)	
	Linking with cluster software	
	Node switching system for models managed for each host (Linking with cluster software)	
	1-to-1 node switching system (Linking with cluster software)	
	Mutual node switching system (Linking with cluster software)	
	N-to-1 recovery system (Linking with cluster software)	

Category	Functionality	Manuals#1
Maintenance functionality	Troubleshooting-related functionality	*Maintenance and Migration Guide*
	Analyzing performance using the performance analysis trace	
	JavaVM functionality of products (hereafter, might be abbreviated as *JavaVM*)	
Migration functionality	Migrating from an earlier version of Application Server	
	Migrating to recommended functionality	
Compatibility functionality	Functionality for compatibility with basic and development functionality	*Compatibility Guide*#2
	Functionality for compatibility with extended functionality	

#1

uCosminexus Application Server is omitted from the manual names.

#2

This manual.

1.2 Explanations of the functionality in this manual

This section describes the meaning of the categories used when explaining functionality in this manual, and describes an example of a table used to describe the categories.

1.2.1 Meaning of explanation categories

The explanations of functionality written in this manual are subdivided into the following five categories. Select and read the explanations according to your goals in referencing the manual.

- Explanation
  This describes the functionality. This describes the purpose, features, and mechanism of the functionality. Read this if you want an overview of the functionality.
- Implementation
  This describes the methods, such as the coding method and the DD writing method. Read this if you develop applications.
- Settings
  This describes how to specify property settings for building systems. Read this if you build a system.
- Operations
  This describes the operation method. This describes the operating procedures and the execution examples of commands to be used. Read this if you operate the system.
- Precautions
  This describes the general precautions when using the functionality. Make sure you read the precautions.

1.2.2 Examples of tables indicating explanation categories

A table indicates the categories into which functionality explanations are subdivided. The table title is *Organization of this chapter* or *Organization of this section*.

The following is an example of a table indicating the categories used to explain some functionality.

Example of a table indicating the categories used to explain some functionality

Table X-1 Organization of this chapter (XX functionality)

Category	Title	Reference
Explanation	What is the *XX* functionality?	*X.1*
Implementation	Implementation of applications	*X.2*
	Definitions in DD and `cosminexus.xml`#	*X.3*
Settings	Settings in the execution environment	*X.4*
Operation	Operations using the *XX* functionality	*X.5*
Precaution	Precautions when using the *XX* functionality	*X.6*

#
For details on `cosminexus.xml`, see *13. Managing Application Attributes* in the *uCosminexus Application Server Common Container Functionality Guide*.

Tip

Property settings for applications that do not contain `cosminexus.xml`

In an application that does not contain `cosminexus.xml`, specify or change properties after importing the properties to the execution environment. You can also change the properties that are already set in the execution environment.

To specify the application settings in the execution environment, use server management commands and property files. For details on application settings for server management commands and property files, see *3.5.2 Procedure for setting the properties of a J2EE application* in the manual *uCosminexus Application Server Application Setup Guide*.

The tags specified in the property files correspond to DD or `cosminexus.xml`. For the correspondence between DD or `cosminexus.xml` and the property file tags, see *2. Cosminexus Application Property File (cosminexus.xml)* in the manual *uCosminexus Application Server Application and Resource Definition Reference Guide*.

Note that the properties specified in each property file can also be specified in the HITACHI Application Integrated Property File.

1.3 Major functionality changes in Application Server 11-00

This section describes the major functionality changes in Application Server 11-00 and the purpose of each change.

This section describes the following content:

- This section gives an overview of the functionality changes and describes the main updates to functionality in Application Server 11-00. For details on the functionality, see the description in the *Reference location* column. The *Reference manual* column and *Reference location* column provide the locations of the main descriptions of the functionality.
- *uCosminexus Application Server* is omitted from the manual names in the *Reference manual* column.

1.3.1 Simplifying implementation and setup

The following table lists the changes to simplify installation and setup.

Table 1–2: Changes made for simplifying implementation and setup

Item	Overview of changes	Reference manual	Reference location
Supporting Windows Server in the development environment	Windows Server OS has been added to the operating systems that support uCosminexus Developer, so that you can build an application development environment in a cloud environment.	--	--

1.3.2 Supporting the standard and existing functionality

The following table describes the changes to support the standard and existing functionality.

Table 1–3: Changes made for supporting the standard and existing functionality

Item	Overview of changes	Reference manual	Referenc e location
Supporting the Servlet 3.0 and 3.1	Servlet 3.0 asynchronous servlets and Servlet 3.1 asynchronous I/O APIs are now supported.	*Web Container Functionality Guide*	*7.1*
Supporting EL 3.0	EL 3.0 is now supported.	*Web Container Functionality Guide*	*2.3.3*
Supporting JSF 2.2	JSF 2.2 is now supported.	*Web Container Functionality Guide*	*Chapter 3*
Supporting JAX-RS 2.0	JAX-RS 2.0 is now supported.	*Web Container Functionality Guide*	*Chapter 4*
Supporting WebSocket 1.0	WebSocket 1.0 is now supported.	*Web Container Functionality Guide*	*Chapter 5*
Adding the NIO HTTP server functionality	Instead of the conventional redirector functionality and in-process HTTP server functionality, the NIO HTTP server functionality was added as an in-process HTTP server that supports non-blocking I/O processing, such as asynchronous servlet and WebSocket.	*Web Container Functionality Guide*	*Chapter 6*

Item	Overview of changes	Reference manual	Referenc e location
Supporting JPA 2.1	JPA 2.1 is now supported, and a JPA provider supporting JPA 2.1 can be used.	*Common Container Functionality Guide*	*Chapter 5*
Supporting CDI 1.2	CDI 1.2 is now supported.	*Common Container Functionality Guide*	*Chapter 8*
Supporting BV 1.1	Bean Validation 1.1 is now supported.	*Common Container Functionality Guide*	*Chapter 9*
Supporting Java Batch 1.0	Batch Applications for the Java Platform (Java Batch) 1.0 is now supported.	*Common Container Functionality Guide*	*Chapter 10*
Supporting JSON-P 1.0	Java API for JSON Processing (JSON-P) 1.0 is now supported.	*Common Container Functionality Guide*	*Chapter 11*
Supporting Concurrency Utilities 1.0	Concurrency Utilities for Java EE 1.0 is now supported.	*Common Container Functionality Guide*	*Chapter 12*
Supporting WebSocket communication	The functionality to relay WebSocket communication from the HTTP Server to J2EE servers was added.	*HTTP Server User Guide*	*4.15*

1.3.3 Maintaining and improving reliability

The following table lists the changes to maintain and improve reliability.

Table 1–4: Changes made for maintaining and improving reliability

Item	Overview of changes	Reference manual	Referenc e location
Changing the encrypted communication module	`mod_ssl` is now used as the encrypted communication module for the HTTP Server.	*HTTP Server User Guide*	*Chapter 5*, *Appendix H*

1.3.4 Other purposes

The following table lists the changes for other purposes.

Table 1–5: Changes made for other purposes

Item	Overview of changes	Reference manual	Referenc e location
Addition of V9 compatibility mode	The V9 compatibility mode was added to maintain compatibility with version 9 for users who migrate from version 9 or an earlier version of J2EE server.	*Maintenance and Migration Guide*	*10.3.3*

2 Overview of V9 Compatibility Mode

This chapter provides an overview on V9 compatibility mode for systems that emphasize compatibility with version 9 or earlier.

2.1 V9 compatibility mode and recommended mode

On Application Server 11-00, functionality for which compatibility with 09-87 and earlier versions cannot be maintained has changed to support the new Java EE 7 specifications, such as Servlet 3.1 and WebSocket 1.0. For this reason, when migrating a system designed for version 09-87 or earlier to an environment that uses version 11-00 or later, some design changes are required.

Therefore, V9 compatibility mode is provided for systems where compatibility with 09-87 and earlier versions is important. The V9 compatibility mode does not use the new functionality of 11-00. On a J2EE server configured in V9 compatibility mode, operation is equivalent to 09-87, and part of the functionality discontinued in 11-00 and later versions can also be used.

Also, the mode that does not use the V9 compatibility mode is called *recommended mode*. In the recommended mode, you can use the new functionality of 11-00, but cannot use the functionality discontinued in 11-00 and later versions.

3 How to Use V9 Compatibility Mode

The following two methods are available for using V9 compatibility mode.

- The method of specifying V9 compatibility mode when creating a new J2EE server

- The method of performing an upgrade installation to migrate an existing J2EE server created in version 09-87 or earlier of Application Server

This chapter describes how to use these methods.

3.1 The method of specifying V9 compatibility mode when creating a new J2EE server

In the following four situations, use the method of specifying V9 compatibility mode when creating a new J2EE server:

- When using the Smart Composer functionality
- When using the management portal
- When using a J2EE server command
- When using the development environment instant setup functionality

3.1.1 When using the Smart Composer functionality

When using the Smart Composer functionality to create a new system that includes a logical J2EE server in V9 compatibility mode, specify `V9` as the value of the `manager.j2ee.compat` parameter, which sets compatibility mode for the J2EE server. This parameter is in the parameters specified for the logical J2EE server in the Easy Setup definition file.

The following example shows items specified in the Easy Setup definition file.

(**Example of items specified in the Easy Setup definition file**)

```
<?xml version="1.0" encoding="UTF-8"?>
<model-definition
xmlns="http://www.cosminexus.com/mngsvr/schema/ModelDefinition-2.5">
  <web-system>
    <name>MyCompatWebSystem</name>
    <tier>
         ...
      <configuration>
        <logical-server-type>j2ee-server</logical-server-type>
        <param>
          <param-name>manager.j2ee.compat</param-name>
          <param-value>V9</param-value>
        </param>
          ...
```

For details, see *4.11.1 Parameters used for setting up the compatibility mode of the J2EE server* in the manual *uCosminexus Application Server Definition Reference Guide*.

3.1.2 When using the management portal (INTENTIONALLY DELETED)

(INTENTIONALLY DELETED)

3.1.3 When using a J2EE server command

When creating a new J2EE server in V9 compatibility mode by using the `cjsetup` command for setting up a J2EE server, specify `-compat V9` for the command line argument of the `cjsetup` command.

The following shows an example of executing the `cjsetup` command.

(Example of executing the `cjsetup` command)

```
cjsetup MyCompatJ2EEServer -compat V9
```

For details, see *cjsetup (set up or unsetup J2EE server)* in *2.2 Commands used for operating J2EE servers* in the manual *uCosminexus Application Server Command Reference Guide*.

3.1.4 When using the development environment instant setup functionality (INTENTIONALLY DELETED)

(INTENTIONALLY DELETED)

3.2 The method of performing an update installation to migrate an existing J2EE server

If you perform an update installation to migrate a J2EE server set up in version 09-87 or earlier to an environment that uses version 11-00 or later, the already set up existing J2EE server automatically uses V9 compatibility mode.

For details on performing an upgrade installation to migrate from an earlier version, see *Chapter 10* in the manual *uCosminexus Application Server Maintenance and Migration Guide*.

3.3 Checking the J2EE server compatibility mode

You can check in which mode the already set up J2EE server is operating by checking the KDJE60001-I message that is output to the message log when the J2EE server starts. The KDJE60001-I messages shown in recommended mode and V9 compatibility mode are as follows:

In recommended mode:

```
KDJE60001-I
The J2EE server will start as 'Java EE Container Mode'. (container versio
n = V11)
```

In V9 compatibility mode:

```
KDJE60001-I
The J2EE server will start as 'Java EE Container Mode'. (container versio
n = V9)
```

3.4 Precautions on use

- For a J2EE server that is already set, you cannot switch it from recommended mode to V9 compatible mode, or switch from V9 compatible mode to recommended mode. Execute the unsetup operation for the J2EE server, and then set up again.
- To delete only the logical J2EE server from a system built by using the Smart Composer functionality and management portal, and then create a logical J2EE server with the same real server name but in a different mode, use the `cjsetup` command to execute the unsetup operation for the J2EE server before creating the logical J2EE server.

  For details on the `cjsetup` command, see *cjsetup (set up or unsetup J2EE server)* in *2.2 Commands used for operating J2EE servers* in the manual *uCosminexus Application Server Command Reference Guide*.

4 Functionality of V9 Compatibility Mode and Recommended Mode

This chapter describes the functionality of V9 compatibility mode and recommended mode.

4.1 Functional differences between V9 compatibility mode and recommended mode

The following shows compatible functionality that is available only in V9 compatibility mode and new functionality that is available only in recommended mode on version 11-00 and later of Application Server. Functionality not described in this section is usable in both V9 compatible mode and recommended mode.

Table 4–1: Availability of each functionality by mode

Functionality	Recommended mode	V9 compatibility mode	Location of description
NIO HTTP server functionality	Y	N	*uCosminexus Application Server Web Container Functionality Guide*
Web server integration functionality (Redirector functionality)	N	Y	*5. Web Server Integration* in this manual
In-process HTTP server functionality	N	Y	*6. In-Process HTTP Server* in this manual
Servlet 3.0 asynchronous servlet functionality	Y	N	*uCosminexus Application Server Web Container Functionality Guide*
Servlet 3.1	Y	N	
WebSocket 1.0	Y	N	*uCosminexus Application Server Web Container Functionality Guide*
Java Batch 1.0	Y	N	*uCosminexus Application Server Common Container Functionality Guide*
Concurrency Utilities for Java EE 1.0	Y	N	*uCosminexus Application Server Common Container Functionality Guide*
JSON-P 1.0	Y	N	*uCosminexus Application Server Common Container Functionality Guide*
CDI 2.1	Y	N	*uCosminexus Application Server Common Container Functionality Guide*
Bean Validation 1.1	Y	N	*uCosminexus Application Server Common Container Functionality Guide*
JSF 2.2	Y	N	*uCosminexus Application Server Web Container Functionality Guide*
JAX-RS 2.0	Y	N	*uCosminexus Application Server Web Container Functionality Guide*
Cosminexus JAX-RS engine (JAX-RS 1.1)	N	Y	*7. Cosminexus JAX-RS Engine (JAX-RS 1.1)* in this manual
JPA 2.1 support provider	Y	N	*uCosminexus Application Server Common Container Functionality Guide*
JPA 1.0 support provider (CJPA provider functionality)	N	Y	*9. Cosminexus JPA Provider* in this manual
JPA container functionality	N	Y	*8. How to Use JPA with Application Server* in this manual
HTTP response compression functionality	N	Y	*10.2 HTTP response compression functionality* in this manual

Legend:

Y: Available

N: Not available

5 Web Server Integration

This chapter describes the functionality related to Web server integration.

5.1 Organization of this chapter

Application Server provides the redirector functionality for Web server integration. A *redirector* refers to a library provided in the Web container. By registering the redirector in the Web server, you will be able to process a specific requests, from the HTTP requests for the Web server, in the specified Web container and distribute and process the requests in multiple Web containers.

The following table lists the functionality and the corresponding reference sections for each functionality related to the Web server integration:

Table 5–1: Functionality and the corresponding reference sections of each functionality related to Web server integration

Functionality	Reference
Distributing requests with the Web server (Redirector)	*5.2*
Distributing requests by URL pattern	*5.3*
Distributing requests by the round-robin format	*5.4*
Distributing requests by the POST data size	*5.5*
Communication timeout (Web server integration)	*5.6*
Specifying the IP address (Web server integration)	*5.7*
Error page customization (Web server integration)#	*5.8*
Viewing the top page by specifying the domain name	*5.9*
Notification of Gateway Information to a Web Container	*5.10*
Controlling the number of concurrently executed threads for each Web container	*5.11*
Objects for communication with redirector	*5.12*
Explicit heap tuning	*5.13*

#

Functionality available only when you use Cosminexus HTTP Server as the Web server. You cannot use this functionality in the case of using Microsoft IIS.

Furthermore, in the case of Web server integration, you can also use the SSL-based authentication and the data encryption functionality of Cosminexus HTTP Server or the SSL-based authentication and the data encryption functionality of Microsoft IIS.

For SSL-based authentication and data encryption in Cosminexus HTTP Server, see *7.2.2 SSL setup with Cosminexus HTTP Server* in the manual *uCosminexus Application Server Security Management Guide*. For details on how to set SSL for Microsoft IIS, see the help of SSL for Microsoft IIS. You can use this functionality only in the Web redirector environment.

Environment settings required for Web server integration

The following environment settings are required for the Web server integration:

- **When Smart Composer is used**

  See *Appendix D Precautions related to Cosminexus HTTP Server Settings* and set up the environment of Cosminexus HTTP Server.

- **When Smart Composer is not used**

  Set up the environment of one of the following Web servers:

- Cosminexus HTTP Server (*Appendix D*)
- Microsoft IIS (*Appendix E*)

5.2 Distributing requests with the Web server (Redirector)

This section explains the distribution of requests with the redirector.

You can use this functionality only when you use the Web server integration functionality. To distribute requests, you must define distribution for the host on which the Web server or the redirector is running.

The following table describes the organization of this section:

Table 5–2: Organization of this section (Distributing requests with the Web server (redirector))

Category	Title	Reference
Description	Mechanism of request distribution with the redirector	*5.2.1*
	User-defined file for setting the request distribution method (When the Smart Composer functionality is used)	*5.2.2*
	User-defined file for setting the request distribution method (When the Smart Composer functionality is not used)	*5.2.3*
Notes	Points to be considered during Web server integration	*5.2.4*

Note:

There is no specific description of *Implementation*, *Settings*, and *Operations* for this functionality.

Note that the required definitions differ according to the type of the Web server used. Also, if the type of the Web server used is Cosminexus HTTP Server, the definitions also differ according to the type of the functionality used for the system building. The following table lists the types of Web servers and the definitions to be used:

Table 5–3: Web server types and the definitions to be used

Type of the Web server	Type of system building function	Definitions
Cosminexus HTTP Server	Smart Composer functionality	• Definition of Easy Setup definition file • Definition of `workers.properties` • Definition of `mod_jk.conf`
	Other than Smart Composer functionality	• Definition of `workers.properties` • Definition of `mod_jk.conf`
Microsoft IIS	--	• Definition of `workers.properties` • Definition of `uriworkermap.properties` • Definition of `isapi_redirect.conf`

Legend:

--: Not applicable

For details on distributing requests by redirection when using the in-process HTTP server, see *6.7 Request distribution with the redirector*.

5.2.1 Mechanism of request distribution with the Redirector

If you use the redirector, from among the HTTP requests sent to the Web server, specific requests can be processed in the specified Web container, and requests can be processed by distributing to multiple Web containers.

In the case of distributing requests with the redirector, use the Web container execution process, called the *worker process*# that runs behind the Web server. A worker process is used to process requests including servlets and JSPs, via the redirector. Data exchange between the Web server and a worker process is based on TCP/IP and is executed through a specific port number set by the user. To specify the redirector settings, use the setup unit that abstracts the Web container called *worker*. The worker includes a worker indicating a stand-alone J2EE server and a worker indicating a J2EE server in a cluster configuration. The worker that forwards requests to the J2EE server is called a *forwarding worker*. A forwarding worker is the `ajp13` type worker.

#

A worker process actually acts as a J2EE server.

(1) Patterns to transfer the requests

The patterns to transfer requests from the redirector to the worker process are as follows:

- Transfer from one Web server to one worker process
- Transfer from one Web server to multiple worker processes

Note that the mechanism of request distribution is not affected even if the Web server and the worker processes are present on the same machine or on different machines.

The following figures show the patterns to transfer requests from the redirector to the worker process:

Figure 5–1: Transfer from one Web server to one worker process

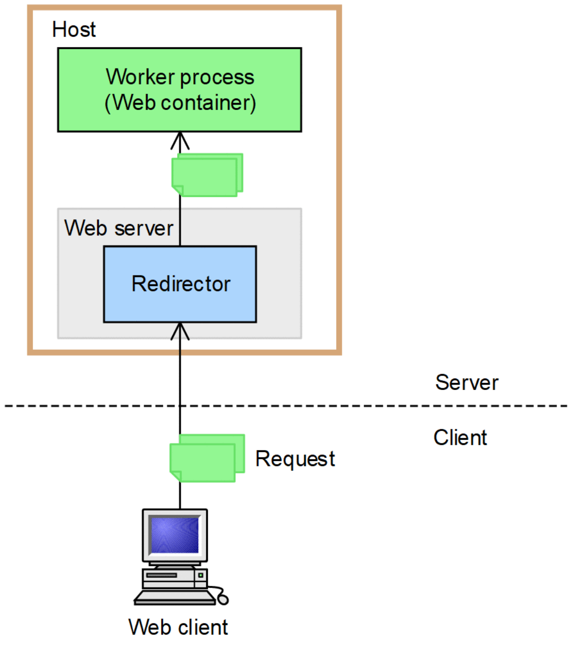
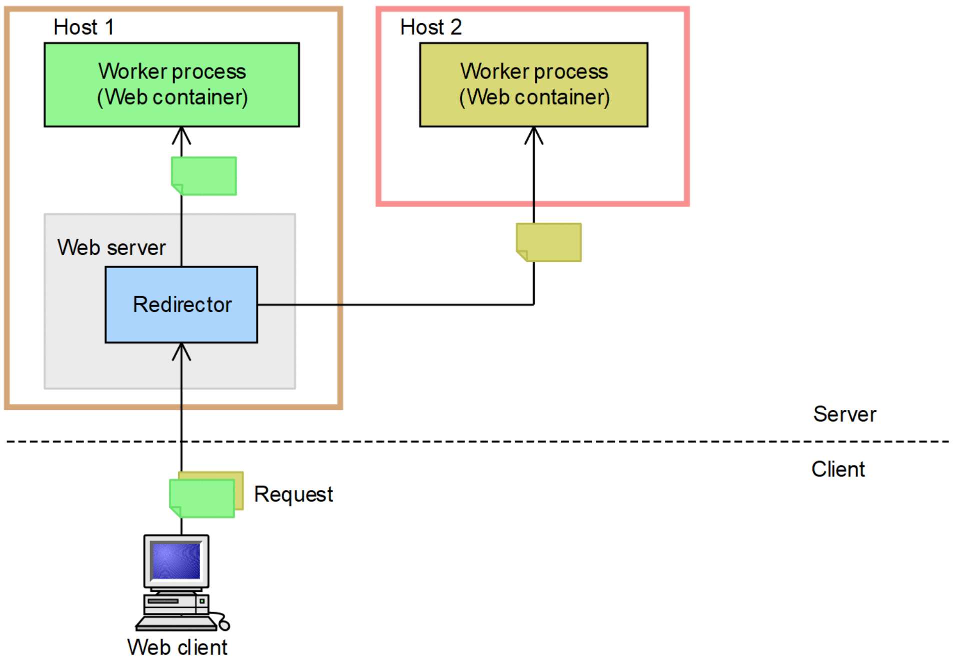


Figure 5–2: Transfer from one Web server to multiple worker processes



To distribute the requests to multiple Web containers, define the worker processes of multiple Web containers as the distribution destinations, in the redirector registered in the Web server.

(2) Request distribution method

The methods to distribute requests with the redirector include:

- **Distributing by URL pattern**
  Use this method when you want to execute a specific processing in a single Web container, and when you want to distribute a process to multiple Web containers.
  You can use this method when there is one worker process, as well as when there are multiple worker processes.
- **Distributing in round-robin format with a load balancer**
  Use this method when you want to distribute a process to multiple Web containers.
- **Distributing by the POST data size**
  Use this method when you want to distribute a process to multiple Web containers. You can specify this distribution method only when the Web server used is Cosminexus HTTP Server.
  Note that you cannot use this distribution method when the following functionality are used:
  - Session failover functionality
  - Distributing requests by the round robin format

To create a worker process, define the following attributes in a file (`workers.properties`) called the *worker definition file*:

- Worker name
- Worker type
- Host name or IP address of the Web server on which the worker is running
- Port number received by the worker

The following workers are already defined in a standard `workers.properties` file. When the Web server and the Web container are operated on the same host, you need not change these parameters.

Worker attributes	Parameter
Worker name	`worker1`
Worker type	`ajp13`
Host name	`Localhost`
Port number	`8007`

For details on how to define a worker process, see *14.2.4 workers.properties (Worker definition file)*.

5.2.2 User-defined file for setting the request distribution method (When the Smart Composer functionality is used)

To distribute requests, edit the following user-defined files in a text editor and specify the worker, mapping between the URL pattern and worker, and the redirector operations.

- **Easy Setup definition file**
  Specify the worker definition, the worker-wise parameters, and the mapping between the URL pattern and workers. Use this file to set up request distribution by URL pattern. For request sorting by using the round-robin format and request sorting by using the POST data size, the settings are output to the free-tier physical tier.
- **workers.properties (Worker definition file)**
  Specify the worker definition and the worker-wise parameters. Use this file to set up request distribution by the round robin format and the request distribution by the POST data size.
- **mod_jk.conf (Redirector action definition file)**
  Specify the mapping between the URL pattern and workers and specify the redirector operations, such as which URL patterns will be forwarded to the Web container with the requests sent to Cosminexus HTTP Server. Use this file to set up request distribution by the round robin format and the request distribution by the POST data size.

For details on the Easy Setup definition file, see *4.3 Easy Setup definition file* in the manual *uCosminexus Application Server Definition Reference Guide*. For details on the Worker definition file, see *14.2.4 workers.properties (Worker definition file)*. For details on the Redirector Operation definition file, see *14.2.1 isapi_redirect.conf (Redirector action definition file for Microsoft IIS)*.

5.2.3 User-defined file for setting the request distribution method (When the Smart Composer functionality is not used)

To distribute requests, edit the following user-defined files in a text editor and specify workers, mapping between the URL pattern and workers, and the redirector operations.

The files to be setup depend on the used Web server. The common user-defined files and the user-defined files for each Web server are separately described here. For details on the user-defined file `httpsd.conf`, see the manual *uCosminexus Application Server HTTP Server User Guide*. For details on other user definition files, see *Part 3 Reference (V9 Compatibility Mode)*.

(1) Common user-defined files

The common user-defined files for Cosminexus HTTP Server and Microsoft IIS are as follows:

- **workers.properties (worker definition file)**
  Specify the worker definitions and the parameters for each worker.
  The storage location of this file is as follows:
  - In Windows
    Cosminexus-installation-directory`\CC\web\redirector\workers.properties`
  - In UNIX
    `/opt/Cosminexus/CC/web/redirector/workers.properties`
- **usrconf.properties (user property file)**
  Set the communication timeout when the Web container receives a request from the redirector.
  The storage location of this file is as follows:
  - In Windows
    Cosminexus-installation-directory`\CC\server\usrconf\ejbs\`*server-name*`\usrconf.properties`
  - In UNIX
    `/opt/Cosminexus/CC/server/usrconf/ejb/server-name/usrconf.properties`

(2) User-defined files for Cosminexus HTTP Server

The user-defined files for Cosminexus HTTP Server are as follows:

- **mod_jk.conf (redirector action definition file)**
  Specify the redirector operations for Cosminexus HTTP Server.
  The storage location of this file is as follows:
  - In Windows
    Cosminexus-installation-directory`\CC\web\redirector\mod_jk.conf`
  - In UNIX
    `/opt/Cosminexus/CC/web/redirector/mod_jk.conf`
- **httpsd.conf (Cosminexus HTTP Server Definition file)**
  Specify the directive (parameter that defines the execution environment of the Web server) for defining the operating environment of Cosminexus HTTP Server.
  The storage location of this file is as follows:
  - In Windows
    Cosminexus-installation-directory`\httpsd\conf\httpsd.conf`
  - In UNIX
    `/opt/hitachi/httpsd/conf/httpsd.conf`

(3) User-defined files for Microsoft IIS

The user-defined files for Microsoft IIS are as follows:

- **uriworkermap.properties (mapping definition file)**
  Specifies the mapping between the URL pattern and worker in Microsoft IIS.

The storage location of this file is as follows:
Cosminexus-installation-directory`\CC\web\redirector\uriworkermap.properties`

- **isapi_redirect.conf (redirector action definition file)**
  Specify the redirector operations for Microsoft IIS.
  The storage location of this file is as follows:
  Cosminexus-installation-directory`\CC\web\redirector\isapi_redirect.conf`

(4) Notes

The followings are the notes related to the user-defined files:

- During the overwrite installation, the user-defined file is not overwritten.
- For the processing of the worker definition file and the Web server definition file when performing the upgrade installation, see *10. Migrating from Application Server of Earlier Versions (In the J2EE Server Mode)* in the *uCosminexus Application Server Maintenance and Migration Guide*.
- The maximum number of characters in one line of the redirector definition file is 1,023. Specify settings within this character count.
- In the following user-defined files, if several parameters are specified with the same name, the value of the parameter specified first is used for operations:
  - `isapi_redirect.conf`
  - `workers.properties`
  - `uriworkermap.properties`

5.2.4 Points to be considered during Web server integration

This section explains the points to be considered when integrating with the Web server.

(1) Upper limit value of the request headers and response headers that can be sent and received by the Web container

When integrating with the Web server, an upper limit is set on the size of the request headers and the response headers that can be sent and received by the Web container. The respective upper-limit value is 16 KB. Note that you cannot send and receive the headers exceeding 16 KB.

(2) Points to be considered when using Cosminexus HTTP Server

You cannot use the virtual host functionality of the HTTP Server when a Web server is integrated.

To perform different redirections for each virtual host, use the management portal or Smart Composer to build multiple HTTP Servers on the same host, and then set a redirector for each HTTP Server.

(3) Points to be considered when using the Microsoft IIS

The points to be considered when using the Microsoft IIS are explained below:

- When multiple Web sites are built by Microsoft IIS, you cannot integrate simultaneously with these Web sites. When you are building multiple Web sites, set a redirector in each Web site.

- Change the URL information for the requests transferred to the Web container with the redirector for Microsoft IIS. Use the changed request URL in the ISAPI filter.

  Consequently, you cannot acquire the request URL received first by the Microsoft IIS with the ISAPI filter executed after the redirector for Microsoft IIS. If you want to acquire the request URL received by Microsoft IIS with the ISAPI filter, you need to set a higher priority order for the ISAPI filter as compared to the redirector for Microsoft IIS. Note that when you need to change the priority order of the redirector for Microsoft IIS to `Medium` or `Low` in order to adjust the priority order of the ISAPI filter, specify the priority order with the `filter_priority` key of the action definition file of the redirector for Microsoft IIS. For the `filter_priority` key, see *14.2.1 isapi_redirect.conf (Redirector action definition file for Microsoft IIS)*.

- When integrating with Microsoft IIS, even if you specify the following HTTP request headers in the Web client, you cannot receive these request headers in a Web application:

  - `tomcaturl`
  - `tomcatquery`
  - `tomcatworker`
  - `tomcattranslate`

  These HTTP request headers are used in the redirector.

- When integrating with Microsoft IIS, you cannot specify settings for distributing requests by the POST data size.

5.3 Distributing requests by URL pattern

This section explains the distribution of requests by the URL patterns.

You can distribute the requests by the URL patterns included in an HTTP request. Consequently, only a specific processing can be executed in the Web container, and a processing can be distributed to multiple Web containers depending on the contents of the processing.

The following table describes the organization of this section.

Table 5–4: Organization of this section (Distributing requests by the URL pattern)

Category	Title	Reference
Description	Overview of distributing requests by URL pattern	*5.3.1*
	Types of URL patterns and priority of applicable patterns	*5.3.2*
Settings	Execution environment settings (When the Smart Composer functionality is used)	*5.3.3*
	Execution environment settings (When the Smart Composer functionality is not used)	*5.3.4*

Note:

There is no specific description of *Implementation*, *Operations*, and *Notes* for this functionality.

5.3.1 Overview of distributing requests by URL pattern

Define the requests transferred to the Web container according to the mapping between the URL pattern and the worker process. The redirector can switch the Web containers that transfer the requests, depending on the set URL pattern.

For example, you can define "Process the HTTP request containing the URL `/examples/` in a Web container", and "Process the HTTP request containing the URL `/examples1/` in Web container A, and the HTTP request containing the URL `/examples2/` in Web container B".

The following figures show the examples of request distribution with the redirector:

Figure 5–3: Request distribution with the redirector (when distributing and transferring specific requests to the Web container)

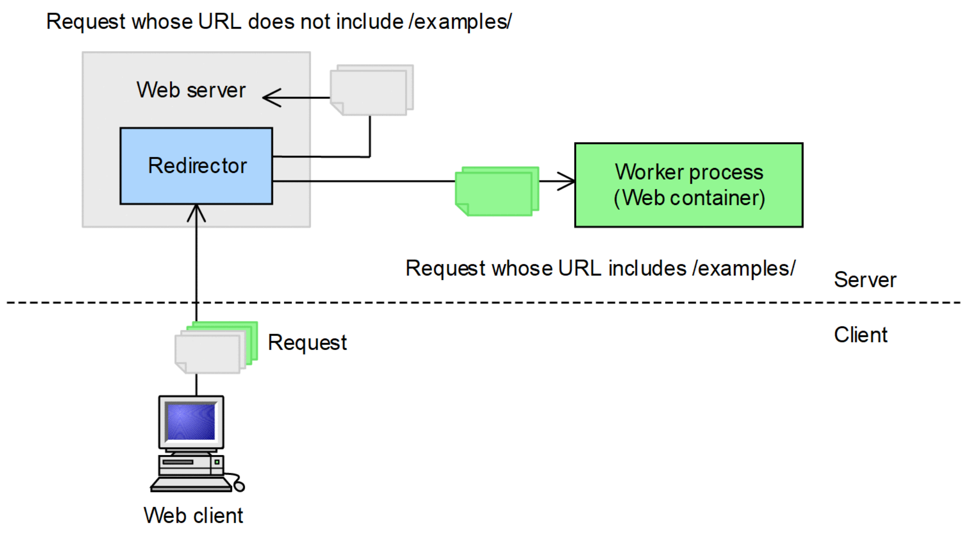
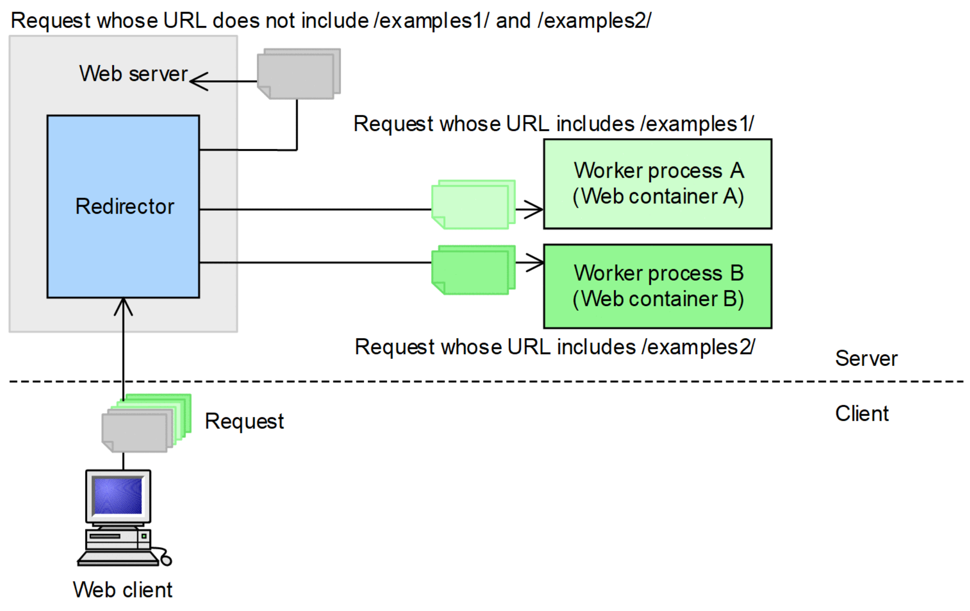

Request whose URL does not include /examples/
Web server
Redirector
Worker process (Web container)
Request whose URL includes /examples/
Server
Client
Request
Web client

Figure 5–4: Request distribution with the redirector (when distributing and transferring requests to multiple Web containers)


Request whose URL does not include /examples1/ and /examples2/
Web server
Redirector
Request whose URL includes /examples1/
Worker process A (Web container A)
Worker process B (Web container B)
Request whose URL includes /examples2/
Server
Client
Request
Web client

5.3.2 Types of URL patterns and priority of applicable patterns

This section describes the types of URL patterns that you can specify for URL mapping of the Redirector and the priority of applicable patterns.

(1) Types of URL patterns

You can specify the following four types of URL patterns for the URL mapping of the redirector:

- **Complete path specification**

  This is a completely matching pattern.

  URL format:

  /path

  When specifying only the route, use "/".

  Characters you can specify in */path*:

  Specify a string having at least one of the following characters:

  Single-byte alphanumeric characters, "/", "*", "-", ".", "_", "~", "!", "$", "&", "'", "(", ")", "+", ",", "=", ":", "@"

  Example:

  When the URL pattern is `/examples/jsp/index.jsp`, and the URL is `/examples/jsp/index.jsp`, it indicates a match.

- **Path specification**

  In this pattern, the paths are matching.

  URLformat:

  */path/**

  When specifying all requests, use `/*`.

Characters you can specify in /*path*:

Specify a string having at least one of the following characters:

Single-byte alphanumeric characters, "/", "*", "-", ".", "_", "~", "!", "$", "&", "'", "(", ")", "+", ",", "=", ":", "@"

Example:

When the URL pattern is `/examples/*`, and the URL is `/examples/jsp/index.jsp`, it indicates a match.

- **Extension specification**

  In this pattern, the extensions are matching. This pattern is applicable to all the hierarchies below the specified path.

  URL format:

  /*path*/*.*extension*

  When specifying all the paths, use /*.*extension*.

  Characters you can specify in *path* and *extension*:

  Specify a string having at least one of the following characters:

  Single-byte alphanumeric characters, "/", "*", "-", ".", "_", "~", "!", "$", "&", "'", "(", ")", "+", ",", "=", ":", "@"

  Example:

  When the URL pattern is `/examples/*.jsp`, and the URL is `/examples/jsp/index.jsp`, it indicates a match.

- **Suffix specification**

  In this pattern, suffixes are matching. This pattern is applicable to all the hierarchies below the specified path.

  URL format:

  /*path*/**suffix*

  When specifying all the paths, use /**suffix*.

  Characters you can specify in *path* and *suffix*:

  Specify a string having at least one of the following characters:

  Single byte alphanumeric characters, "/", "*", "-", ".", "_", "~", "!", "$", "&", "'", "(", ")", "+", ",", "=", ":", "@"

  Example:

  When the URL pattern is `/examples/servlet/*Servlet`, and the URL is `/examples/servlet/HelloServlet`, it indicates a match.

Important note

The following are the notes on specifying a URL pattern:

- A URL pattern must not begin with anything but "/". In Windows, if you specify a character other than "/", the KDJE41012-E message is output and the mapping is ignored. In any other OS, a message is displayed and Cosminexus HTTP Server fails to start.
- A "*", when used as a wildcard, cannot be specified before the /* in a URL pattern. If you specify anything other than "/" just before the first "*" in a URL pattern, the URL pattern is treated as a *Complete path specification* and /* is not treated as a wildcard even if it is a part of the URL.
- Do not describe multiple mappings of the same URL pattern. The behavior in case you specify multiple mappings, is as follows.

The former URL pattern mapping is used in the *Complete path specification* and the *Path specification*. The latter mapping is used in the case of *Extension specification* and *Suffix specification*.

- You must use only valid values in *path*, *extension*, and *suffix*. If you use an invalid character in a URL pattern, some types of characters might not be forwarded to the Web container.
- The string length of *extension* or *suffix* must be at least one character long. If the length is shorter than one character, the *Extension specification* outputs the KDJE41041-W message, and the mapping is ignored. In the *Suffix specification*, the value that you specify is treated as a URL pattern of the *Path specification*.

(2) Priority of applicable patterns

Among the mapping to these four URL patterns, the URL pattern with the highest priority is *Complete path specification*. When the URL does not match with *Complete path specification*, path matching is judged in the following order, and the applicable URL pattern is decided:

1. When the URL does not match with *Complete path specification*

   The longest matching URL pattern from among *Path specification*, *Extension specification*, and *Suffix specification* is applied. Longest match refers to the longest matching URL from the beginning (`"/"`) until the high order path of `"*"`.

   The URL in which the following two mappings are defined is illustrated below as an example:

   Mapping definition:
   ```
   /examples/* worker1
   /examples/jsp/* worker2
   ```

   In this case, when the URL is `/examples/jsp/index.jsp`, the mapping of `worker2` is applied, and when the URL is `/examples/test/index.jsp`, the mapping of `worker1` is applied.

2. In addition to the conditions of 1, when multiple longest matching *Path specification*, *Extension specification*, and *Suffix specification* URL patterns are present

   Extension specification or *Suffix specification* is given priority over *Path specification*.

   The URL in which the following two mappings are defined is illustrated below as an example:

   Mapping definition:
   ```
   /examples/jsp/* worker1
   /examples/jsp/*.jsp worker2
   ```

   In this case, when the URL is `/examples/jsp/index.jsp`, the mapping of `worker2` is applied, and when the URL is `/examples/jsp/test.html`, the mapping of `worker1` is applied.

3. In addition to the conditions of 1 and 2, when multiple longest matching *Extension specification* and *Suffix specification* URL patterns are present

   The URL pattern specified later is given priority.

   The URL in which the following two mappings are defined is illustrated below as an example:

   Mapping definition:
   ```
   /examples/*.jsp worker1
   /examples/*jsp worker2
   ```

   When URL is specified in this order, the mapping of `worker2` is applied when the URL is `/examples/jsp/index.jsp`.

Important note

You must note the following points when judging the priority of applicable patterns:

- If a request URL includes a query (string after a "?" mark in the URL), the query part is not used when comparing with the URL pattern.
  Example:
  If the request URL is `/examples/jsp/index.jsp?query=foo`, the URL used for comparison is `/examples/jsp/index.jsp`.
- If a request URL includes a parameter (string from a semicolon (`;`)), the parameter part is not used when comparing with the URL pattern.
  Example:
  If the request URL is `/examples/jsp/index.jsp;jsessionid=0000`, the URL used for comparison is `/examples/jsp/index.jsp`.
- A request URL path is first normalized and then the URL is compared with the URL pattern to judge whether both the URLs match.
  Example:
  If a request URL is `/examples/../examples/./jsp//index.jsp`, the URL used for comparison is `/examples/jsp/index.jsp`.
- A URL pattern is never normalized. Therefore, a URL pattern that includes `./` or `../` does not match with the request URL.
- In Windows, an extension of a URL pattern specified in the *Extension specification* is not case sensitive.

5.3.3 Execution environment settings (When the Smart Composer functionality is used)

Distributing requests by the URL patterns included in the HTTP requests enables you to execute only the specific processes in the Web container and to distribute requests to multiple Web containers according to the processing contents. Note that, when you use 'Distributing requests by URL pattern' method, as a principle, the requests are distributed in the Web application. Define the URL pattern as an operation of the redirector.

(1) Setup procedure

To set the distribution of requests by the URL pattern:

1. Define the workers and mapping between the URL pattern and workers in the Easy Setup definition file.
   Specify the list of worker names, worker types (set `ajp13`), port number, and host name.
   If mapping is already defined, delete or replace the definition.

2. Set up the Web server environment and restart the Web server.
   For details on the Web server settings, see *Appendix D Precautions related to Cosminexus HTTP Server Settings*.

(2) Settings in the Easy Setup definition file

Specify the definition for distributing requests by the URL pattern in the `<configuration>` tag of the logical Web server (web-server) in the Easy Setup definition file.

The following table lists the definitions in the Easy Setup definition file for distributing requests by the URL pattern:

Table 5–5: Definitions in the Easy Setup definition file for distributing requests by the URL pattern

Category	Parameter	Description
Worker definitions	`worker.list`	Specifies a list of one or multiple worker names.
	`worker.`*worker-name*`.host`	Specifies the worker host name or IP address.
	`worker.`*worker-name*`.port`	Specifies the worker port number.
	`worker.`*worker-name*`.type`	Specifies the worker type (`ajp13`, `lb`, or `post_size_lb`).
	`worker.`*worker-name*`.cachesize`	Specifies the number of worker connections that are reused in the redirector. This parameter can only be specified in Windows.
	`worker.`*worker-name*`.receive_timeout`	Specifies the communication timeout value.
	`worker.`*worker-name*`.delegate_error_code`	Specifies the error status code used when the creation of the error page is entrusted to the Web server.
Definition of mapping between the URL pattern and worker	`JkMount`	Specifies some combination of workers specified in the URL pattern and `worker.list`.

Note:

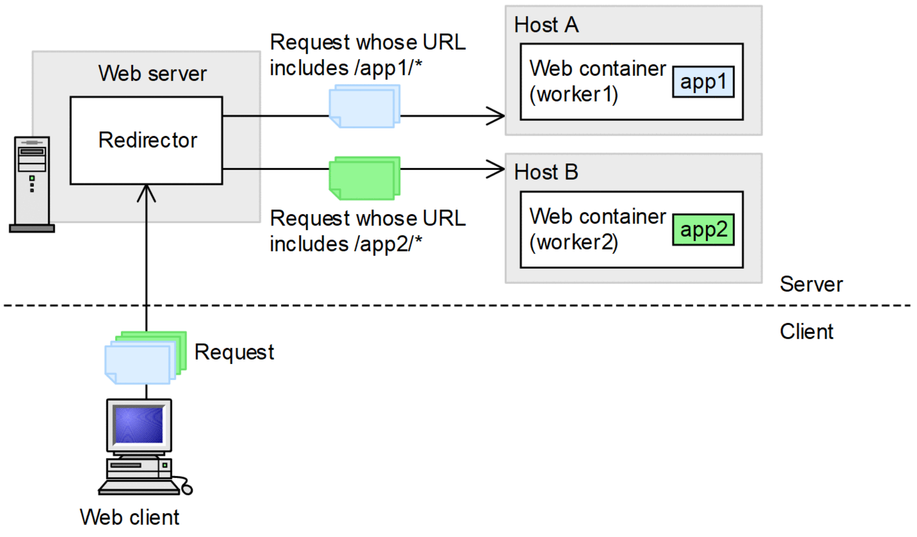
Define the name of the workers specified in the `worker.list` parameter in *worker-name*.

For details on the Easy Setup definition file and the parameters, see *4.3 Easy Setup definition file* in the *uCosminexus Application Server Definition Reference Guide*.

(3) Example settings

The following figure shows the distribution of requests by the URL patterns.

Figure 5–5: Example of distribution of requests by the URL patterns



In this example, the Web application `app1` is deployed on host A and the Web application `app2` is deployed on host B. By including the name of the Web application you want to process in the URL pattern of the request, the request with the

URL pattern `/app1/*` can be processed on the host A and the request with the URL pattern `/app2/*` can be processed on the host B. The worker name of the host A is `worker1` and the worker name of the host B is `worker2`.

An example of the Easy Setup definition file is described below. To distribute the requests to multiple Web containers, specify the worker processes of multiple Web containers as the distribution destinations, in the redirector registered in the Web server. Also, the URL pattern `/app1/*` are mapped with `worker1` and the URL pattern `/app2/*` are mapped with `worker2`.

Note that to implement this configuration, you must distribute requests to multiple Web systems.

Example of Easy Setup definition file

```
...
<param>
  <param-name>JkMount</param-name>
  <param-value>/app1/* worker1</param-value>
  <param-value>/app2/* worker2</param-value>
</param>
<param>
  <param-name>worker.list</param-name>
  <param-value>worker1, worker2</param-value>
</param>
<param>
  <param-name>worker.worker1.port</param-name>
  <param-value>8007</param-value>
</param>
<param>
  <param-name>worker.worker1.host</param-name>
  <param-value>hostA</param-value>
</param>
<param>
  <param-name>worker.worker1.type</param-name>
  <param-value>ajp13</param-value>
</param>
<param>
  <param-name>worker.worker1.cachesize</param-name>
  <param-value>64</param-value>
</param>
<param>
  <param-name>worker.worker2.port</param-name>
  <param-value>8007</param-value>
</param>
<param>
  <param-name>worker.worker2.host</param-name>
  <param-value>hostB</param-value>
</param>
<param>
  <param-name>worker.worker2.type</param-name>
  <param-value>ajp13</param-value>
</param>
<param>
  <param-name>worker.worker2.cachesize</param-name>
  <param-value>64</param-value>
</param>
...
```

5.3.4 Execution environment settings (When the Smart Composer functionality is not used)

Distributing requests by the URL patterns included in the HTTP requests enables you to execute only the specific processes in the Web container and to distribute requests to multiple Web containers according to the processing contents. Note that, when you use 'Distributing requests by URL pattern' method, as a principle, the requests are distributed in the Web application. Define the URL pattern as an operation of the redirector.

(1) Setup procedure

To set the distribution of requests by the URL pattern:

1. Define the worker in workers.properties.

   Specify the list of worker names, worker types (set `ajp13`), port number, and host name.

   The default value is defined in `workers.properties` that is provided by default. To use the default definition defined as a comment, delete the hash mark (`#`) at the beginning of the applicable line.

   For details on `workers.properties` (worker definition file), see *14.2.4 workers.properties (Worker definition file)*.

2. When using Cosminexus HTTP Server, define the mapping between the URL pattern and worker in `mod_jk.conf`. When using Microsoft IIS, define the mapping between the URL pattern and worker in uriworkermap.properties.

   If mapping is already defined, delete or replace the definition.

   For details on `mod_jk.conf` (redirector operation definition file for HTTP Server), see *14.2.2 mod_jk.conf (Redirector action definition file for Cosminexus HTTP Server)*.

   For details on `uriworkermap.properties` (mapping definition file for Microsoft IIS), see *14.2.3 uriworkermap.properties (Mapping definition file for Microsoft IIS)*.

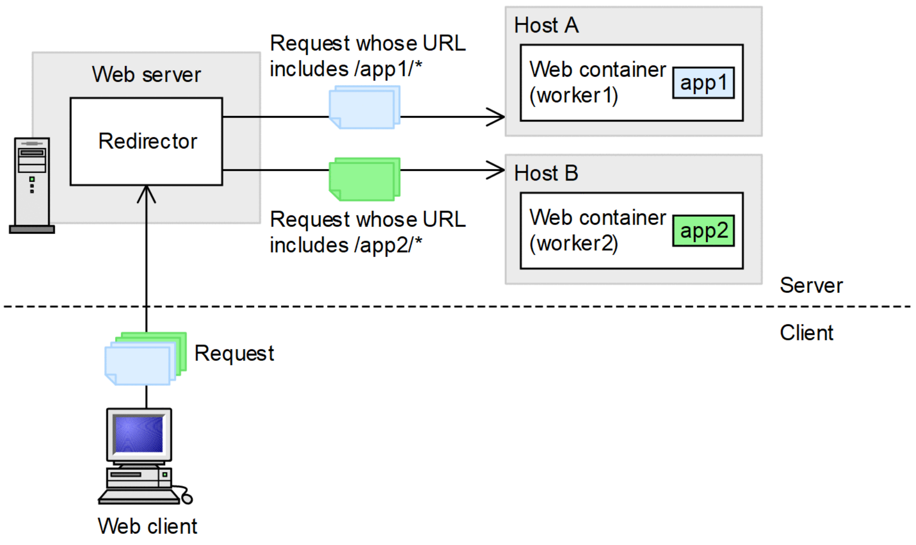
3. Set up the Web server environment and restart the Web server.

   For details on the Web server settings, see *Appendix D Precautions related to Cosminexus HTTP Server Settings* or *Appendix E Microsoft IIS Settings*.

(2) Example settings

The following figure shows the distribution of requests by the URL patterns.

Figure 5–6: Example of distribution of requests by the URL patterns



In this example, the Web application `app1` is deployed on host A and the Web application `app2` is deployed on host B. By including the name of the Web application you want to process in the URL pattern of the request, the request with the URL pattern `/app1/*` can be processed on the host A and the request with the URL pattern `/app2/*` can be processed on the host B. The worker name of host A is `worker1` and the worker name of host B is `worker2`.

An example of the `workers.properties` file is shown below. To distribute the requests to multiple Web containers, specify the worker processes of multiple Web containers as the distribution destinations, in the redirector registered in the Web server.

Example of workers.properties (In Windows)

```
worker.list=worker1, worker2

worker.worker1.port=8007
worker.worker1.host=hostA
worker.worker1.type=ajp13
worker.worker1.cachesize=64

worker.worker2.port=8007
worker.worker2.host=hostB
worker.worker2.type=ajp13
worker.worker2.cachesize=64
```

Example of workers.properties (In UNIX)

```
worker.list=worker1, worker2

worker.worker1.port=8007
worker.worker1.host=hostA
worker.worker1.type=ajp13

worker.worker2.port=8007
worker.worker2.host=hostB
worker.worker2.type=ajp13
```

Examples of the `mod_jk.conf` and `uriworkermap.properties` files are shown below. Here, the URL pattern `/app1/*` are mapped with `worker1` and the URL pattern `/app2/*` are mapped with `worker2`.

Example of mod_jk.conf (In Cosminexus HTTP Server)

```
JkMount /app1/* worker1
JkMount /app2/* worker2
```

Example of uriworkermap.properties (In Microsoft IIS)

```
/app1/*=worker1
/app2/*=worker2
```

5.4 Distributing requests by the round-robin format

This section explains the distribution of requests by the round-robin format.

The following table describes the organization of this section.

Table 5–6: Organization of this section (Distributing requests by the round-robin format)

Category	Title	Reference
Description	Overview of distributing requests by the round-robin format	*5.4.1*
	Examples of Request Distribution in the Round-robin Format	*5.4.2*
	Defining request distribution in the round robin format	*5.4.3*
Settings	Execution environment settings (When the Smart Composer functionality is used)	*5.4.4*
	Execution environment settings (When the Smart Composer functionality is not used)	*5.4.5*
Notes	Precautions related to request distribution in the round-robin format	*5.4.6*

Note:

There is no specific description of *Implementation* and *Operations* for this functionality.

When Web containers are deployed with a cluster configuration, by using the redirector, requests are distributed in round-robin format to the respective Web containers. By referencing the session ID appended to each request, the redirector distributes the requests so that the requests from the same Web client are always transferred to the same Web container. However, if the session cookie name is changed from the default `JSESSIONID` to another name, the operation of transferring requests from the same Web client to the same Web container is not guaranteed.

If the processing efficiency of the Web containers to which requests are distributed is different, you can adjust the proportion of load on each host by defining load parameters. When distributing requests in the round-robin format, as a prerequisite, you must deploy the same Web application on each Web container performing the distribution processing.

5.4.1 Overview of distributing requests by the round-robin format

For distribution of requests in the round-robin format with a cluster configuration, use a worker definition called *load balancer*. The list of worker processes that act as the distribution destinations is defined in the load balancer. Based on this definition, requests are distributed to the worker processes in the round-robin format.

An HTTP request is distributed. The HTTP requests belonging to the same session, however, are distributed to the same worker as the previous distribution destination.

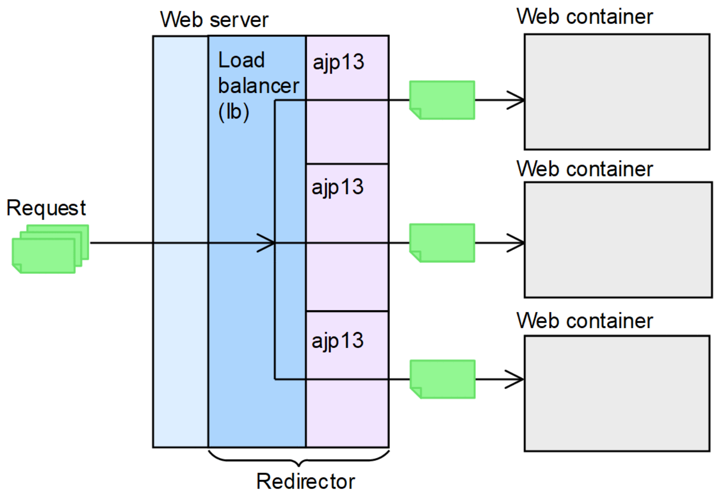
Reference note

During the Web server integration, if you specify request distribution with the round-robin format in the redirector settings, the worker name will be added in the session ID of `HttpSession`. Also, regardless of whether the settings for adding the server ID are specified, the server ID will not be added in the session ID of `HttpSession`.

5.4.2 Examples of request distribution in the round-robin format

The following figure shows an example of request distribution with the load balancer.

Figure 5–7: Example of request distribution with the load balancer



5.4.3 Defining request distribution in the round robin format

The following load balancer is already defined in a standard `workers.properties` file.

```
#worker.list=loadbalancer1

#worker.loadbalancer1.type=lb
#worker.loadbalancer1.balanced_workers=worker1, worker2
```

Set the type of the worker in `worker.loadbalancer1.type`. Set the name of the worker process to which the request is to be distributed in `worker.loadbalancer1.balanced_workers`. In `workers.properties`, `lb` and `worker1, worker2` are defined respectively as `loadbalancer1`.

This definition is, however, described as a comment. Consequently, when using the above-mentioned load balancer, delete the hash mark (`#`) present at the beginning of the corresponding rows of `workers.properties`.

You can define the ratio of request distribution in `lbfactor` parameter of each worker definition that is the distribution target. Larger the `lbfactor`, larger is the ratio of requests transferred to the worker process.

For example, when requests are distributed to two worker processes, namely worker1 and worker2, the ratio of request distribution is defined as follows in the `lbfactor` parameter of the worker definition:

- `lbfactor` parameter of worker1: 2.0
- `lbfactor` parameter of worker2: 1.0

In this case, `worker1` is in charge of double the number of Web clients of worker2.

5.4.4 Execution environment settings (When the Smart Composer functionality is used)

By defining the list of workers that act as the distribution destinations in the load balancer, requests are distributed to the workers in the round-robin format.

If you set a load balancing value in each worker that acts as the distribution destination and define the request distribution ratio, you can adjust the proportion of load on each host. Since the redirector distributes requests with the round-robin format for each HTTP request at this ratio, higher the ratio for a worker the greater will be the proportion of forwarded requests. However, the HTTP requests belonging to the same session are distributed to the same worker as the last time.

(1) Setup procedure

To set the distribution of requests by the round-robin format, specify the settings using the following procedure:

1. Define the load balancer and worker in workers.properties.

   Definitions for the load balancer

      Specify the list of worker names, worker types (specify `lb`), and list of workers for load balancing.

   Definitions for each worker

      Specify the worker types (specify `ajp13`), port number, host name, and the load balancing value.

   The default value is defined in `workers.properties` that is provided by default. To use the default definition defined as a comment, delete the hash mark (#) at the beginning of the applicable line.

   For details on `workers.properties` (worker definition file), see *14.2.4 workers.properties (Worker definition file)*.

2. Define the mapping between the URL pattern and worker in mod_jk.conf.

   If mapping is already defined, delete or replace the definition.

   For details on `mod_jk.conf` (redirector operation definition file for HTTP Server), see *14.2.2 mod_jk.conf (Redirector action definition file for Cosminexus HTTP Server)*.

3. Set up the Web server environment and restart the Web server.

   For details on the Web server settings, see *Appendix D Precautions related to Cosminexus HTTP Server Settings*.

(a) Notes

- When you use load balancing with the redirector and if a failure is detected in a worker, the worker is excluded from the choices for redirect destination workers for 60 seconds from the detection of failure. Therefore, even if the failure is recovered, the worker might not be used for a maximum of 60 seconds.
- In UNIX, when the server processes of Cosminexus HTTP Server are generated or destroyed according to the load, multiple server processes are allocated by the worker defined first in `workers.properties`. Also, even if the number of server processes is fixed, the server process to which the request is allocated is uncertain, and therefore, if you specify the same load balancing value, round-robin might not occur. Unless destroyed, the server processes are allowed to increase according to the load, and therefore you must specify a directive in such a way so that the server processes are not generated or destroyed in a short time.

  Specify the `httpsd.conf` directive of Cosminexus HTTP Server in such a way so that the following conditions are fulfilled:

Conditions	Meaning
`MaxSpareServers` ≥ `MaxClients`	The server processes increase up to `MaxClients` and stay resident even after the processing ends.

Conditions	Meaning
`MaxRequestsPerChild 10000`	The HTTP request is processed 10,000 times and then the server process is terminated to refresh the operations (10,000 is the recommended value). Specify an adequately large value for the number of J2EE servers that act as the distribution destinations.
`StartServers = MaxClients`	You specify this condition to start all the server processes first.

(b) Example specification of directive

```
StartServers 256
MaxClients 256
MaxSpareServers 256
MaxRequestsPerChild 10000
```

(2) Settings in workers.properties and mod_jk.conf

Define the settings for distributing requests by the round-robin format in `workers.properties` and `mod_jk.conf`. The following table lists the keys defined in `workers.properties` and `mod_jk.conf`:

Table 5–7: Keys defined in workers.properties and mod_jk.conf (When distributing requests by the round-robin format)

Types of files	Key name	Description
`workers.properties`	`worker.list`	Specifies a list of one or multiple worker names.
	`worker.`*worker-name*`.host`	Specifies the worker host name or IP address.
	`worker.`*worker-name*`.port`	Specifies the worker port number.
	`worker.`*worker-name*`.type`	Specifies the worker type. Specify `lb` in the load balancer and `ajp13` in the worker for load balancing.
	`worker.`*worker-name*`.balanced_workers`	Specifies the list of workers for load balancing.
	`worker.`*worker-name*`.lbfactor`	Specifies the load balancing value.
	`worker.`*worker-name*`.cachesize`	Specifies the number of worker connections that are reused in the redirector. This key can only be specified in Windows.
	`worker.`*worker-name*`.receive_timeout`	Specifies the communication timeout value.
	`worker.`*worker-name*`.delegate_error_code`	Specifies the error status code used when the creation of the error page is entrusted to the Web server.
`mod_jk.conf`	`JkMount`	Specifies some combination of workers specified in the URL pattern and `worker.list`.

Note:

In *worker-name*, define the worker name specified in the `worker.list` key or the `worker.`*worker-name*`.balanced_workers` key.

The following table lists the parameters that you can specify for each worker type:

Table 5–8: Keys that can be specified for each worker type

Key name	Worker type (value specified in worker.worker-name.type)	
	Load balancer (Specify `lb`)	Worker (Specify `ajp13`)
`worker.`*worker-name*`.host`	--	Y
`worker.`*worker-name*`.port`	--	Y
`worker.`*worker-name*`.type`	Y	Y
`worker.`*worker-name*`.balanced_workers`	Y	--
`worker.`*worker-name*`.lbfactor`	--	O
`worker.`*worker-name*`.cachesize`	--	O
`worker.`*worker-name*`.receive_timeout`	--	O
`worker.`*worker-name*`.delegate_error_code`	--	O

Legend:

Y: Can be specified

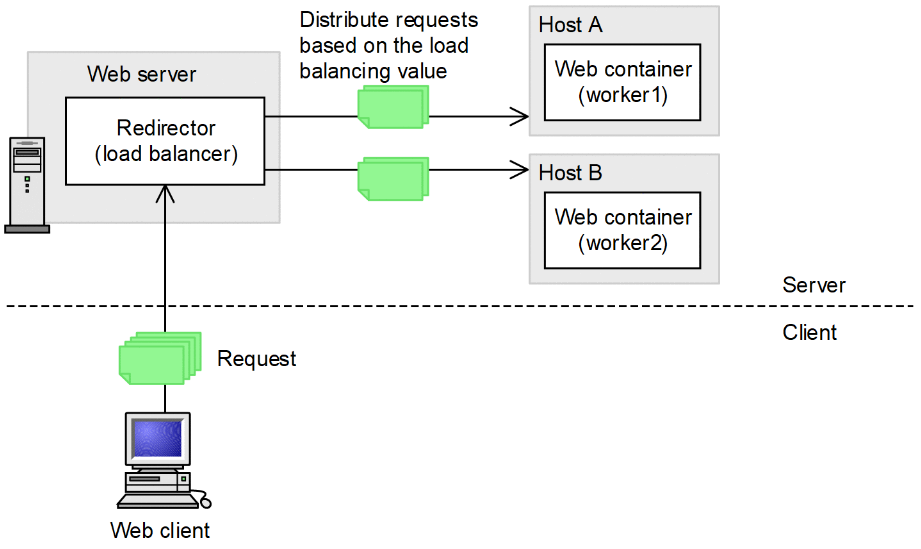
--: Cannot be specified

O: Can be optionally specified

(3) Example settings

The following figure shows the distribution of requests by the round-robin format.

Figure 5–8: Example of distribution of requests by the round-robin format



In this example, the requests under `/examples` are equally distributed on host A and on host B. The worker name in host A is `worker1` and the worker name in host B is `worker2`.

An example of the `workers.properties` file is shown below. The load balancer and worker are defined here. Since the requests are distributed at an equal rate, `1` is specified as the load balancing value for both `worker1` and `worker2`.

Example of workers.properties (In Windows)

```
worker.list=loadbalancer1

worker.loadbalancer1.balanced_workers=worker1, worker2
worker.loadbalancer1.type=lb

worker.worker1.port=8007
worker.worker1.host=hostA
worker.worker1.type=ajp13
worker.worker1.cachesize=64
worker.worker1.lbfactor=1

worker.worker2.port=8007
worker.worker2.host=hostB
worker.worker2.type=ajp13
worker.worker2.cachesize=64
worker.worker2.lbfactor=1
```

Example of workers.properties (In UNIX)

```
worker.list=loadbalancer1

worker.loadbalancer1.balanced_workers=worker1, worker2
worker.loadbalancer1.type=lb

worker.worker1.port=8007
worker.worker1.host=hostA
worker.worker1.type=ajp13
worker.worker1.lbfactor=1

worker.worker2.port=8007
worker.worker2.host=hostB
worker.worker2.type=ajp13
worker.worker2.lbfactor=1
```

An example of the `mod_jk.conf` file is shown below. The load balancer name `loadbalancer1` is specified here.

Example of mod_jk.conf

```
JkMount /examples/* loadbalancer1
```

5.4.5 Execution environment settings (When the Smart Composer functionality is not used)

By defining the list of workers that act as the distribution destinations in the load balancer, the requests are distributed to the workers with the round-robin format.

If you specify a load balancing value in each worker that acts as the distribution destination and define the request distribution ratio, you can adjust the proportion of load on each host. Since the redirector distributes requests with the round-robin format for each HTTP request at this ratio, higher the ratio for a worker the greater will be the proportion of forwarded requests. However, the HTTP requests belonging to the same session are distributed to the same worker of the last time.

(1) Setup procedure

To set the distribution of requests by the round-robin format, use the following procedure:

1. Define the load balancer and worker in workers.properties.

   Definitions of load balancer

   Specify the list of worker names, worker types (specify `lb`), and list of workers for load balancing.

   Definitions for each worker

   Specify the worker types (specify `ajp13`), port number, host name, and the load balancing value.

   The default value is defined in `workers.properties` that is provided as standard. To use the default definition defined as a comment, delete the hash mark (#) at the beginning of the applicable line.

   For details on `workers.properties` (worker definition file), see *14.2.4 workers.properties (Worker definition file)*.

2. When using Cosminexus HTTP Server, define the mapping between the URL pattern and worker in mod_jk.conf. When using Microsoft IIS, define the mapping between the URL pattern and worker in uriworkermap.properties.

   If a mapping is already defined, delete the definition or replace the mapping.

   For details on `mod_jk.conf` (redirector operation definition file for HTTP Server), see *14.2.2 mod_jk.conf (Redirector action definition file for Cosminexus HTTP Server)*.

   For details on `uriworkermap.properties` (mapping definition file for Microsoft IIS), see *14.2.3 uriworkermap.properties (Mapping definition file for Microsoft IIS)*.

3. Set up the Web server environment and restart the Web server.

   For details on the Web server settings, see *Appendix D Precautions related to Cosminexus HTTP Server Settings* or *Appendix E Microsoft IIS Settings*.

(a) Notes

- When you perform load balancing in redirector and if a failure is detected in a worker, that worker is excluded from the choices of the redirect destination workers for the period of 60 seconds after the failure is detected. Therefore, even if the failure is recovered, the worker might not be used for a maximum period of 60 seconds.
- When you use Microsoft IIS and specify multiple worker processes on which the redirector is running, and if two or more workers are set, many requests are allocated to the worker defined initially in workers.properties that is the initial redirect destination.

  Also, allocation of a request to a work process is dependent on the Microsoft IIS control and therefore, even if the same load balancing value is specified, there are cases when a round-robin is not performed.

  In such cases, by setting the number of application pools to one, a round robin can be performed even for the first redirect allocation destination.
- In UNIX, when the server processes of Cosminexus HTTP Server are generated or destroyed according to the load, more requests are allocated by the worker defined first in workers.properties. Also, even if the number of server processes is fixed, the server process to which the request is allocated is uncertain, and therefore if you specify the same load balancing value, the round-robin might not occur. Unless the server process is destroyed, the server process is allowed to increase according to the load, so you must specify a directive in such a way so that the server processes are not generated or destroyed in a short time.

  Specify the httpsd.conf directive of Cosminexus HTTP Server in such a way so that the following conditions are fulfilled:

Conditions	Meaning
`MaxSpareServers ≥MaxClients`	The server processes increase up to `MaxClients` and stay resident even after the processing ends.
`MaxRequestsPerChild 10000`	The HTTP request is processed 10,000 times and then the server process is terminated to refresh the operations (10,000 is the recommended value). Specify an adequately large value for the number of J2EE servers that act as the distribution destinations.
`StartServers＝MaxClients`	You specify this condition to start all the server processes first.

(b) Example specification of directive

```
StartServers 256
MaxClients 256
MaxSpareServers 256
MaxRequestsPerChild 10000
```

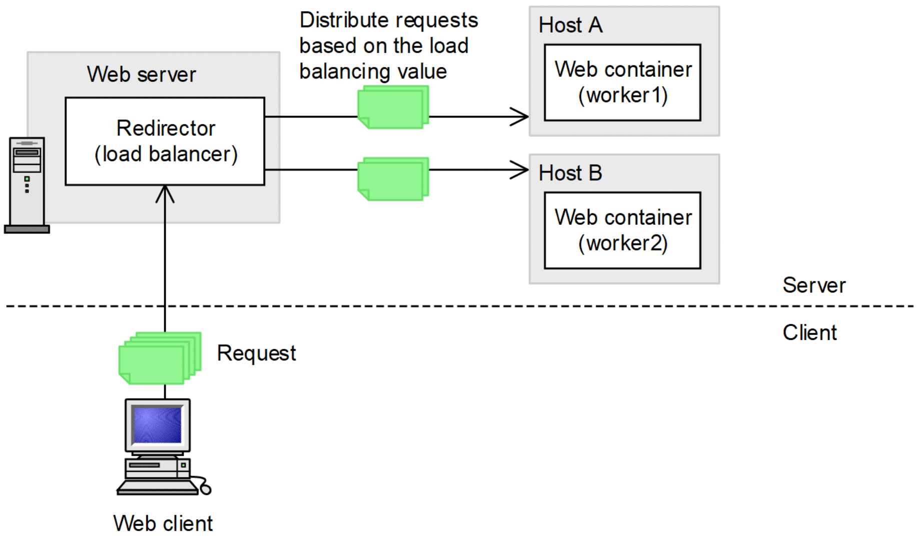
(2) Settings in workers.properties and mod_jk.conf

The settings in `workers.properties` and `mod_jk.conf` are the same settings that are specified when Smart Composer is used. For details, see *5.4.4(2) Settings in workers.properties and mod_jk.conf*.

(3) Example settings

The following figure shows the distribution of requests by the round-robin format.

Figure 5–9: Example of distribution of requests by the round robin format



In this example, the requests under `/examples` are equally distributed on host A and on host B. The worker name in host A is `worker1` and the worker name in host B is `worker2`.

An example of the `workers.properties` file is shown below. The load balancer and worker will be defined here. Since the requests are distributed at an equal rate, `1` is specified as the load balancing value for both `worker1` and `worker2`.

Example of workers.properties (In Windows)

```
worker.list=loadbalancer1

worker.loadbalancer1.balanced_workers=worker1, worker2
worker.loadbalancer1.type=lb

worker.worker1.port=8007
worker.worker1.host=hostA
worker.worker1.type=ajp13
worker.worker1.cachesize=64
worker.worker1.lbfactor=1

worker.worker2.port=8007
worker.worker2.host=hostB
worker.worker2.type=ajp13
worker.worker2.cachesize=64
worker.worker2.lbfactor=1
```

Example of workers.properties (In UNIX)

```
worker.list=loadbalancer1

worker.loadbalancer1.balanced_workers=worker1, worker2
worker.loadbalancer1.type=lb

worker.worker1.port=8007
worker.worker1.host=hostA
worker.worker1.type=ajp13
worker.worker1.lbfactor=1

worker.worker2.port=8007
worker.worker2.host=hostB
worker.worker2.type=ajp13
worker.worker2.lbfactor=1
```

Examples of the `mod_jk.conf` and `uriworkermap.properties` files are shown below. The load balancer name `loadbalancer1` will be specified here.

Example of mod_jk.conf (in Cosminexus HTTP Server)

```
JkMount /examples/* loadbalancer1
```

Example of uriworkermap.properties (In Microsoft IIS)

```
/examples/*=loadbalancer1
```

5.4.6 Precautions related to request distribution in the round-robin format

The precautions related to request distribution in the round-robin format for the redirector are as follows:

- **Sending requests to the Web container when the J2EE application is not running**
  When distributing requests with the round-robin format, requests are sent to the Web container even when the J2EE application is not running. Therefore, you must isolate all the Web containers from the system, and then implement the changes in J2EE applications.

- **Disabling of health check by the load balancer**

  When the load balancer and request distribution with the round-robin format are combined and used, requests are normally forwarded to the Web container by the redirector even if a failure occurs in the J2EE server. As a result, the failure on the J2EE server cannot be detected in the load balancer and the Web container cannot be monitored.

5.5 Distributing requests by the POST data size

This section explains the distribution of requests by the POST data size.

The following table describes the organization of this section.

Table 5–9: Organization of this section (Distributing requests by the POST data size)

Category	Title	Reference
Description	Overview of distributing requests by the POST data size	*5.5.1*
	Examples of Distributing Requests by the POST Data Size	*5.5.2*
	Request distribution conditions	*5.5.3*
	Definition for distributing requests by the POST data size	*5.5.4*
Settings	Execution environment settings (When the Smart Composer functionality is used)	*5.5.5*
	Execution environment settings (When the Smart Composer functionality is not used)	*5.5.6*

Note:

There is no specific description of *Implementation*, *Operations*, and *Notes* for this functionality.

5.5.1 Overview of distributing requests by the POST data size

When Web containers are deployed with a cluster configuration, by using the redirector, requests are distributed by the POST data size to the respective Web containers. If you use this functionality, you can forward the very long POST data requests with a long processing time to the specific Web containers. As a result, you can avoid the decrease in the throughput of requests other than the very long POST data requests or can avoid the decrease in response time. When distributing requests in this method, as a prerequisite, you must deploy a Web application on each Web container performing the distribution processing. However, the Web applications deployed on each J2EE server is not required to be the same.

For distributing requests by the POST data size in the cluster configuration, use the worker definition called *POST request-distributing worker*. The list of worker processes that act as the distribution destinations is defined in the POST request-distributing worker. Based on this definition, requests are distributed to the worker processes by the POST data size. The worker process that acts as the distribution destination of the POST request-distributing worker is called *POST request-forwarding worker*.

The HTTP requests are distributed to the POST request-forwarding worker.

Important note

Even when the session is managed with control based on HTTP Cookie or URL rewriting, if distribution by POST data size is specified, the request distribution destination is determined by the POST data size. Therefore, the session ID of the `HttpSession` is not inherited even when the request is from the same client.

For example, if `HttpSession` session ID is generated on J2EE server 1 and the request is forwarded to J2EE server 2, a new `HttpSession` session ID is generated on J2EE server 2. In this case, if J2EE server 1 is accessed again, the `HttpSession` session ID generated on J2EE server 2 is being used in the client, so a new `HttpSession` session ID is generated on J2EE server 1. Therefore, the session of the `HttpSession` is not inherited.

Note that when the `HttpSession` session ID is generated on J2EE server 1 and a request is forwarded to J2EE server 2, in that case if `HttpSession` session ID is not generated on J2EE server 2, the `HttpSession` session ID of J2EE server 1 will be used as it is, when you re-access the J2EE server 1.

5.5.2 Examples of distributing requests by the POST data size

For distributing requests by the POST data size, the value set as the upper limit of the POST data size will differ depending on whether the request forwarded to the POST request-distributing worker can be limited or not.

- **When the request forwarded to the POST request-distributing worker can be limited**

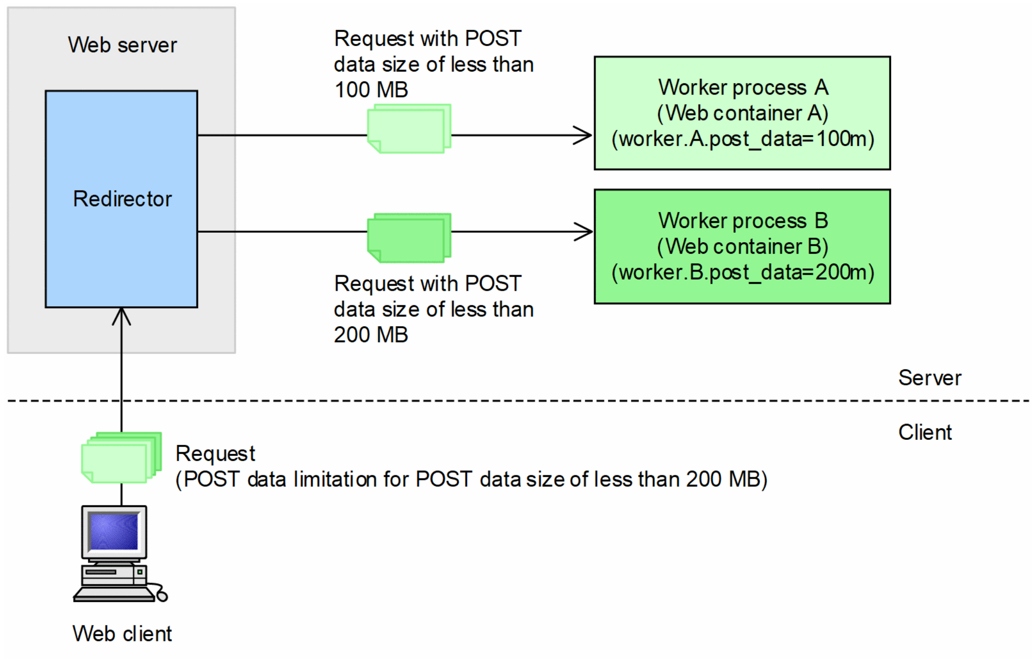
  The requests fulfilling the following conditions are forwarded to the POST request-distributing worker:

  - The request is a POST data request.
  - The POST data size of the request is less than 200 MB.

  If the request can be limited, the range of very long POST data that you want to process can also be limited. Set an upper limit for each request-forwarding worker so that a request of a specific POST data size is forwarded to a worker that is processing a very long POST data request.

  The following figure shows a distribution example of requests by the POST data size when the requests can be limited:

  Figure 5–10: Example of distribution of requests by the POST data size (When the request can be limited)


Web server
Redirector
Request with POST data size of less than 100 MB
Worker process A (Web container A) (worker.A.post_data=100m)
Worker process B (Web container B) (worker.B.post_data=200m)
Request with POST data size of less than 200 MB
Server
Client
Request (POST data limitation for POST data size of less than 200 MB)
Web client

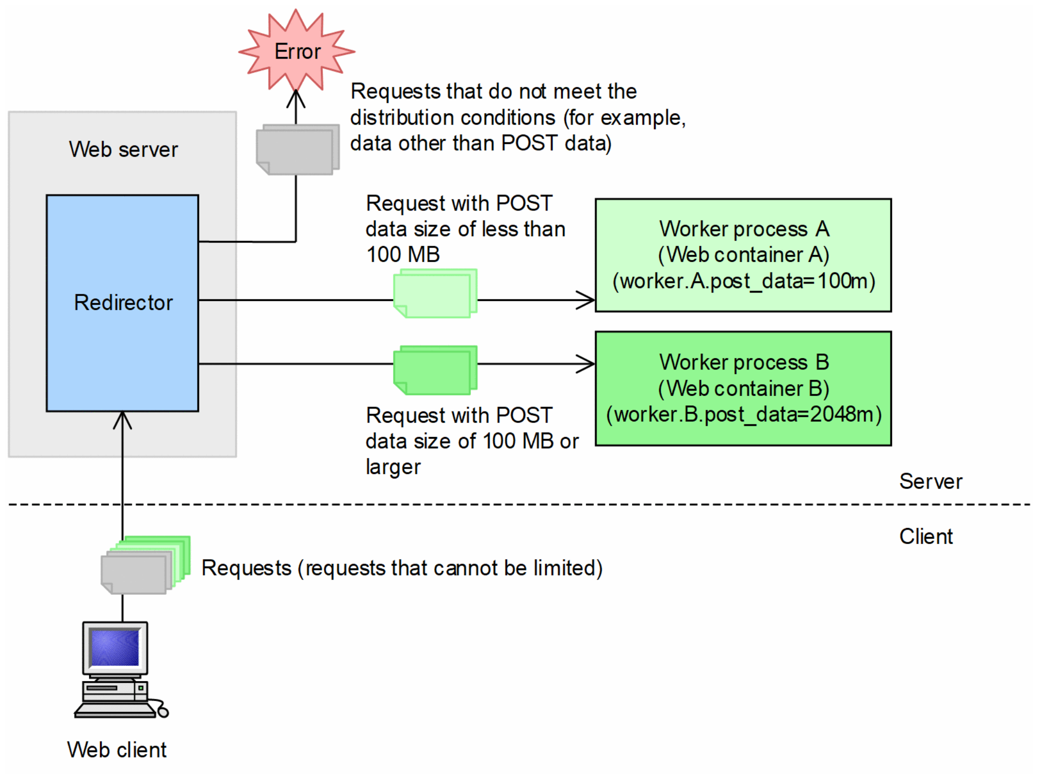
  In this figure, two POST request-forwarding workers are prepared. An upper limit is set for the POST data size in such a way so that the requests with POST data size between 100 MB and 200 MB are forwarded to worker process B. If the POST data size of the request is less than the upper limit, the request is distributed to the respective POST request-forwarding worker. If the POST data size of a request is applicable to multiple POST request-forwarding workers, the request is distributed to the POST request-forwarding worker with the smallest POST data size upper limit. For example, a request with POST data size 80 MB is applicable to both workers, but is distributed to worker process A.

- **When the request forwarded to the POST request-distributing worker cannot be limited**

If the request cannot be limited, set the maximum value as the upper limit for the POST data size of the worker that processes very long POST data.

The following figure shows a distribution example of requests by the POST data size when the request cannot be limited:

Figure 5–11: Example of distribution of requests by the POST data size (When the request cannot be limited)


Error
Requests that do not meet the distribution conditions (for example, data other than POST data)
Web server
Redirector
Request with POST data size of less than 100 MB
Worker process A (Web container A) (worker.A.post_data=100m)
Worker process B (Web container B) (worker.B.post_data=2048m)
Request with POST data size of 100 MB or larger
Server
Client
Requests (requests that cannot be limited)
Web client

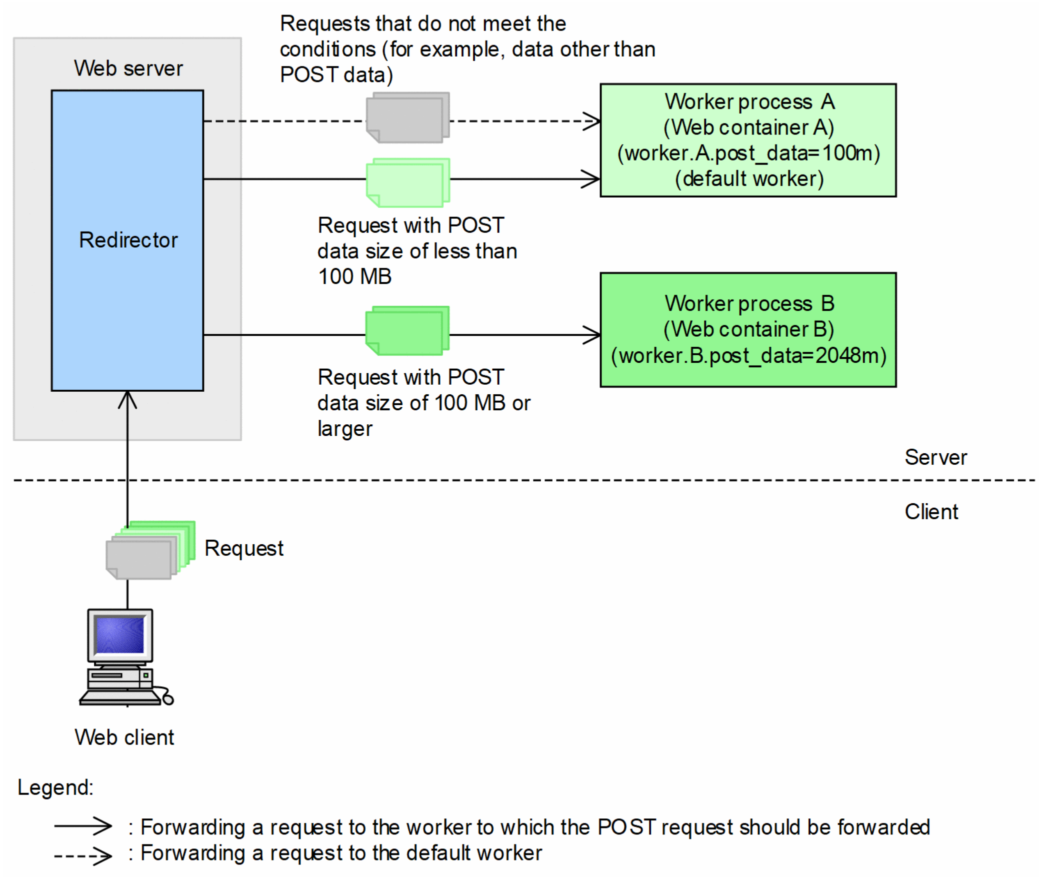
In this figure, 2 POST request-forwarding workers are prepared and upper limit is set for the POST data size in each of the workers. Maximum value is set in the upper limit for the POST data size of worker process B in such a way so that all the requests of very long POST data are processed in worker process B. POST data requests that are more than the upper limit of worker process A (100 MB or more) are forwarded to the worker process B. Note that in this case, if non-POST data requests and requests that cannot reference the POST data size are forwarded, these requests are not distributed by the request-distributing workers, so an error occurs and the redirector returns the error status code 400.

While distributing requests by the POST data size, if requests not fulfilling the request distribution conditions are forwarded to the POST request-distributing worker, and error occurs and the redirector returns the error status code 400. For details on the request distribution conditions, see *5.5.3 Request distribution conditions*.

If you want to process requests that do not fulfill the request distribution conditions, you must set up the worker process to forward that request. The worker process that forwards requests which do not fulfill the request distribution conditions is called a *default worker*. The default worker settings are optional.

The following figure shows an example in which the requests cannot be limited, and the requests not fulfilling the request distribution conditions are forwarded to the default worker:

Figure 5–12: Example of distribution of requests by the POST data size (When the default worker is set)



Requests that do not meet the conditions (for example, data other than POST data)
Web server
Redirector
Worker process A (Web container A) (worker.A.post_data=100m) (default worker)
Request with POST data size of less than 100 MB
Worker process B (Web container B) (worker.B.post_data=2048m)
Request with POST data size of 100 MB or larger
Server
Client
Request
Web client
Legend:
: Forwarding a request to the worker to which the POST request should be forwarded
: Forwarding a request to the default worker

In this figure, settings are specified in such a way so that the requests that do not fulfill the request distribution conditions are forwarded to the worker process A of the default worker.

5.5.3 Request distribution conditions

The requests distributed to the POST request-forwarding worker must fulfill the following conditions:

Conditions of the requests distributed to the POST request-forwarding worker

- The request method is POST.
- The request has a Content-Length header (body data is not in chunk format).
- The value of the Content-Length header of the request is less than the POST data size set in the worker.

A request that does not fulfill even one of these conditions is distributed to the default worker. If the default worker is not set, an error occurs and an error with error status code 400 is returned.

5.5.4 Definition for distributing requests by the POST data size

The following workers that are used for distributing requests by the POST data size are already defined in a standard `workers.properties` file.

```
#worker.list=postsizelb1
#worker.postsizelb1.type=post_size_lb
#worker.postsizelb1.post_size_workers=worker1,worker2
#worker.postsizelb1.default_worker=worker1
```

Set the type of the worker in `worker.postsizelb1.type`. In `worker.postsizelb1.post_size_workers`, set the worker process name of the POST request-forwarding worker that forms the target of distribution. In `worker.postsizelb1.default_worker`, set the default worker. In `workers.properties`, define `post_size_lb` in the worker type, `worker1` and `worker2` in the POST request-forwarding worker, and `worker1` in the default worker as `postsizelb1`.

This definition is, however, described as a comment. Therefore, when using the POST request-distributing worker of this definition, delete the hash mark (#) at the beginning of the applicable line in `workers.properties`.

Specify the POST data size for distributing the request in the `post_data` parameter of the worker definition in `workers.properties`.

For example, use the `postsizelb1` definition that is provided by default to define the following POST data size in the 2 POST request-forwarding workers named `worker1` and `worker2` respectively:

- `post_data` parameter for `worker1`: 100m
- `post_data` parameter for `worker2`: 200m

In this case, the requests of size less than 100 MB are distributed to `worker1` and the requests of size between 100 MB to 200 MB are distributed to `worker2`. If the request-distributing worker distributes a request of size 200 MB or more, the request is forwarded to `worker1` that is set as the default worker.

5.5.5 Execution environment settings (When the Smart Composer functionality is used)

By defining the list of workers that act as the distribution destinations in the POST request-distributing worker, requests are distributed to the workers by the POST data size.

Set the upper limit for the POST data size and define the request distribution destination in the POST request-forwarding worker that acts as the distribution destination. As a result, request processing of very long POST data size with a long processing time is distributed to a specific host. The redirector distributes the requests to each HTTP request with the upper limit of the POST data size, so you can avoid the decrease in throughput of requests other than the very long POST data requests and can avoid the decrease in response time. Note that when the upper limit of the POST data size is specified, the value of the POST data size is given priority even if the HTTP request belongs to the same session.

(1) Setup procedure

To specify settings for distributing requests by the POST data size, use the following procedure:

1. Define the POST request-distributing worker and POST request-forwarding worker in workers.properties.

   Definitions for POST request-distributing worker

   Specify the list of worker names, worker types (specify `post_size_lb`), and list of workers for distribution by the POST data size. As needed, set the default worker.

Definitions for each POST request-forwarding worker

Specify the worker types (specify `ajp13`), port number, host name, and the upper limit of the POST data size.

The default value is defined in `workers.properties` that is provided by default. To use the default definition defined as a comment, delete the hash mark (#) at the beginning of the applicable line.

For details on `workers.properties` (worker definition file), see *14.2.4 workers.properties (Worker definition file)*.

2. Define the mapping between the URL pattern and worker in mod_jk.conf.
   If a mapping is defined, delete the definition or replace the mapping.
   For details on `mod_jk.conf` (redirector operation definition file for HTTP Server), see *14.2.2 mod_jk.conf (Redirector action definition file for Cosminexus HTTP Server)*.

3. Set up the Web server environment and restart the Web server.
   For details on the Web server settings, see *Appendix D Precautions related to Cosminexus HTTP Server Settings*.

(2) Settings in workers.properties and mod_jk.conf

Define the settings for distributing requests by the POST data size in `workers.properties` and `mod_jk.conf`. The following table lists the keys defined in `workers.properties` and `mod_jk.conf`.

Table 5–10: Keys defined in workers.properties and mod_jk.conf (When distributing requests by the POST data size)

Types of files	Key name	Description
`workers.properties`	`worker.list`	Specifies a list of one or multiple worker names.
	`worker.`*worker-name*`.host`	Specifies the worker host name or IP address.
	`worker.`*worker-name*`.port`	Specifies the worker port number.
	`worker.`*worker-name*`.type`	Specifies the worker type. Specify `post_size_lb` in the POST request-distributing worker and `ajp13` in the POST request-forwarding worker that forms the target for distribution.
	`worker.`*worker-name*`.post_size_workers`	Specifies the list of workers that form the target of distribution by the POST data size.
	`worker.`*worker-name*`.post_data`	Specifies the upper limit (bytes) of the POST data size for the request.
	`worker.`*worker-name*`.default_worker`	Specifies the worker (default worker) for forwarding the requests if the worker applicable to the request distribution destination is not in the POST request-forwarding worker within the cluster configuration.
	`worker.`*worker-name*`.cachesize`	Specifies the number of worker connections that are reused in the redirector. This key can only be specified in Windows.
	`worker.`*worker-name*`.receive_timeout`	Specifies the communication timeout value.
	`worker.`*worker-name*`.delegate_error_code`	Specifies the error status code used when the creation of the error page is entrusted to the Web server.
`mod_jk.conf`	`JkMount`	Specifies some combination of workers specified in the URL pattern and `worker.list`.

Note:

In *worker-name*, define the worker name specified in the `worker.list` key or `worker.`*worker-name*`.post_size_workers` key.

The following table lists the keys that you can specify for each worker type:

Table 5–11: Keys that can be specified for each worker type

Key name	Worker type (value specified in worker.worker-name.type key)	
	POST request-distributing worker (Specify `post_size_lb`)	POST request-forwarding worker (Specify `ajp13`)
`worker.`*worker-name*`.host`	--	Y
`worker.`*worker-name*`.port`	--	Y
`worker.`*worker-name*`.type`	Y	Y
`worker.`*worker-name*`.post_size_workers`	Y	--
`worker.`*worker-name*`.post_data`	--	Y
`worker.`*worker-name*`.default_worker`	O	--
`worker.`*worker-name*`.cachesize`	--	O
`worker.`*worker-name*`.receive_timeout`	--	O
`worker.`*worker-name*`.delegate_error_code`	--	O

Legend:

Y: Can be specified

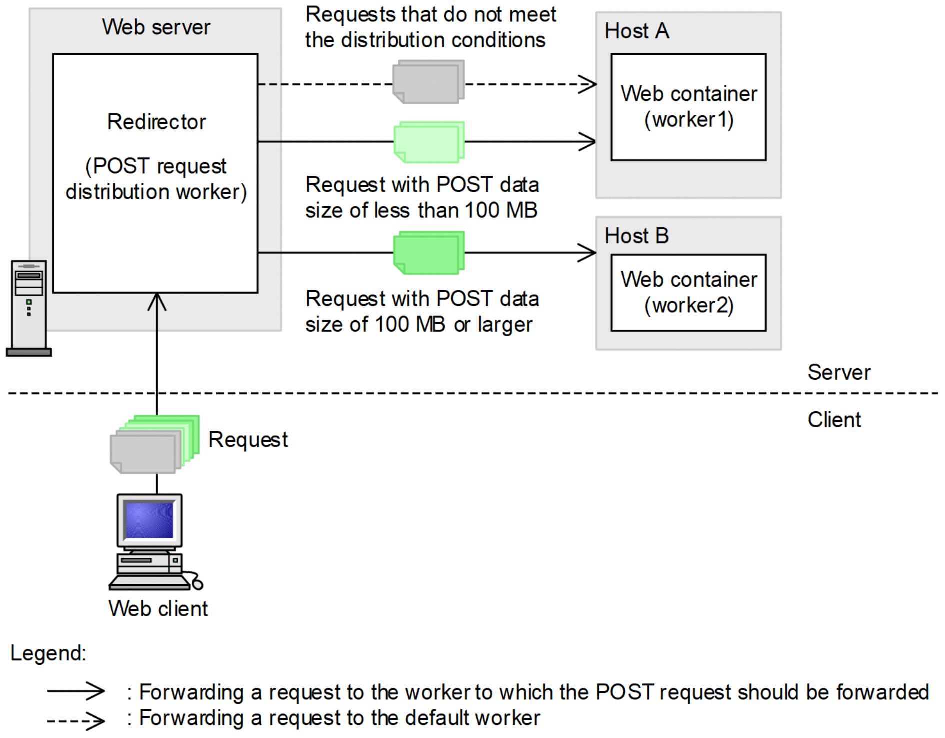
--: Cannot be specified

O: Can be optionally specified

(3) Example settings

The figure below shows the distribution of requests by POST data size. This figure shows an example when the requests cannot be limited:

Figure 5–13: Example of distribution of requests by the POST data size


Web server
Redirector
(POST request distribution worker)
Requests that do not meet the distribution conditions
Request with POST data size of less than 100 MB
Request with POST data size of 100 MB or larger
Host A
Web container (worker1)
Host B
Web container (worker2)
Server
Client
Request
Web client
Legend:
: Forwarding a request to the worker to which the POST request should be forwarded
: Forwarding a request to the default worker

In this example, among the requests under `/examples`, the requests with POST data size of less than 100 MB are distributed to host A and the requests with POST data size between 100 MB and 200 MB are distributed to host B. The requests that do not fulfill the request distribution conditions are distributed to the host A that is set as the default worker. The worker name of the host A is `worker1` and the worker name of the host B is `worker2`. For details on the request distribution conditions, see *5.5.3 Request distribution conditions*.

An example of the `workers.properties` file is described here. The POST request-distributing worker, POST request-forwarding worker, and default worker is defined here. As the upper limit of POST data size, `100m` is specified for `worker1` and `2048m` is specified for `worker2` (maximum value of the upper limit for POST data size). In the default worker, `worker1` is specified.

Example of workers.properties (In Windows)

```
worker.list=postsizelb1

worker.postsizelb1.post_size_workers=worker1, worker2
worker.postsizelb1.type=post_size_lb
worker.postsizelb1.default_worker=worker1

worker.worker1.port=8007
worker.worker1.host=hostA
worker.worker1.type=ajp13
worker.worker1.cachesize=64
worker.worker1.post_data=100m

worker.worker2.port=8007
worker.worker2.host=hostB
worker.worker2.type=ajp13
worker.worker2.cachesize=64
worker.worker2.post_data=2048m
```

Example of workers.properties (In UNIX)

```
worker.list=postsizelb1

worker.postsizelb1.post_size_workers=worker1, worker2
worker.postsizelb1.type=post_size_lb
worker.postsizelb1.default_worker=worker1

worker.worker1.port=8007
worker.worker1.host=hostA
worker.worker1.type=ajp13
worker.worker1.post_data=100m

worker.worker2.port=8007
worker.worker2.host=hostB
worker.worker2.type=ajp13
worker.worker2.post_data=2048m
```

An example of the `mod_jk.conf` file is shown below. POST request-distributing worker `postsizelb1` is specified here.

Example of mod_jk.conf

```
JkMount /examples/* postsizelb1
```

Reference note

You can also set the upper limit for the POST data size in the `LimitRequestBody` directive of `httpsd.conf` (Cosminexus HTTP Server definition file). For details on the `LimitRequestBody` directive, see the *uCosminexus Application Server HTTP Server User Guide*.

5.5.6 Execution environment settings (When the Smart Composer functionality is not used)

By defining the list of workers that act as the distribution destinations in the POST request-distributing worker, requests are distributed to the workers by the POST data size.

Set the upper limit for the POST data size and define the request distribution destination in the POST request-forwarding worker that acts as the distribution destination. As a result, request processing of very long POST data size with a long processing time is distributed to a specific host. The redirector distributes the requests to each HTTP request with the upper limit of the POST data size, so you can avoid the decrease in throughput of requests other than the very long POST data requests and can avoid the decrease in the response time. Note that when the upper limit of the POST data size is specified, the value of the POST data size is given priority even if the HTTP request belongs to the same session.

Important note

When integrating with Microsoft IIS, you cannot specify settings for distributing requests by the POST data size.

(1) Setup procedure

To specify settings for distributing requests by the POST data size, you use the following procedure:

1. Define the POST request-distributing worker and POST request-forwarding worker in workers.properties.

   Definitions for POST request-distributing worker

   Specify the list of worker names, worker types (specify `post_size_lb`), and list of workers for distribution by the POST data size. As needed, set the default worker.

   Definitions for each POST request-forwarding worker

   Specify the worker types (specify `ajp13`), port number, host name, and the upper limit of the POST data size.

   The default value is defined in `workers.properties` that is provided by default. To use the default definition defined as a comment, delete the hash mark (#) at the beginning of the applicable line.

   For details on `workers.properties` (worker definition file), see *14.2.4 workers.properties (Worker definition file)*.

2. Define the mapping between the URL pattern and worker in mod_jk.conf.

   If a mapping is already defined, delete the definition or replace the mapping.

   For details on `mod_jk.conf` (redirector operation definition file for HTTP Server), see *14.2.2 mod_jk.conf (Redirector action definition file for Cosminexus HTTP Server)*.

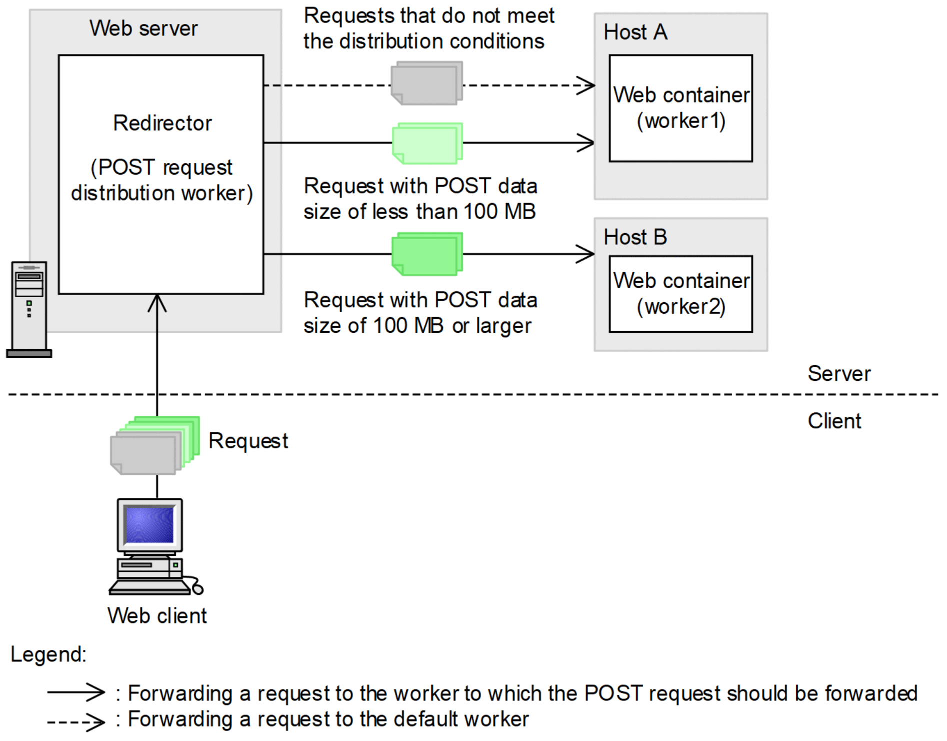
3. Set up the Web server environment and restart the Web server.

   For details on the Web server settings, see , *Appendix D Precautions related to Cosminexus HTTP Server Settings*.

(2) Example settings

The figure below shows the distribution of requests by POST data size. This figure shows an example for the case when request is not limited:

Figure 5–14: Example of distribution of requests by the POST data size


Web server
Redirector
(POST request distribution worker)
Requests that do not meet the distribution conditions
Request with POST data size of less than 100 MB
Request with POST data size of 100 MB or larger
Host A
Web container (worker1)
Host B
Web container (worker2)
Server
Client
Request
Web client
Legend:
→ : Forwarding a request to the worker to which the POST request should be forwarded
- - -> : Forwarding a request to the default worker

In this example, among the requests under `/examples`, the requests with POST data size of less than 100 MB are distributed to host A and the requests with POST data size between 100 MB and 200 MB are distributed to host B. The requests that do not fulfill the request distribution conditions are distributed to the host A that is set as the default worker.

The worker name of the host A is `worker1` and the worker name of the host B is `worker2`. For details on the request distribution conditions, see *5.5.3 Request distribution conditions*.

An example of the `workers.properties` file is shown below. The POST request-distributing worker, POST request-forwarding worker, and default worker are defined here. As the upper limit of POST data size, `100m` is specified for `worker1` and `2048m` is specified for `worker2` (maximum value of the upper limit for POST data size). In the default worker, `worker1` is specified.

Example of workers.properties (In Windows)

```
worker.list=postsizelb1

worker.postsizelb1.post_size_workers=worker1, worker2
worker.postsizelb1.type=post_size_lb
worker.postsizelb1.default_worker=worker1

worker.worker1.port=8007
worker.worker1.host=hostA
worker.worker1.type=ajp13
worker.worker1.cachesize=64
worker.worker1.post_data=100m

worker.worker2.port=8007
worker.worker2.host=hostB
worker.worker2.type=ajp13
worker.worker2.cachesize=64
worker.worker2.post_data=2048m
```

Example of workers.properties (In UNIX)

```
worker.list=postsizelb1

worker.postsizelb1.post_size_workers=worker1, worker2
worker.postsizelb1.type=post_size_lb
worker.postsizelb1.default_worker=worker1

worker.worker1.port=8007
worker.worker1.host=hostA
worker.worker1.type=ajp13
worker.worker1.post_data=100m

worker.worker2.port=8007
worker.worker2.host=hostB
worker.worker2.type=ajp13
worker.worker2.post_data=2048m
```

An example of the `mod_jk.conf` file is shown below. POST request-distributing worker `postsizelb1` will be specified here.

Example of mod_jk.conf

```
JkMount /examples/* postsizelb1
```

Reference note

You can also set the upper limit for the POST data size in the `LimitRequestBody` directive of `httpsd.conf` (Cosminexus HTTP Server definition file). For details on the `LimitRequestBody` directive, see the *uCosminexus Application Server HTTP Server User Guide*.

5.6 Communication timeout (Web server integration)

This section describes communication timeout in Web server integration.

When you use the functionality for Web server integration, you can set the communication timeout for receiving requests and sending responses between the client and the Web server, and also between the Web server and the Web container. When the response is awaited due to the network and application failure, you can detect the occurrence of failure from the occurrence of a timeout, if the communication timeout is set.

The following table describes the organization of this section.

Table 5–12: Organization of this section (Communication timeout (Web server integration))

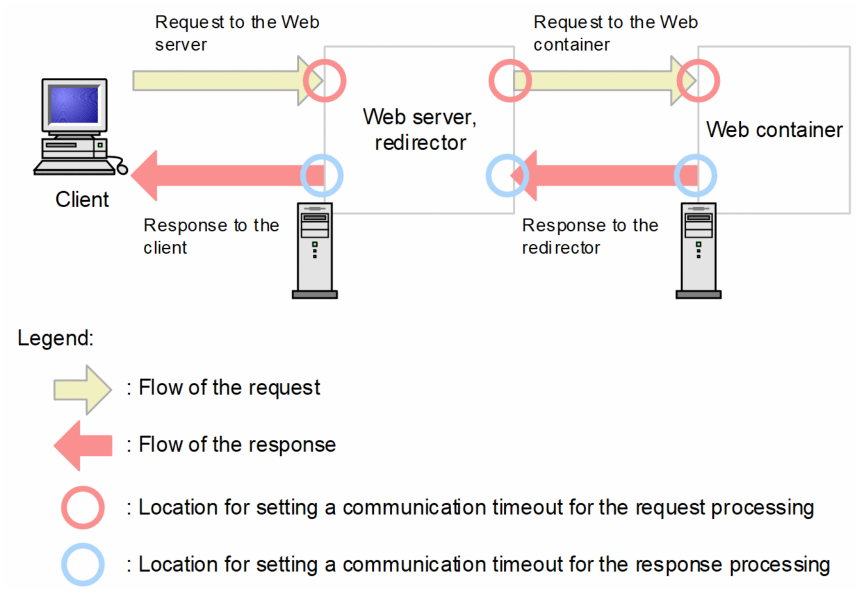
Category	Title	Reference
Description	Communication timeout when sending and receiving a request	*5.6.1*
	Communication timeout when sending and receiving a response	*5.6.2*
Settings	Setting the communication timeout	*5.6.3*
	Setting the communication timeout when sending and receiving a request (When the Smart Composer functionality is used)	*5.6.4*
	Setting the communication timeout when sending and receiving a request (When the Smart Composer functionality is not used)	*5.6.5*
	Setting the communication timeout when sending and receiving a response (When the Smart Composer functionality is used)	*5.6.6*
	Setting the communication timeout when sending and receiving a response (When the Smart Composer functionality is not used)	*5.6.7*

Note:

There is no specific description of *Implementation*, *Operations*, and *Notes* for this functionality.

When you use the Web server integration functionality, set the communication timeout for the communication indicated by the four arrows in the following figure. In the case of communication between the redirector and the Web container, you can set the communication timeout in both the redirector and Web container. In the case of sending requests, you can set the communication timeout when the redirector sends the requests and the Web container receives the requests. Similarly, when sending responses, you can set the communication timeout when the Web container sends the responses and the redirector receives the responses. The following figure shows the communication for which timeout can be set.

Figure 5–15: Communication for which timeout can be set

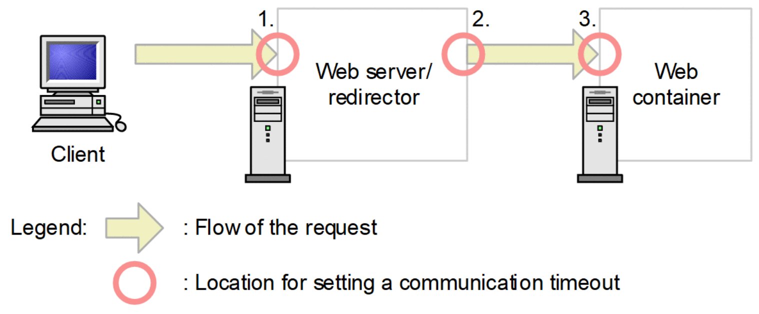

Request to the Web server
Request to the Web container
Web server, redirector
Web container
Client
Response to the client
Response to the redirector
Legend:
: Flow of the request
: Flow of the response
: Location for setting a communication timeout for the request processing
: Location for setting a communication timeout for the response processing

The setting of communication timeout is explained separately for receiving of a request and sending of a response.

5.6.1 Communication timeout when sending and receiving a request

This section explains the setting of communication timeout for sending and receiving requests when the Web server integration functionality is used. The following figure shows the locations to set the communication timeout for sending and receiving requests.

Figure 5–16: Locations to set the communication timeout for sending and receiving requests



When you use the Web server integration functionality, set the communication timeout at the locations marked with a O sign in the figure. The locations to set the communication timeout are explained below. The numbers in the figure correspond to the numbers in the following explanation:

1. When a request is received by the Web server (Client - Web server)

   Set the communication timeout when the Web server receives a request from the client. Set the communication timeout in the Web server.

   For details on the failures that can be detected by setting the communication timeout when the request is received by the Web server, see *(1) When a request is received by the Web server* explained below.

2. When a request is sent by the redirector (Redirector - Web container)

   Set the communication timeout when a request is sent from the redirector to the Web container. Set the communication timeout in the redirector.

For details on the failures that can be detected by setting a communication timeout when the request is sent by the redirector, see *(2) When a request is sent by the redirector* explained below.

3. When a request is received by the Web container (Redirector - Web container)

   Set the communication timeout when the Web container receives a request from the redirector. Set the communication timeout in the Web container.

   For details on the failures that can be detected by setting a communication timeout when the request is received by the Web container, see *(3) When a request is received by the Web container* explained below.

(1) When a request is received by the Web server

You can set the timeout period in the Web server, for receiving a request transferred from the client. You can detect the occurrence of failure in the client by using the communication timeout that was set up. You can detect the following failures:

Detectable failures

- The host on which the client is running is down.
- A network failure occurs between the client and the Web server.
- A failure occurs in client application.

(2) When a request is sent by the redirector

You can set the timeout period in the redirector, for sending a request to the Web container. When the set timeout period is exceeded and a timeout occurs, a message is output to the error log of Cosminexus HTTP Server.

(a) Communication timeout that can be set and detectable failures

You can set the timeout to the following two time-periods when a request is sent by the redirector:

- The time to establish a connection for sending request to the Web container
- The time to send a request to the Web container

You can detect the occurrence of failure in the Web container or on the network using the communication timeout that was set up. You can detect the following failures:

Detectable failures

- The host on which the Web container is running is down.
- A network failure occurs between the redirector and the Web container.

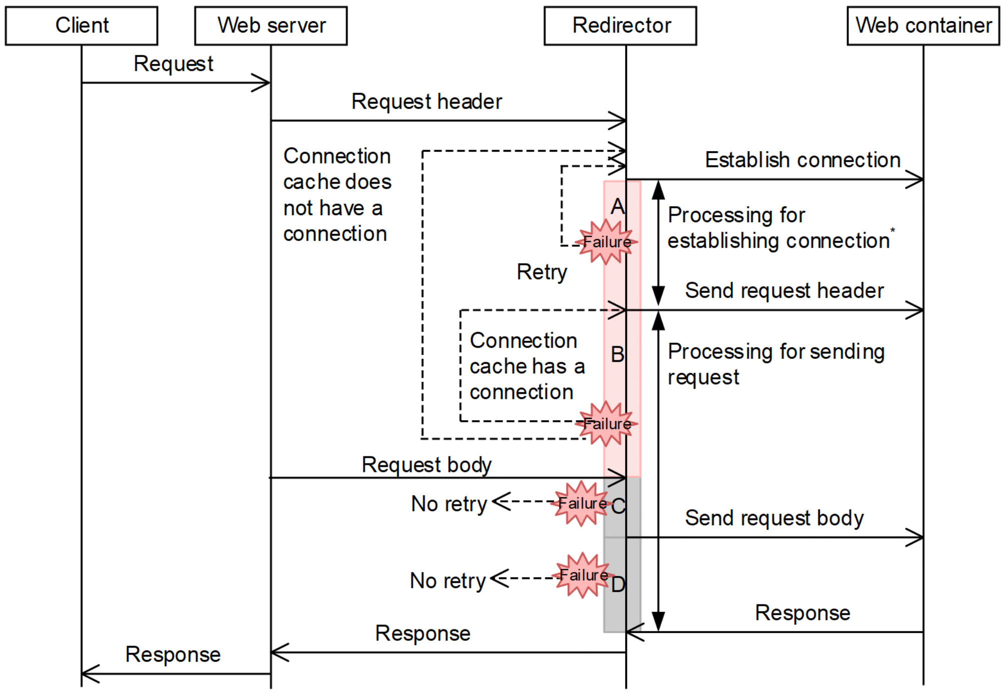
(b) Retrying sending of requests

If a request cannot be sent temporarily from the redirector, you can retry sending the request. A request might not be sent temporarily in the following cases:

- In the case of temporary failure of the network
- When requests are centralized in the Web container during establishment of a connection, and a request for establishing connection overflows temporarily from the listen queue
- When a Web container is yet to start completely

You can retry establishing a connection, and sending request headers. The following figure shows the flow of retry process.

Figure 5–17: Flow of retry process when requests are sent to the Web container



Legend:

🟥: Retry occurs when a timeout occurs

⬜: Retry does not occur when a timeout occurs

A: Establish connection to the Web container
B: Send the request header to the Web container
C: Receive the request body from the Web server
D: Send the request body to the Web container

* The processing for establishing connection is carried out only when the connection cache does not have a connection.

Retry is executed when a timeout occurs in A or B of the figure. Retry is not executed when processing fails at C or D of the figure, and timeout occurs. The connection is closed and an error is returned to the client. Note that the failure of processing at C or D in the figure indicates failure in receiving request body from the Web server, or failure in sending request body to the Web container.

Tip

In the case of C or D in the figure, the processing of the request may have started already in the Web container. If you execute retry in the case of failure in receiving the request body from the Web server, or in the case of failure in sending the request body to the Web container, the send process may be duplicated, and therefore, retry is not done.

The retry operation in the case of failure in processing at A and B in the figure is explained below:

- **Retry operation in the case of failure in processing at A in the figure**

  After sending a request from the redirector to the Web container for establishment of a connection, if the power supply to the host on which the Web container is running is disrupted or a network failure occurs, the operation is performed as follows:

1. If the set timeout period elapses, a message indicating the occurrence of timeout during establishment of the connection is output.
2. Retry establishing connection only for specified number of times.

Note that if a connection cannot be established in spite of retrying for the specified number of times, a message indicating failure in sending request is output, and an error (status code 500) is returned to the client.

- **Retry operation in the case of failure in processing at B part in the figure**

After a connection is established successfully, or the request header is sent to the Web container, if the power supply to the host on which the Web container is running is disrupted and a network failure occurs, perform the operation as follows:

1. If the set timeout period elapses, a message indicating the occurrence of timeout when sending a request is output.
2. Close the connection that was used for sending the request header.
3. Retry sending the request header only for specified number of times.

   Note that at this point, the operation depends upon whether the connections are available in the connection cache.

   - When connections are available in the connection cache

   Use a connection in the connection cache and retry the process from sending the request header.

   - When there are no connections in the connection cache

   Re-establish a connection, and then retry sending the request header.

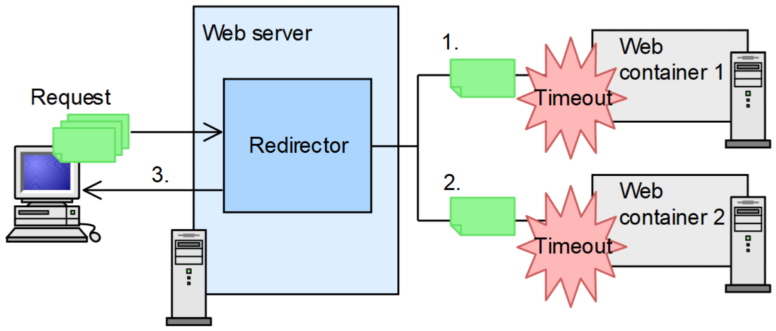
Note that if the request header cannot be sent in spite of retrying for the specified number of times, a message indicating failure in sending request is output, and status code 500 is returned to the client.

- **Retry operation when a request is distributed using a load balancer**

The following figure shows the retry operation when you use a load balancer to distribute a request:

Note that in this figure, a request is first distributed to Web container 1 and then to Web container 2, and the retry frequency is set as three times.

Figure 5–18: Retry operation when a request is distributed using a load balancer



The retry operation shown in the figure is as follows:

1. Sending request to the Web container 1

   A request is sent to Web container 1. Retry operation is performed in the case of failure in establishing a connection to Web container 1, or failure in sending the request header. Retry is executed up to three times.

2. Sending request to the Web container 2

   If the retry operation fails three times in Web container 1, the request is transferred to Web container 2. When the request is transferred, retry is counted again from 1, and hence, retry is executed up to three times even in Web container 2.

3. Error notification to the client

   If the process of establishing a connection or sending the request header fails three times even in Web container 2, an error (status code 500) is returned to the client.

Reference note

A request is transferred only to the set number of Web containers.

(3) When a request is received by the Web container

You can set a timeout period in the Web container, for receiving a request transferred from the redirector. You can detect the occurrence of failure in the redirector using the communication timeout that was set up. You can detect the following failures:

Detectable failures

- The host on which the Web server is running is down.
- A network failure occurs between the Web server and the Web container.
- Timeout occurred in the Web container before the processing between the client and the Web server completes.
  This implies that when the data size requested by the Web container was being read between the client and the Web server, the communication timeout set in the Web container occurred before the data is read completely, due to insufficient communication speed between the client and the Web server.

The following table describes the conditions for occurrence of communication timeout and the operations after the timeout occurs.

Table 5–13: Conditions for occurrence of communication timeout and the operations after occurrence

Conditions for occurrence of communication timeout	Operations after occurrence
In the case all the following conditions are satisfied when a request is received: • The request contains a body data. • The body data is not in the chunk format. • After the reading process starts, a failure occurs in the host on which the redirector is running, or in the network between the redirector and the Web container.	• Request is not processed. • Message KDJE39188-E is output to the message log.
In the case all the following conditions are satisfied when using API# in the servlet (JSP): • The request contains a body data. • The body data is not in the chunk format. • After the reading process starts, a failure occurs in the host on which the redirector is running, or in the network between the redirector and the Web container.	• `java.lang.IllegalStateException` occurs. • The connection to the redirector is closed, and thereafter you cannot read or write the data. • Message KDJE39188-E is output to the message log.
In the case all the following conditions are satisfied when POST data is read by using the `java.io.BufferedReader` class acquired by `getReader` method of `javax.servlet.ServletRequest` class or `javax.servlet.ServletInputStream` class in the servlet (JSP): • The request contains a body data. • After the reading process starts, a failure occurs in the host on which the redirector is running, or in the network between the redirector and the Web container.	• `java.net.SocketTimeoutException` occurs. • The connection to the redirector is closed, and thereafter you cannot read or write the data. • Message KDJE39188-E is output to the message log.

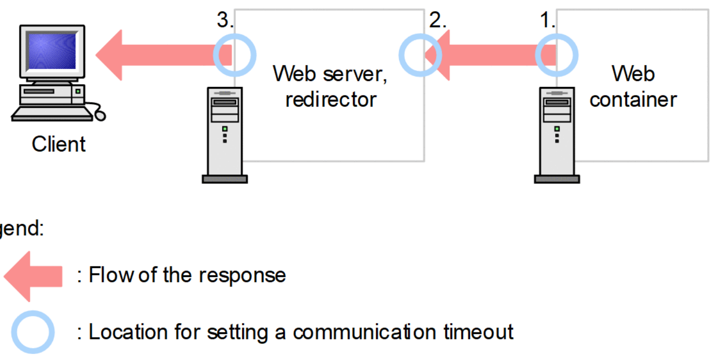
#

Indicates the case of using the `getParameter` method, `getParameterMap` method, `getParameterNames` method, and `getParameterValues` method of `javax.servlet.ServletRequest`.

5.6.2 Setting the communication timeout when sending and receiving a response

This section explains the setting of communication timeout for sending and receiving responses when the Web server integration functionality is used. The following figure shows the locations to set the communication timeout for sending and receiving responses:

Figure 5–19: Locations to set the communication timeout for sending and receiving responses (when the Web server integration functionality is used)



When you use the Web server integration functionality, set the communication timeout at the locations marked with a O sign in the figure. The locations to set the communication timeout are explained below. The numbers in the figure correspond to the numbers in the following explanation:

1. When a response is sent by the Web container (Web container - redirector)

   Set the communication timeout when a response is sent from the Web container to the redirector. Set the communication timeout in the Web container.

   For details on the failures that you can detect by setting the communication timeout when a response is sent by the Web container, see *(1) When a response is sent by the Web container* explained below.

2. When a response is received by the redirector (Web container - redirector)

   Set communication timeout when the redirector receives a response from the Web container. Set the communication timeout in the redirector.

   For details on the failures that you can detect by setting communication timeout when response is received by the redirector, see *(2) When a response is received by the redirector* explained below.

3. When a response is sent by the Web server (Web server - Client)

   Set communication timeout when a response is sent from the Web server to the client. Set the communication timeout in the Web server.

   For details on the failures that you can be detect by setting the communication timeout when a response is sent by the Web server, see *(3) When a response is sent by the Web server* explained below.

(1) When a response is sent by the Web container

You can set a timeout period in the Web container, for sending a response to the redirector. You can detect the occurrence of failure in the redirector using the communication timeout that was set up. You can detect the following failures:

Detectable failures

- The host on which the redirector is running is down.
- A network failure occurs between the Web container and the redirector.

When the communication timeout occurs, KDJE39507-E (timeout occurred when sending a response) is output to the message log. The following table describes the conditions for occurrence of communication timeout and the operations after the timeout occurs.

Table 5–14: Conditions for occurrence of communication timeout and the operations after occurrence

Timing of occurrence of communication timeout	Operation of the method after occurrence of communication timeout	Operation of servlets or JSPs after occurrence of communication timeout
When response data is sent to the client by using the method of `javax.servlet.ServletOutputStream` class acquired by `getOutputStream` method of `javax.servlet.ServletResponse` class, in the servlet	Exception `java.net.SocketTimeoutException` occurs.	Since the connection to the redirector is closed, you cannot send or receive the request data and the response data.
When response data is sent to the client by using the method of `java.io.PrintWriter` class acquired by `getWriter` method of `javax.servlet.ServletResponse` class, in the servlet	The send process is interrupted and returned.	• The `checkError` method of `java.io.PrintWriter` class returns true. • Since the connection to the redirector is closed, you cannot send or receive the request data and the response data.
When response data is sent to the client by using the method of `javax.servlet.jsp.JspWriter` class, in JSP	Exception `java.net.SocketTimeoutException` occurs.	Since the connection to the redirector is closed, you cannot send or receive the request data and the response data.
When the response data of static contents is sent to the client	--	--

Legend:

--: Not applicable

(2) When a response is received by the redirector

When a request is sent to the Web container, the redirector awaits for a response from the Web container. You can set the timeout for this response waiting time. You can detect the occurrence of failure in the Web container using the communication timeout that was set up. You can detect the following failures:

Detectable failures

- The host on which the Web container is running is down.
- A network failure occurs between the Web container and the redirector.
- A Web application failure occurs in the Web container.

The following failures occur in the Web application:

Web application failures

- A response is not returned due to an infinite loop in the servlets or JSPs processing.
- The Enterprise Bean and database are invoked as an extension of servlets or JSPs, and response from them is awaited.
- Dead lock occurs in the Web application.
- The Web application does not catch up with the server processing and is running slow during the peak access.

Operation after communication timeout in the redirector

When communication timeout occurs, the redirector disconnects the connection to the Web container, and returns error with status code 500 to the client.

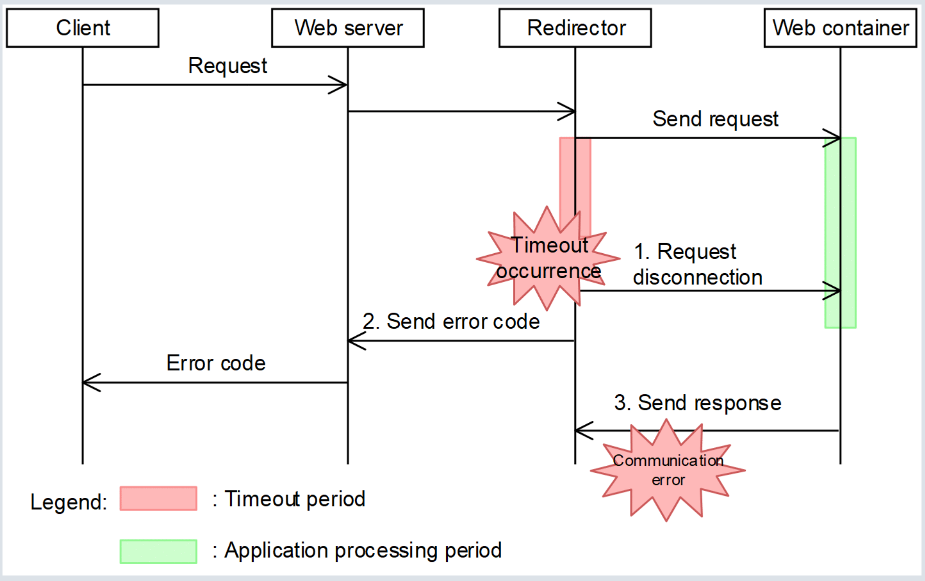
Tip

Operation when timeout occurs during processing of an application

Even if the redirector times out during processing in the Web container, you cannot detect that the redirector has timed out, in the Web container.

You can detect the timeout in the redirector once the processing of the Web container finishes, and a response is transferred to the redirector. In such a case, however, since the redirector has already disconnected the connection to the Web container, an error occurs when sending a response. The following figure illustrates the operation when a timeout occurs during processing of an application.

Figure 5–20: peration when a timeout occurs during processing of an application



The figure is explained below.

1. When a timeout occurs in the redirector, a request is sent for disconnecting the connection to the Web container.
2. Redirector sends an error code to the Web server.
3. When processing of the application in the Web container finishes, a response is sent to the redirector. However, since the connection between the redirector and the Web container is already disconnected in 1, a communication error occurs.

(3) When a response is sent by the Web server

You can set a timeout period in the Web server, for sending data to the client. You can detect the occurrence of failure in the client by using the communication timeout that was set up. You can detect the following failures:

Detectable failures

- The host on which the client is running is down.
- A network failure occurs between the client and the Web server.

- A failure occurs in client application.

5.6.3 Setting the communication timeout

This section describes the communication timeout settings between a client and a Web server and between a Web server (redirector) and a Web container.

You set a communication timeout when sending and receiving requests or when sending and receiving responses. The method of specifying communication timeout differs according to the availability of Smart Composer. The following table lists the communication timeout setting method and the corresponding reference sections:

Table 5–15: Setting method and references of communication timeout (Web server integration)

Set up timing	Usage of Smart Composer	
	Used	Not used
When sending and receiving a request	*5.6.4*	*5.6.5*
When sending and receiving a response	*5.6.6*	*5.6.7*

Note that you can also set communication timeout in the EJB client that invokes EJB. Specify the settings in the EJB client during J2EE application development. For the setting method, see *2.11.5 Timeout of RMI-IIOP communications* in the *uCosminexus Application Server EJB Container Functionality Guide*.

5.6.4 Setting the communication timeout when sending and receiving a request (When the Smart Composer functionality is used)

You set the communication timeout for sending and receiving requests between the client and Web server and the redirector and Web Container.

The following are the methods of setting a communication timeout in each of these cases:

(1) Settings in the Web server for receiving requests

Specify the communication timeout in the Web server, when receiving the requests from the client into the Web server. You can use Cosminexus HTTP Server as the Web server.

(a) How to set

Set the communication timeout for the receiving process of requests forwarded from the client in `httpsd.conf`.

- **Waiting time for the request receiving process**
  In the `Timeout` directive, set the waiting time (seconds) for the process of receiving requests from the client. The default value of the `Timeout` directive is 300 seconds. Note that the value set here is shared with the communication timeout for the process of sending data to the client. For details on the communication timeout for the process of sending data to the client, see *5.6.6(3) Settings in the Web server for sending responses*.

(b) Precautions for setup

To set the timeout for receiving requests in the Web server, take into consideration the network configuration and traffic status between the client and the Web server and specify a time in which the occurrence of failure can be determined.

(2) Settings in the redirector for sending requests

When sending a request from the redirector to the Web container, first establish a connection with the Web container. You can set the communication timeout for sending requests from the redirector to the Web container when the connection is established and when the request is sent. You can also set the retry frequency to be used when an attempt to establish connection and to send the request header fails.

(a) How to set

Specify the communication timeout for establishing the connection and the request sending process and the retry frequency from the redirector in the Easy Setup definition file.

- **Communication timeout in establishing connection**
  Set the waiting time (seconds) for the process of establishing a connection with the Web container in the `JkConnectTimeout` parameter in the `<configuration>` tag of the logical Web server (web-server). The default value of the `JkConnectTimeout` parameter is 30 seconds.
- **Timeout for the request sending process**
  Set the waiting time (seconds) for the process of sending a request to the Web container in the `JkSendTimeout` parameter in the `<configuration>` tag of the logical Web server (web-server). The default value of the `JkSendTimeout` parameter is 100 seconds.
- **Retry frequency for establishing a connection and sending a request**
  Set the retry frequency for establishing a connection and sending a request to the Web container in the `JkRequestRetryCount` parameter in the `<configuration>` tag of the logical Web server (web-server). The default value of the `JkRequestRetryCount` parameter is three times.

(b) Precautions for setup

Take the followings into consideration when you specify the communication timeout value for sending requests and the retry frequency set up in the redirector:

- The retry frequency includes the first connection established and the first request sent. Therefore, if the first connection or request sending process fails, the retrying of connection or request sending process is counted as the second time.
- If you specify `0` as the communication timeout for establishing a connection and for the request sending process or if you set a longer time than the re-send timer for establishing a connection and sending data by TCP, the TCP timeout value is applied to communication timeout value.

(3) Settings in the Web Container for receiving requests

Set the communication timeout when the Web container receives a request from the redirector, to the Web container.

(a) How to set

Set the communication timeout for the process of receiving requests forwarded from the redirector in the Easy Setup definition file:

- **Timeout for the request receiving process**
  Set the waiting time (seconds) for the process of receiving requests from the redirector in `webserver.connector.ajp13.receive_timeout parameter` in the `<configuration>` tag of the logical J2EE server (j2ee-server). The default value of the parameter is 100 seconds.

(b) Precautions for setup

Take the followings into the consideration when you specify the timeout value set in the Web container for receiving requests:

- Set a value bigger than the time set in the timeout for receiving Web server requests.
  If a value smaller than the time specified as the timeout for receiving Web server requests is set, when network failure occurs in the client and between the client and Web server, timeout occurs in the Web container before the Web server. In this case, one cannot determine whether the failure has occurred in the Web server or in the client.
- If data must be received from the client, set the time in which data can be received taking into consideration the communication speed with the client.
- When failure occurs in the redirector while the data is being sent to the Web container, the failure is detected by the timeout in the TCP re-send timer.
  Note that in UNIX, the time until timeout depends on the OS.

5.6.5 Setting the communication timeout when sending and receiving a request (When the Smart Composer functionality is not used)

You set the communication timeout for sending and receiving requests between the client and the Web server and between the redirector and the Web container.

The following are the methods for setting the communication timeout in each of the cases:

(1) Settings in the Web server for receiving requests

In the Web server, you specify the communication timeout for receiving the requests from the client in the Web server. You can use either Cosminexus HTTP Server or Microsoft IIS as the Web server.

(a) How to set

Set the communication timeout for the process of receiving requests forwarded from the client in the following files:

- `httpsd.conf` (In Cosminexus HTTP Server)
  In the `Timeout` directive, set the waiting time (seconds) for the process of receiving requests from the client. The default value of the `Timeout` directive is 300 seconds. Note that the value set here is shared with the communication timeout for the process of sending data to the client. For details on the communication timeout for the process of sending data to the client, see *5.6.6(3) Settings in the Web server for sending responses*.
- `isapi_redirect.conf` (In Microsoft IIS)
  Set the waiting time (seconds) for the process of receiving requests from the client in the `receive_client_timeout` key.

(b) Precautions for setup

To set the timeout for receiving requests in the Web server, take into consideration the network configuration and traffic status between the client and the Web server and specify a time in which the occurrence of failure can be determined.

(2) Settings in the redirector for sending requests

When sending a request from the redirector to the Web container, first establish a connection with the Web container. You can set the communication timeout for sending requests from the redirector to the Web container when the connection is established and when the request is sent. You can also set the retry frequency to be used when an attempt to establish connection and to send the request header fails.

(a) How to set

Specify the communication timeout for establishing the connection and the request sending process and the retry frequency from the redirector in the following files:

- `mod_jk.conf` (In Cosminexus HTTP Server)
  - Communication timeout in establishing connection
    Set the waiting time (seconds) for the process of establishing a connection with the Web Container in the `JkConnectTimeout` key. The default value of the `JkConnectTimeout` key is 30 seconds.
  - Timeout for the request sending process
    Set the waiting time (seconds) for the process of sending a request to the Web container in the `JkSendTimeout` key. The default value of the `JkSendTimeout` key is 100 seconds.
  - Retry frequency for establishing a connection and sending a request
    Set the retry frequency for establishing a connection and sending a request to the Web container in the `JkRequestRetryCount` key. The default value of the `JkRequestRetryCount` key is three times.
- `isapi_redirect.conf` (In Microsoft IIS)
  - Communication timeout in connection process
    Set the waiting time (seconds) for the process of establishing a connection with the Web container in the `connect_timeout` key. The default value of the `connect_timeout` key is 30 seconds.
  - Timeout for the request sending process
    Set the waiting time (seconds) for the process of sending a request to the Web container in the `send_timeout` key. The default value of the `send_timeout` key is 100 seconds.
  - Retry frequency for establishing a connection and sending a request
    Set the retry frequency for establishing a connection and sending a request to the Web container in the `request_retry_count` key. The default value of the `request_retry_count` key is three times.

(b) Precautions for setup

Take the followings into the consideration when you specify the communication timeout value for sending requests and the retry frequency set in the redirector:

- The retry frequency includes the first connection established and the first request sent. Therefore, if the first connection or request sending process fails, the retrying of connection or request sending process is counted as the second time.
- If you specify `0` as the communication timeout for establishing a connection and for the request sending process or if you set a longer time than the re-send timer for establishing a connection and sending data by TCP, the TCP timeout value is applied to communication timeout value.

(3) Settings in the Web container for receiving requests

Set the communication timeout when the Web container receives a request from the redirector, to the Web container.

(a) How to set

Set the communication timeout for the process of receiving requests forwarded from the redirector in the following file:

- `usrconf.properties`
  Set the waiting time (seconds) for the process of receiving requests from the redirector in the `webserver.connector.ajp13.receive_timeout` key.

(b) Precautions for setup

Take the followings into consideration when you specify the timeout value set in the Web container for receiving requests:

- Set a value bigger than the time set in the timeout for receiving Web server requests.
  If a value smaller than the time specified as the timeout for receiving Web server requests is set, and network failure occurs in the client and between the client and Web server, the timeout occurs in the Web container before the Web server. In this case, you cannot determine whether the failure has occurred in the Web server or in the client.
- If data needs to be received from the client, set the time in which data can be received taking into consideration the communication speed with the client.
- When failure occurs in the redirector while the data is being sent to the Web Container, the failure is detected by the timeout in the TCP re-send timer.
  Note that in UNIX, the time until timeout depends on the OS.

5.6.6 Setting the communication timeout when sending and receiving a response (When the Smart Composer functionality is used)

You set the communication timeout for sending and receiving responses between the Web container and the redirector and between the Web server and the client.

The followings are the methods for setting the communication timeout in each of the cases:

(1) Settings in the Web container for sending responses

In the Web container, you specify the communication timeout for the process of sending a response from the Web container to the redirector. Set the waiting time for sending a response from the Web container in the Easy Setup definition file.

- **Timeout for the response sending process**
  Set the timeout (seconds) for sending a response from the Web container in the `webserver.connector.ajp13.send_timeout` parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server). The default value of the parameter is 600 seconds.

(2) Settings in the redirector for receiving responses

Set communication timeout when the redirector receives a response from the Web container, to the redirector.

(a) How to set

Set the waiting time for the response from the Web container in the Easy Setup definition file.

- **Timeout for the response receiving process**
  Set the response waiting time (seconds) for each worker in the `worker.`*worker-name*`.receive_timeout` parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server). The default value of the parameter is 3600 seconds. If you want to set communication timeout for J2EE applications, define and map workers for each J2EE application.

(b) Precautions for setup

You can detect Web application failure by the timeout in receiving a response in the redirector. Therefore, depending on the operation state, consider the time required for processing Web applications and set a value from which failure can be detected as the communication timeout value.

Take the followings into the consideration when you set the timeout value for receiving responses in the redirector:

- Set a time longer than the time required for processing the Web applications in communication timeout.
  If the set timeout value is shorter than the Web application processing time, even if the Web application is processing normally, the timeout is determined in the redirector and an error is returned to the client.
- Consider the waiting time based on the controlling of the number of concurrently executing threads at the peak of the Web container.
  Because of the controlling of the number of concurrently executing threads, at the peak of the Web Container, there might be requests awaiting processing. Therefore, if controlling the number of concurrently executing threads is set, you must specify the communication timeout considering the extension of the request processing time. For details on settings for controlling the number of concurrently executed threads for each Web container, see *5.11 Controlling the number of concurrently executing threads in the Web container*. For details on settings for controlling the number of concurrently executed threads for each Web application, see the manual *uCosminexus Application Server Web Container Functionality Guide*. For details on settings for controlling the number of concurrently executed threads for each URL group, see the manual *uCosminexus Application Server Web Container Functionality Guide*.
- When failure occurs in the Web container while the data is being sent to the redirector, the failure is detected by the timeout in the TCP re-send timer.
  Note that in UNIX, the time until timeout depends on the OS.

(3) Settings in the Web server for sending responses

In the Web server, you set the communication timeout for sending a response to the client.

(a) How to set

Set the waiting time for the response from the client in `httpsd.conf`.

- **Communication timeout for the data sending process**
  Set the communication timeout in the `Timeout` directive. Note that the communication timeout specified in the `Timeout` directive is specified both as the communication timeout for the request receiving process and the communication timeout for the response sending process. Therefore, you cannot set a different time in the communication timeout for the request receiving process and the communication timeout for the response sending process.
  For details on the communication timeout for the process of receiving requests from clients, see *5.6.4(1) Settings in the Web server for receiving requests*.

(b) Precautions for setup

To set the timeout for sending responses in the Web server, take into the consideration the network configuration and traffic status between the client and the Web server and specify adequate time for sending and receiving data with the client.

Also, for the timeout value set in the Web server for sending responses, specify a value smaller than the timeout value in the Web container for sending responses. If the timeout for sending responses in the Web server is greater than timeout for sending responses in the Web container, and a failure occurs between the client and Web server, the timeout in the Web container for sending a response to the redirector might occur earlier than the timeout in the Web server for sending response to the client. In this case, one cannot determine whether the failure has occurred between the client and the Web server or between the redirector and Web container.

5.6.7 Setting the communication timeout when sending and receiving a response (When the Smart Composer functionality is not used)

You set the communication timeout for sending and receiving responses between the Web container and the redirector and between the Web server and the client.

The following are the methods for setting a communication timeout in each of the cases:

(1) Settings in the Web container for sending responses

In the Web container, you specify the communication timeout for the process of sending a response from the Web container to the redirector. Set the waiting time for sending a response from the Web container in the following file:

- `usrconf.properties`

  Set the timeout (seconds) for sending a response from the Web container in the `webserver.connector.ajp13.send_timeout` key. The default value is 600 seconds.

(2) Settings in the redirector for receiving responses

Set the communication timeout when the redirector receives a response from the Web container, to the redirector.

(a) How to set

Set the waiting time for the response from the Web container in the following file:

- `workers.properties`

  Set the response waiting time (seconds) for each worker in the `worker.`*worker-name*`.receive_timeout` key. If you want to set a communication timeout for J2EE applications, define and map workers for each J2EE application.

(b) Precautions for setup

You can detect Web application failure by a timeout in receiving a response in the redirector. Therefore, depending on the operation state, consider the time required for processing Web applications and set a value from which failure can be detected as the communication timeout value.

Take the following into the consideration when you set the timeout value for receiving responses in the redirector:

- Set a time longer than the time required for processing the Web applications in the communication timeout.

  If the set timeout value is shorter than the Web application processing time, the timeout is determined in the redirector and an error is returned to the client even if the Web application is processing normally.
- Consider the waiting time based on the controlling of the number of concurrently executing threads at the peak of the Web container.

  Because of the controlling of the number of concurrently executing threads, at the peak of the Web container, there might be requests awaiting processing. Therefore, if server management commands are used to specify the control of the number of concurrently executing threads, you must specify the communication timeout considering the extension of the request processing time. For details on settings for controlling the number of concurrently executed threads for each Web container, see *5.11 Controlling the number of concurrently executing threads in the Web container*. For details on settings for controlling the number of concurrently executed threads for each

Web application, see the manual *uCosminexus Application Server Web Container Functionality Guide*. For details on settings for controlling the number of concurrently executed threads for each URL group, see the manual *uCosminexus Application Server Web Container Functionality Guide*.

- When failure occurs in the Web container while the data is being sent to the redirector, the failure is detected by the timeout in the TCP re-send timer.

  Note that in UNIX, the time until timeout depends on the OS.

(3) Settings in the Web server for sending responses

In the Web server, you can set the communication timeout for sending a response to the client.

(a) How to set

Set the waiting time for the response from the client in the following files:

- `httpsd.conf` (In Cosminexus HTTP Server)

  Set the communication timeout in the `Timeout` directive. Note that the communication timeout specified in the `Timeout` directive is specified both as the communication timeout for the request receiving process and the communication timeout for the response sending process. Therefore, you cannot set the different time for the communication timeout of the request receiving process and for the communication timeout of the response sending process.

  For details on the communication timeout for the process of receiving requests from clients, see *5.6.4(1) Settings in the Web server for receiving requests*.

- `MinFileBytesPerSec` property (In Microsoft IIS)

  Set the communication timeout in the `MinFileBytesPerSec` property. In the `MinFileBytesPerSec` property, specify the throughput (byte/second) of the response data sent from the Web server to the client. The communication timeout occurs when the response data throughput falls below the value set in `MinFileBytesPerSec`. Note that, the default value for a communication timeout is 240 byte/second.

  For details on the settings, see the manual *Microsoft IIS*.

(b) Precautions for setup

To set the timeout for sending responses in the Web server, take into consideration the network configuration and traffic status between the client and the Web server and specify adequate time for sending and receiving data with the client.

Also, as the timeout value set in the Web server for sending responses, specify a value smaller than the timeout value in the Web container for sending responses. If the timeout for sending responses in the Web server is greater than timeout for sending responses in the Web container, when failure occurs between the client and the Web server, the timeout in the Web container for sending a response to the redirector might occur earlier than the timeout in the Web server for sending response to the client. In this case, one cannot determine whether the failure has occurred between the client and the Web server or between the redirector and the Web container.

5.7 Specifying the IP address (Web server integration)

This section describes the control of communication with the Web client by specifying the IP address in Web server integration.

The following table describes the organization of this section.

Table 5–16: Organization of this section (Specifying the IP address (Web server integration))

Category	Title	Reference
Description	Bind address specification functionality	*5.7.1*
Settings	Execution environment settings (J2EE server settings)	*5.7.2*
Notes	Precautions for specifying the IP address in Web server integration	*5.7.3*

Note:
There is no specific description of *Implementation* and *Operations* for this functionality.

5.7.1 Bind address specification functionality

In a Web container, you can explicitly specify the IP address to be used in Web server integration. This functionality is called the *Bind address specification functionality*. By using the bind address specification functionality, you can specify the setting so that only a single specific IP address is used for a host having multiple physical network interfaces or single physical network interface, when executing with a host.

Customize the properties of the J2EE server to set the bind IP address. For details on customizing the J2EE server operation settings, see *5.7.2 Execution environment settings (J2EE server settings)*.

5.7.2 Execution environment settings (J2EE server settings)

To specify the IP address in Web server integration, you must set up the J2EE server.

Implement the J2EE server settings in the Easy Setup definition file. To define the IP address in Web server integration, specify the following parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file:

`webserver.connector.ajp13.bind_host`
Specifies the IP address or host name used when the Web server integration functionality is used.

For details on the Easy Setup definition file and the parameters to be specified, see *4.3 Easy Setup definition file* in the *uCosminexus Application Server Definition Reference Guide*.

5.7.3 Precautions for specifying the IP address in Web server integration

The following are the precautions for specifying the IP address in Web server integration:

- When the host name or the IP address is set, only requests for connecting to the specified IP address can be received. Instead of setting the IP address, a connection to any IP address on that host can be received, by specifying the wild card address. By default, the setting is specified to use the wild card address. When using the wild card address, note the following points:

- If the specified host name cannot be resolved in the hosts file or DNS, start the server by using the wild card address.
- If the specified host name or IP address is a remote host, start the server by using the wild card address.

5.8 Error page customization with the Web server integration functionality

When the client accesses a non-existent resource, and a servlet in which an exception occurred, the Web container returns an error status code. An error page corresponding to the error status code returned from the Web container is displayed in the client. In an application server, instead of the error pages displayed in the client, pages created by the user can be displayed in the client. This is called *error page customization*.

This section describes the customization of the error page with the Web server functionality when the system is integrated with the Web server.

The following table describes the organization of this section.

Table 5–17: Organization of this section (Error page customization (Web server integration))

Category	Title	Reference
Description	Overview of error page customization	*5.8.1*
	Mechanism of error page customization	*5.8.2*
Settings	Execution environment settings (When the Smart Composer functionality is used)	*5.8.3*
	Execution environment settings (When the Smart Composer functionality is not used)	*5.8.4*
Notes	Precautions related to error page customization	*5.8.5*

Note:

There is no specific description of *Implementation* and *Operations* for this functionality.

Important note

You can use the error page customization with the Web server functionality, only when you use the Web server integration functionality. You can use the error page customization functionality only in Cosminexus HTTP Server. You cannot use this functionality in Microsoft IIS.

5.8.1 Overview of error page customization

The methods to customize the error pages include: Customization with the `<error-page>` tag of `web.xml` specified in Servlet specifications and customization with the Web server functionality. However, the error page used when the redirector returns an error such as when the communication between the redirector and Web container fails cannot be customized with the method of using the `<error-page>` tag of `web.xml`. Use the Web server functionality to customize the error page when the redirector returns an error. The following table describes the error locations and the corresponding error page customization methods:

Table 5–18: Error locations and the corresponding error page customization methods

Error location	Customization method	
	Method of using the Web server functionality	Method of using the `<error-page>` tag of `web.xml`
Web container	Y	Y
Redirector	Y	--

Legend:
Y: Can be customized
--: Cannot be customized

For details on the conditions for occurrence of an error in the Web container, and the corresponding error status codes, see *Appendix C Error Status Code*.

5.8.2 Mechanism of error page customization

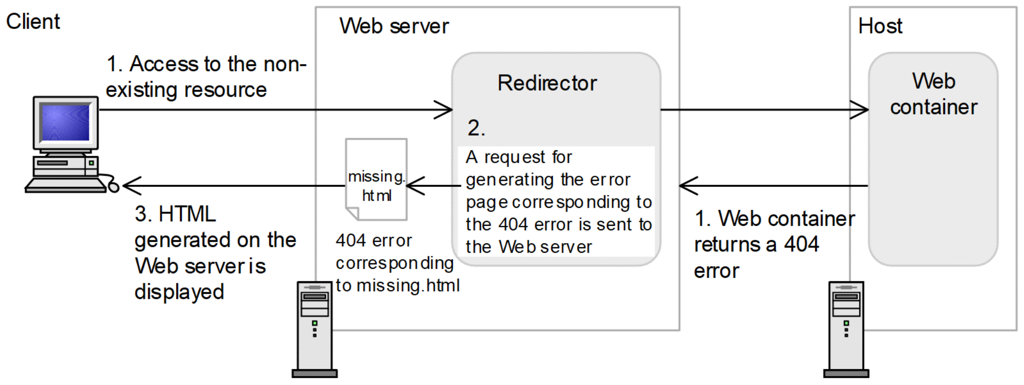
This section describes the mechanism of the process of error page customization when an error occurs in the Web container and when an error occurs in the redirector.

When an error occurs in the Web container

The redirector receives the error status codes sent from the Web container. The redirector assigns creation of error pages to the Web server, and the Web server sends the user created pages corresponding to the error status codes to the client. As a result, the pages created by the user are displayed in the client.

The following figure shows the processing flow of error page customization:

Figure 5–21: Processing of displaying error pages created by the user (when the Web server functionality is used)


Client
Web server
Host
1. Access to the non-existing resource
Redirector
2.
A request for generating the error page corresponding to the 404 error is sent to the Web server
Web container
missing.html
3. HTML generated on the Web server is displayed
404 error corresponding to missing.html
1. Web container returns a 404 error

Stages 1 to 3 of the figure are explained below:

1. If the client accesses a non-existent resource, the Web container sends error 404 to the Web server.
2. When the redirector receives error 404, it requests the Web server to generate an error page corresponding to error 404, based on the setting information#.
3. The Web server returns the error page `missing.html` corresponding to error 404 to the client according to the setting information#.

When an error occurs in the redirector

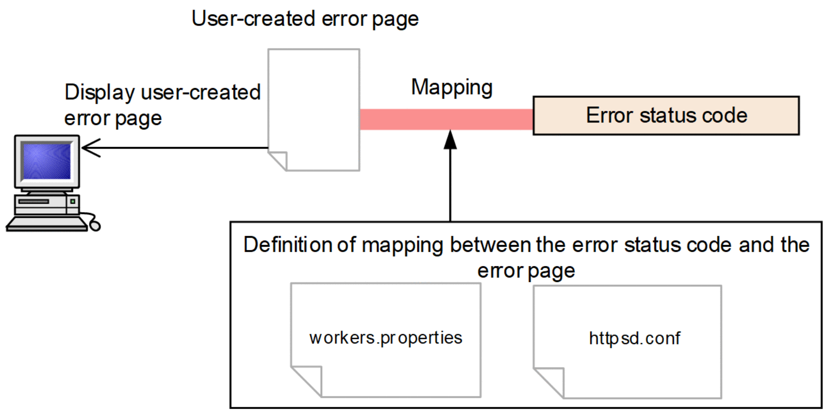
If an error occurs in the redirector, the redirector requests the Web server to generate an error page corresponding to the occurred error, on the basis of the setup information. The Web server sends the user-created page corresponding to the error status code to the client, based on the setup information#. As a result, the pages created by the user are displayed in the client.

#

To customize the error pages, you need to specify the relationship between the error status code and the error page, beforehand.

The following figure shows an overview of the relationship.

Figure 5–22: Specifying the relationship between the error status code and the error page (by using the Web server functionality)


User-created error page
Display user-created error page
Mapping
Error status code
Definition of mapping between the error status code and the error page
workers.properties
httpsd.conf

When an error occurs, in order to display the error page created by the user instead of the error page displaying the error status code, you associate the error page created by the user to a specific error status code. When an error with the applicable error status code occurs, the error page corresponding to the error status code is sent to the client, on the basis of the information set in the Web server (Cosminexus HTTP Server).

5.8.3 Execution environment settings (When the Smart Composer functionality is used)

This section describes the settings for the error page customization.

(1) How to set

Define the association between the error status code and the error page in the following files:

- Easy Setup definition file

  Specify the error status code that you want to associate with the error page in the `worker.`*worker-name*`.delegate_error_code` parameter in the `<configuration>` tag of the logical Web server (web-server).

  For details on the Easy Setup definition file and the parameters, see *4.3 Easy Setup definition file* in the *uCosminexus Application Server Definition Reference Guide*.
- `httpsd.conf`

  Associate the error status code and the file name of the corresponding error page in the `ErrorDocument` directive.

  For details on httpsd.conf (HTTP Server definition file), see the *uCosminexus Application Server HTTP Server User Guide*.

(a) Precautions when specifying the error status codes

Take the following precautions when specifying the error status codes in the Easy Setup definition file:

- Specify the error status code for each worker.
- The worker type that can specify the error status code is only `ajp13`. If the worker type is `lb` (settings specified for load balancing based on the round-robin format) and `post_size_lb` (settings specified for distributing requests based on the POST data size), the specified contents are ignored.
- The specifiable error status code is listed in the following table. The error page cannot be associated with error status codes other than the followings:

Table 5–19: Error status codes that can be associated with error pages

Error status codes	Explanation
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Time-out
409	Conflict
410	Gone
411	Length Required
412	Precondition Failed
413	Request Entity Too Large
414	Request-URI Too Long
415	Unsupported Media Type
416	Requested Range Not Satisfiable
417	Expectation Failed
422	Unprocessible Entity
423	Locked
424	Failed Dependency
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Time-out
505	HTTP Version not supported
507	Insufficient Storage
510	Not Extended

(b) Precautions for specifying the ErrorDocument directive

Take the following precautions when you specify the `ErrorDocument` directive in `httpsd.conf`:

- When using the local URL in the `ErrorDocument` directive, specify a URL that the redirector will not forward to the Web container.

- When the URL pattern `/*` is mapped to a worker in the redirector settings such as for using the root context, all the requests are forwarded to the Web container. Therefore, in the ErrorDocument directive, set the resources on the Web container by using the complete URL.

  The following is the example settings for displaying `error404.jsp` under the root context on the Web container when the root context is used and the error status code 404 occurs. The `hostA` is the host operating the Web server.

  Example:

  ```
  ErrorDocument 404 http://hostA/error404.jsp
  ```

  Also, when the Web container is not running, the redirector returns an error with error status code 500. Therefore, for customizing the error page when the Web container is not running, you must specify other Web server resources for the error status code 500 using the complete URL, in the `ErrorDocument` directive.

(2) Example settings

The following example describes the error page customization:

Example of Easy Setup definition file

```
...
<param>
  <param-name>worker.list</param-name>
  <param-value>worker1</param-value>
</param>
<param>
  <param-name>worker.worker1.type</param-name>
  <param-value>ajp13</param-value>
</param>
<param>
  <param-name>worker.worker1.host</param-name>
  <param-value>host1</param-value>
</param>
<param>
  <param-name>worker.worker1.delegate_error_code</param-name>
  <param-value>404</param-value>
</param>
...
```

Define the error status code '404(Not Found)' in the `worker.`*worker-name*`.delegate_error_code` parameter.

Example of httpsd.conf

```
# Description of httpsd.conf#
#  ...
ErrorDocument 404 /missing.html
```

The error status code and the file name of the corresponding error page are associated. When an error with error status code '404(Not Found)' occurs, the `missing.html` file is displayed.

For details on the `ErrorDocument` directive, see the *uCosminexus Application Server HTTP Server User Guide*.

5.8.4 Execution environment settings (When the Smart Composer functionality is not used)

This section describes the settings for error page customization.

(1) How to set

Define the association between the error status code and error page in the following files:

- `workers.properties`

  Specify the error status code that you want associated with the error page in the `worker.`*worker-name*`.delegate_error_code` key.

  For details on `workers.properties` (worker definition file), see *14.2.4 workers.properties (Worker definition file)*.

- `httpsd.conf`

  Associate the error status code and the file name of the corresponding error page in the `ErrorDocument` directive.

  For details on httpsd.conf (HTTP Server definition file), see the *uCosminexus Application Server HTTP Server User Guide*.

Precautions related to workers.properties settings

- Specify the error status code for each worker.
- The worker type that can specify the error status code is only `ajp13`. If the worker type is `lb` (settings specified for load balancing based on the round-robin format), the specified contents are ignored.
- The specifiable error status code is listed in the following table. The error page cannot be associated with error status codes other than the followings:

Table 5–20: Error status codes that can be associated with error pages

Error status codes	Explanation
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Time-out
409	Conflict
410	Gone
411	Length Required
412	Precondition Failed
413	Request Entity Too Large
414	Request-URI Too Long

Error status codes	Explanation
415	Unsupported Media Type
416	Requested Range Not Satisfiable
417	Expectation Failed
422	Unprocessible Entity
423	Locked
424	Failed Dependency
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Time-out
505	HTTP Version not supported
507	Insufficient Storage
510	Not Extended

Precautions for specifying the ErrorDocument directive

- When using the local URL in the ErrorDocument directive, specify a URL that the redirector will not forward to the Web Container.
- When the URL pattern `/*` is mapped to a worker in the redirector settings such as for using the root context, all the requests are forwarded to the Web container. Therefore, in the `ErrorDocument` directive, set the resources on the Web container by using the complete URL.

  The following example describes the settings for displaying `error404.jsp` under the root context on the Web container when the root context is used and the error status code 404 occurs. The `hostA` is the host operating the Web server.

  `ErrorDocument 404 http://hostA/error404.jsp`

  Also, when the Web container is not running, the redirector returns an error with error status code 500. Therefore, for customizing the error page when the Web container is not running, you must specify other Web server resources for the error status code 500 using the complete URL, in the `ErrorDocument` directive.

(2) Example settings

The following is an example of error page customization:

Example of workers.properties

```
# Description of worker definition file
worker.list=worker1

worker.worker1.type=ajp13
worker.worker1.host=host1
worker.worker1.port=8007
worker.worker1.delegate_error_code=404
```

Define the error status code '404(Not Found)' in the `worker.`*worker-name*`.delegate_error_code` key.

Example of httpsd.conf

```
# Description of httpsd.conf#
#  ...
ErrorDocument 404 /missing.html
```

The error status code and the file name of the corresponding error page are associated. When an error with error status code '404(Not Found)' occurs, the `missing.html` file is displayed.

For details on the `ErrorDocument` directive, see the *uCosminexus Application Server HTTP Serve*r *User Guide*.

5.8.5 Precautions related to error page customization

Note the following points when customizing the error pages with the Web server functionality, in the case of using the Web server integration functionality:

- You can use the error page customization functionality only in Cosminexus HTTP Server. For this reason, when integrating with the Microsoft IIS, even if error page customization is set in `workers.properties`, it becomes invalid.
- When the Web application supports the Servlet 2.3 specifications and you use the error page customization functionality with the `<error-page>` tag of `web.xml` specified in the Servlet specifications, the Web container returns the result of access to the pages described in the `<error-page>` tag, as the status code. Therefore, if an error does not occur in access to the pages described in the `<error-page>` tag , this functionality does not work.
- If the settings for error page customization are specified only in either the worker definition (the `workers.properties` or Easy Setup definition file) or `httpsd.conf`, then even if the specified error occurs in the Web container, the file set by the user is not displayed.

  If the error status code entrusted with the generation of the error page is only specified in the worker definition (the `ErrorDocument` directive of `httpsd.conf` is not defined), the error page returned when the error with that error status code occurs is the page that is automatically generated by Cosminexus HTTP Server.

5.9 Viewing the top page by specifying the domain name

When accessing a deployed Web application merely by specifying the domain name in the URL, the top pages of Web applications, such as `index.html` and `index.jsp` can be displayed. You can use this functionality only when you use the Web server integration functionality. Files such as `index.html` and `index.jsp` are called *welcome files*.

This section describes the viewing of the top page by specifying the domain name.

The following table describes the organization of this section.

Table 5–21: Organization of this section (Viewing the top page by specifying the domain name)

Category	Title	Reference
Description	Viewing the top page by specifying the domain name	*5.9.1*
Settings	Execution environment settings (When the Smart Composer functionality is used)	*5.9.2*
	Execution environment settings (When the Smart Composer functionality is not used)	*5.9.3*

Note:

There is no specific description of *Implementation*, *Operations*, and *Notes* for this functionality.

5.9.1 Viewing the top page by specifying the domain name

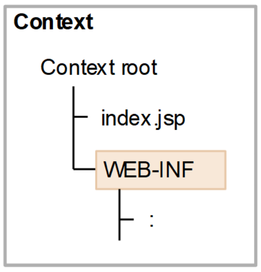
To display the top page only by specifying the domain name, you need to deploy the welcome file in the *root context*. Root context refers to a context whose context root# is a null character (name is not specified in the context root).

#

The unit of management that compile the Web applications is called a *context*. The root path of this context is called a *context root*. When accessing a Web application, specify the context root on the URL.

The following figure explains the context and the context root:

Figure 5–23: Context and context root



The following settings are required to display the top page only by specifying the domain name:

- **Settings of the redirector**
  The root context is accessed via the Web server. Consequently, you need to specify the settings in the URL mapping definition of the redirector, so that the corresponding URL is redirected. Specify the settings in either `mod_jk.conf` (in Cosminexus HTTP Server) or `uriworkermap.properties` (in Microsoft IIS).
- **Settings of the application**
  Specify a null character in the context root of the imported J2EE application.

(1) Notes

Note the following points when using the 'Viewing the top page by specifying the domain name' functionality:

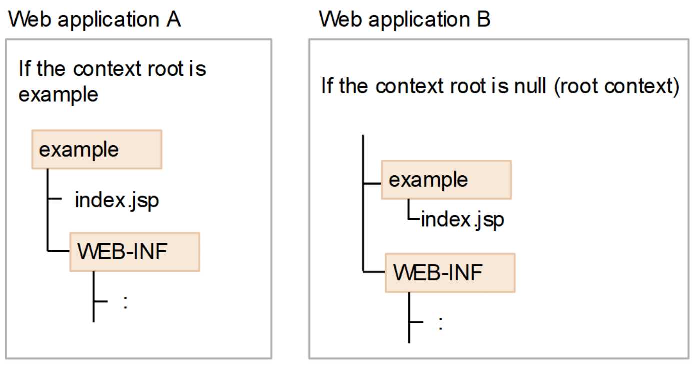
- **Accessed hierarchy when the context root and the root context have the same hierarchies**

  When the context root and the root context have the same hierarchies, the hierarchy of the context root is accessed. An example is shown below.

  Example:

  In this example, the context root of Web application A is `example`, while the context root of Web application B is a null character, and both the Web applications have the hierarchy called `example`.

  Figure 5–24: Example of accessed hierarchy when the context root and root context have the same hierarchies



  In this case, when `http://`*host-name*`/example` is accessed, `example/index.jsp` of Web application A that has a context root is executed.

  If, however, the directory contains `forward` and `include`, and for example `forward` in the directory is accessed, URL is forwarded to the `index.jsp` of the root context.

- **Configuration in the Web application**

  You cannot use `ejb` and `web` at the beginning of the URL.

  Examples of URLs in which you cannot use the `ejb` and `web` at the beginning:

  `http://`*host-name*`:`*port-number*`/ejb/`

  `http://`*host-name*`:`*port-number*`/web/`

  For this reason, do not configure a Web application to be deployed as the root context, so that `ejb` or `web` is at the beginning.

- **How to display in a message text**

  In the messages output to the console and log files, the context root is displayed as a null character.

5.9.2 Execution environment settings (When the Smart Composer functionality is used)

This section describes the settings for viewing the top page by specifying the domain name.

When accessing a deployed Web application merely by specifying the domain name in the URL, the top pages of Web applications, such as `index.html` and `index.jsp` can be displayed.

(1) How to set

To view the top page by specifying the domain name:

1. Specify the root context.

   The root context is a context in which the name is not specified for the context root. The specification of the root context differs according to the operation mode.

   For defining the context root for the J2EE application, see *9.11.1 Defining the context root of a J2EE application* in the *uCosminexus Application Server Application Setup Guide*.

2. Specify distribution of requests to the root context in the redirector.

   Specify the distribution of requests to the root context in the Easy Setup definition file.

   For details about the Easy Setup definition file and parameters, see *4.3 Easy Setup definition file* in the *uCosminexus Application Server Definition Reference Guide*.

(2) Example settings

The following is an example of settings for distributing requests to the root context.

To view the top page of a Web application by specifying only the domain name in the URL, specify settings in the URL mapping definition of the redirector in such a way so that the requests are distributed to the root context. For example, to distribute root context to `worker1` and `/examples` to `worker2`, specify as follows:

Example of Easy Setup definition file

```
...
<param>
  <param-name>JkMount</param-name>
  <param-value>/* worker1</param-value>
  <param-value>/examples/* worker2</param-value>
</param>
...
```

5.9.3 Execution environment settings (When the Smart Composer functionality is not used)

This section describes the settings for viewing the top page by specifying the domain name.

When accessing a deployed Web application merely by specifying the domain name in the URL, the top pages of Web applications, such as `index.html` and `index.jsp` can be displayed.

(1) How to set

To view the top page by specifying the domain name:

1. Specify the root context.

   The root context is a context in which the name is not specified for the context root. The specification of the root context differs according to the operation mode.

   Use the server management commands to specify the root context when you define the J2EE application properties. To set the root context as the context root, specify a null character. For defining the context root for the J2EE application, see *9.11.1 Defining the context root of a J2EE application* in the *uCosminexus Application Server Application Setup Guide*.

2. Specify distribution of requests to the root context in the redirector.

   Specify the distribution of requests to the root context in `mod_jk.conf` when using Cosminexus HTTP Server as the Web server and in `uriworkermap.properties` when using Microsoft IIS as the Web server.

   For details on `mod_jk.conf` (redirector operation definition file for HTTP Server), see *14.2.2 mod_jk.conf (Redirector action definition file for Cosminexus HTTP Server)*.

   For details on `uriworkermap.properties` (mapping definition file for Microsoft IIS), see *14.2.3 uriworkermap.properties (Mapping definition file for Microsoft IIS)*.

(2) Example settings

An example of settings for distributing requests to the root context is as follows.

To view the top page of a Web application by specifying only the domain name in the URL, specify settings in the URL mapping definition of the redirector in such a way so that the requests are distributed to the root context. For example, to distribute root context to `worker1` and `/examples` to `worker2`, specify as follows:

Example of mod_jk.conf (in Cosminexus HTTP Server)

```
JkMount /* worker1
JkMount /examples/* worker2
```

Example of uriworkermap.properties (In Microsoft IIS)

```
/*=worker1
/examples/*=worker2
```

5.10 Notification of gateway information to a Web container

This section describes the reporting of the gateway information to a Web container.

This functionality notifies a Web container of gateway information so that the Web container can properly redirect to a welcome file or Form authentication window.

The following table describes the organization of this section.

Table 5–22: Organization of this section (Reporting the gateway information to a Web Container)

Category	Title	Reference
Description	Gateway specification functionality	*5.10.1*
Settings	Execution environment settings (When the Smart Composer functionality is used)	*5.10.2*
	Execution environment settings (When the Smart Composer functionality is not used)	*5.10.3*
Notes	Precautions related to reporting the gateway information to a Web Container	*5.10.4*

Note:

There is no specific description of *Implementation* and *Operations* for this functionality.

5.10.1 Gateway specification functionality

If a gateway such as an SSL accelerator or a load balancer is placed between a client and a Web server, when the Web container automatically redirects to a welcome file or the Form authentication window, the Web container may not properly create a forwarding URL because the container cannot acquire the information about the gateway.

To avoid this problem, you can use the *gateway specification functionality*. This functionality notifies a Web container of gateway information so that the Web container can properly redirect to a welcome file or Form authentication window.

The gateway specification functionality is used in the following case:

- **When an SSL accelerator is placed between a client and Web server:**
  Even if a client accesses an SSL accelerator via HTTPS, the SSL accelerator accesses a Web server via HTTP, which causes the Web container to assume that the access uses HTTP. For this reason, HTTP is used for the URL scheme for the welcome file or Form authentication window that is the redirection destination.
  In this situation, by using the gateway specification function to specify that the scheme be always considered as HTTPS, you can ensure that accesses are properly redirected.
- **When a request without a Host header needs to be redirected away from the Web server that received the request:**
  When redirecting a request without a Host header, the host name and the port number of the redirection destination URL will be the host name and the port number of the Web server that receives the request.
  Use the gateway specification functionality when the host name and port number of the URL accessed by the client is different from the Web server that receives the request, such as when a load balancer is deployed before the Web server. As a result, the host name and port number accessed from the client are specified, so the request can be redirected properly.

Note that in the case of Web server integration, when accessing a redirector via multiple different routes (such as when HTTP requests are transferred to the Web container from multiple gateways), the gateway specification functionality cannot be used. To use the gateway specification functionality in the case of Web server integration, use a configuration in which there is one access route to the Web Container.

5.10.2 Execution environment settings (When the Smart Composer functionality is used)

This section describes the settings to use the gateway specification functionality.

When a gateway such as an SSL accelerator or load balancer is placed between a client and a Web server, you can use the gateway specification functionality to report the gateway information to the Web container and can properly redirect the access to the top page of the Web application or Form authentication window.

(1) How to set

To use the gateway specification functionality:

1. Specify the gateway host name, port number, and URL scheme for redirect destination for each redirector.
2. Restart the Web server.

Specify the gateway host name, port number, and URL scheme for redirect destination in the Easy Setup definition file. Specify the following parameters in the `<configuration>` tag of the logical Web server (web-server):

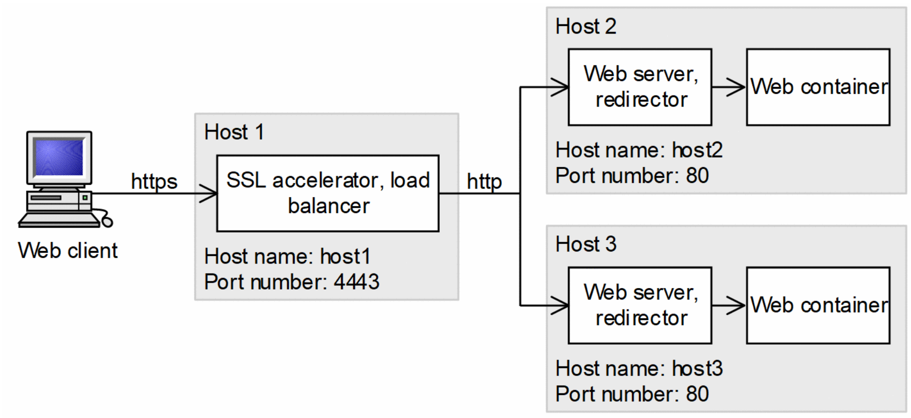
- Host name: `JkGatewayHost`
- Port number: `JkGatewayPort`
- URL scheme for redirect destination: `JkGatewayHttpsScheme`

For details about the Easy Setup definition file and the parameters, see *4.3 Easy Setup definition file* in the *uCosminexus Application Server Definition Reference Guide*.

(2) Example settings

The following figure shows the example settings for the gateway specification functionality:

Figure 5–25: Example settings for the gateway specification functionality



In this example, an SSL accelerator is placed between the client and Web server. Even if a client accesses an SSL accelerator via HTTPS, the SSL accelerator accesses a Web server via HTTP, which causes the Web container to assume that the access uses HTTP. For this reason, HTTP is used for the URL scheme for the top page of the Web application or Form authentication window that is the redirection destination. In this situation, by using the gateway specification function to specify that the scheme be always considered as HTTPS, you can ensure that accesses are properly redirected.

An example of the Easy Setup definition file is described below. Specify `On` in the `JkGatewayHttpsScheme` parameter so that the URL scheme for redirect destination is always considered to be HTTPS.

Example of Easy Setup definition file

```
...
<param>
  <param-name>JkGatewayHost</param-name>
  <param-value>host1</param-value>
</param>
<param>
  <param-name>JkGatewayPort</param-name>
  <param-value>4443</param-value>
</param>
<param>
  <param-name>JkGatewayHttpsScheme</param-name>
  <param-value>On</param-value>
</param>
...
```

5.10.3 Execution environment settings (When the Smart Composer functionality is not used)

This section describes the settings to use the gateway specification functionality.

When a gateway such as an SSL accelerator or load balancer is placed between a client and a Web server, you can use the gateway specification functionality to report the gateway information to the Web Container and can properly redirect the access to the top page of the Web application or Form authentication window.

(1) How to set

To use the gateway specification functionality:

1. Specify the gateway host name, port number, and URL scheme for redirect destination for each redirector.
2. Restart the Web server.

Specify the gateway host name, port number, and URL scheme for redirect destination in `mod_jk.conf` when using Cosminexus HTTP Server as the Web server and in `isapi_redirect.conf` when using Microsoft IIS as the Web server. The keys specified are as follows:

For details on `mod_jk.conf` (redirector operation definition file for HTTP Server), see *14.2.2 mod_jk.conf (Redirector action definition file for Cosminexus HTTP Server)*.

For details on `isapi_redirect.conf` (redirector operation definition file for Microsoft IIS), see *14.2.1 isapi_redirect.conf (Redirector action definition file for Microsoft IIS)*.

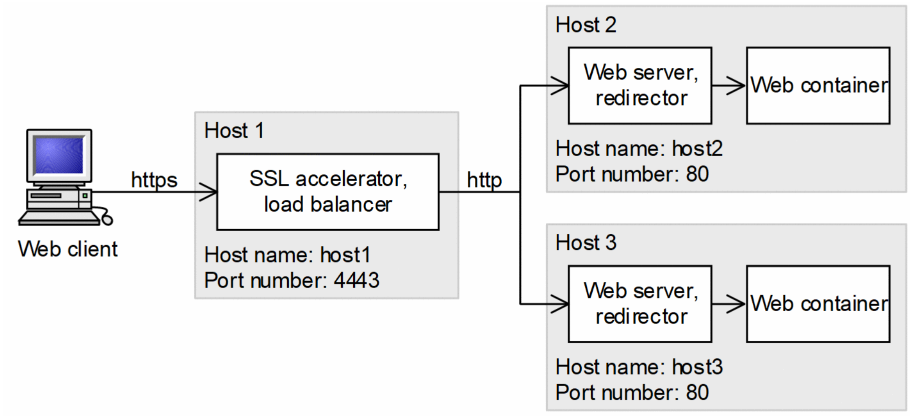
- In `mod_jk.conf`
  Host name: `JkGatewayHost` key
  Port number: `JkGatewayPort` key
  URL scheme for redirect destination: `JkGatewayHttpsScheme` key
- In `isapi_redirect.conf`
  Host name: `gateway_host` key
  Port number: `gateway_port` key

URL scheme for redirect destination: `gateway_https_scheme` key

(2) Example settings

The following figure shows the example settings for the gateway specification functionality:

Figure 5–26: Example settings for the gateway specification functionality



In this example, an SSL accelerator is placed between the client and Web server. Even if a client accesses an SSL accelerator via HTTPS, the SSL accelerator accesses a Web server via HTTP, which causes the Web container to assume that the access uses HTTP. For this reason, HTTP is used for the URL scheme for the top page of the Web application or Form authentication window that is the redirection destination. In this situation, by using the gateway specification function to specify that the scheme be always considered as HTTPS, you can ensure that accesses are properly redirected.

Examples of the `mod_jk.conf` and `isapi_redirect.conf` files are shown below. Specify `On` in the `JkGatewayHttpsScheme` key of `mod_jk.conf` and `true` in the `gateway_https_scheme` key of `isapi_redirect.conf` so that the URL scheme for redirect destination is always considered to be HTTPS.

Example of mod_jk.conf (in Cosminexus HTTP Server)

```
JkGatewayHost host1
JkGatewayPort 4443
JkGatewayHttpsScheme On
```

Example of isapi_redirect.conf (In Microsoft IIS)

```
gateway_host=host1
gateway_port=4443
gateway_https_scheme=true
```

5.10.4 Precautions related to reporting the gateway information to a Web Container

The following are cautionary notes on using the gateway specification functionality:

Specifying the host name and port number of an URL where an access is redirected:

A browser usually sends a request with the Host header appended, so it is not necessary to specify the host name or port number for an URL where access is to be redirected.

Note that you can check whether or not the request has the Host header by calling the `getHeader` method of the `javax.servlet.http.HttpServletRequest` class, with the Host argument specified.

Servlet API behavior:

Using the gateway specification functionality causes some servlet API functions to behave differently. Take care when using API functions with a Web application.

When you use the gateway specification functionality, the behavior of some servlet API functions changes. The following describes the precautions on servlet APIs when using the gateway specification functionality for each method to be used:

- The `sendRedirect` method of the `javax.servlet.http.HttpServletResponse` class

  When you specify a relative URL for the argument, and if the request does not have the Host header, the host name and port number of the URL of the redirection destination are the values specified by the gateway specification functionality. When you specify a relative URL for the argument and use the gateway specification functionality to specify that a scheme is to be considered as `https`, the scheme of the URL of the redirection destination is always `https`.

- The `getRequestURL` method of the `javax.servlet.ServletRequest` interface

  When you use the gateway specification functionality to specify that a scheme is to be considered as `https`, the return value is always a URL starting with `https://`.

- The `getServerName` method of the `javax.servlet.ServletRequest` interface

  When you use the gateway specification functionality to specify the host name of the URL of the redirection destination, and if the request does not have the Host header, the return value is the value you specified.

- The `getServerPort` method of the `javax.servlet.ServletRequest` interface

  When you use the gateway specification functionality to specify the port number of the URL of the redirection destination, and if the request does not have the Host header, the return value is the value you specified. When you use the gateway specification functionality to specify the host name of the URL of the redirection destination, and if the port number is omitted, the return value is `80` when the request scheme is `http`, and `443` when the request scheme is `https`.

- The `getScheme` method of the `javax.servlet.ServletRequest` interface

  When you use the gateway specification functionality to specify that the scheme is to be considered as `https`, the return value is always `https`.

- The `isSecure` method of the `javax.servlet.ServletRequest` interface

  When you use the gateway specification functionality to specify that the scheme is to be considered as `https`, the return value is always `true`.

- The `getAttribute` method of the `javax.servlet.ServletRequest` interface

  The following attributes cannot be obtained even if you used the gateway specification functionality to specify that the scheme is to be considered as `https`:

  - `javax.servlet.request.cipher_suite` (When Microsoft IIS is used for the Web server, this attribute cannot be obtained regardless of whether the gateway specification functionality is used.)

  - `javax.servlet.request.key_size`

  - `javax.servlet.request.X509Certificate`

The `<transport-guarantee>` tag in `web.xml`:

When you use the gateway specification functionality to specify that a scheme is to be considered as HTTPS, a request to a Web server will be considered to use HTTPS even if the request actually uses HTTP. Note that this prevents an access from being redirected to an URL that uses HTTPS, even if you specify `INTEGRAL` or `CONFIDENTIAL` in the `<transport-guarantee>` tag in `web.xml`.

The Secure attribute for cookies:

When you use the gateway specification functionality to specify that a scheme is to be considered as HTTPS, when a session ID generated by a Web container is returned to the client by the session cookie, the `Secure` attribute is appended to the cookie.

Communicating with the Web server without passing the gateway:

When you enable the gateway specification functionality in the redirector, you cannot perform direct HTTP communication without unless passing through the gateway, such as the SSL accelerator and load balancer, in the Web server.

5.11 Controlling the number of concurrently executing threads in the Web container

This section describes the settings for controlling the number of concurrently executing threads in the Web Container.

The following table describes the organization of this section.

Table 5–23: Organization of this section (Controlling the number of concurrently executing threads in the Web Container)

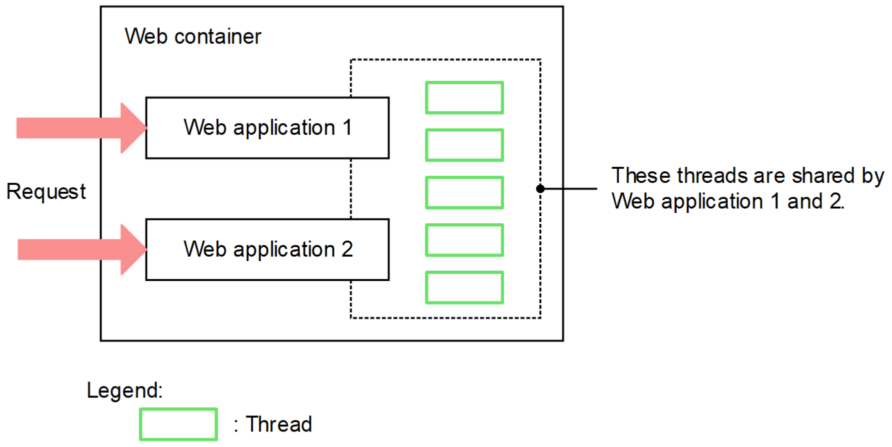
Category	Title	Reference
Description	Mechanism for controlling the number of concurrently executing threads (Web container)	*5.11.1*
Settings	Execution environment settings (J2EE server settings)	*5.11.2*

Note:
There is no specific description of *Implementation*, *Operations*, and *Notes* for this functionality.

5.11.1 Mechanism for controlling the number of concurrently executing threads (Web container)

The following figure shows the mechanism of controlling the number of concurrently executing threads in the Web container:

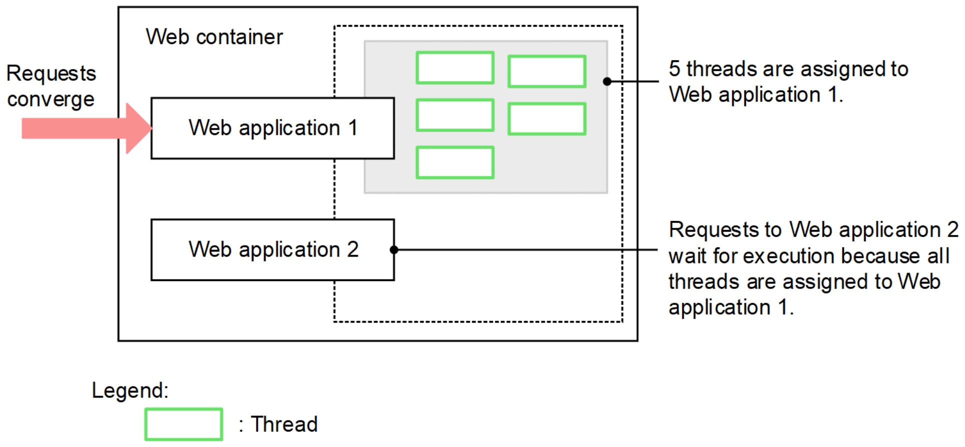
Figure 5–27: Controlling the number of concurrently executing threads in the Web container



For example, if two Web applications are deployed on the Web container, and 5 is set as the number of concurrently executing threads, the number of threads that you can concurrently execute in two Web applications will be 5.

Even when the access is centralized in one of the multiple Web applications deployed on the Web container, by setting the number of concurrently executing threads in the Web container, you can assign the threads to the Web application in which the access is centralized. The following figure shows this mechanism:

Figure 5–28: Handling of threads when the access is centralized (For Web containers)


Web container
Requests converge
Web application 1
Web application 2
5 threads are assigned to Web application 1.
Requests to Web application 2 wait for execution because all threads are assigned to Web application 1.
Legend:
: Thread

As shown in the figure, when two Web applications are deployed on the Web container and `5` is set as the number of concurrently executing threads, all the five threads are assigned to the Web application 1 if the requests are concentrated in Web application 1.

On the other hand, the requests for the Web application 2 are accumulated in the pending queue of the Web container until the request processing of the Web application 1 is complete. Note that the requests accumulated in the pending queue of the Web container are executed in a sequence, after the request processing is complete.

5.11.2 Execution environment settings (J2EE server settings)

Implement the J2EE server settings in the Easy Setup definition file. To define the settings for controlling the number of concurrently executing threads in the Web container, specify one of the following parameters in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file:

- `webserver.connector.ajp13.max_threads`
  Set the maximum number of concurrently executing threads in the entire Web container. Specify this parameter in the case of Web server integration.
- `webserver.connector.inprocess_http.max_execute_threads`
  Set the maximum number of concurrently executing threads in the entire Web container. Specify this parameter when using the in-process HTTP server.

For details on the Easy Setup definition file and the parameters to be specified, see *4.3 Easy Setup definition file* in the *uCosminexus Application Server Definition Reference Guide*.

5.12 Objects for communication with redirector

The Web container secures and releases the Explicit memory block area. The objects for communication used for communication between a Web container and a redirector are usually reused as a permanent connection, and retained when the Web server is being started.

If the connection is disconnected or reconnected due to the occurrence of failure between the Web container and the redirector, the objects for communication are destroyed and regenerated. At that time, the destroyed objects for communication remain in the Tenured area.

To prevent this, on a J2EE server, place the objects for communication between a Web container and a redirector in the Explicit heap and thus prevent unnecessary objects from remaining in the Tenured area and inhibit a Full GC.

This section describes the flow of placing Explicit memory blocks corresponding to the timing of establishing and disconnecting communication.

When communication is established

When connection is established, one Explicit memory block is created for one connection. The objects for communication with the redirector are placed in the created Explicit memory block.

When communication is disconnected

When communication is disconnected, release of one Explicit memory block is reserved for each object, for communication with the redirector placed in the Explicit memory block.

The release is reserved immediately after communication is disconnected. The Explicit memory blocks reserved for release are actually released when the copy GC or Full GC is executed after that. At that time, all the areas reserved for release, are released.

5.13 Explicit heap tuning

This section describes about the Explicit heap tuning.

5.13.1 How to estimate the memory size of Explicit heap (Estimating memory size used in J2EE server)

The settings to use the Explicit Memory Management functionality are required as a prerequisite for tuning Explicit heap. The Explicit Memory Management functionality is enabled when `-XX:+HitachiUseExplicitMemory` is specified as the start option of JavaVM. In J2EE servers, there are default settings to use the Explicit Memory Management functionality. Also, the objects responsible for increasing the memory size of the Tenured area are set to be allocated in Explicit heap. Therefore, always estimate the memory size of Explicit heap required for the objects allocated by J2EE servers.

If the memory size of Explicit heap is not estimated properly, the Explicit Memory Management functionality is not effective.

In a J2EE server, deploy the following objects that cause increase in the memory size of the Tenured area, in Explicit heap:

- Object for communicating with redirector
- Object related to the HTTP session

You can calculate the memory size of Explicit heap used by the object for communicating with redirector, from the setup value of the definition file. For details about how to estimate, see *5.13.2 Memory size used by the object for communicating with redirector*.

The memory size of Explicit heap used by the object related to HTTP session is actually estimated after acquiring information by running an application. For details on how to estimate the memory size, see *7.11.2 Memory size used by the object related to the HTTP session* in the manual *uCosminexus Application Server System Design Guide*.

As a result of estimating the memory size of Explicit heap, if the memory size of Explicit heap used by the object related to the HTTP session is extremely large, correct the application design by referencing *Appendix A Efficient Usage of the Explicit Heap Used in an HTTP Session* in the manual *uCosminexus Application Server System Design Guide*. When estimating the HTTP session in V9 compatibility mode, specify `false` for the `ejbserver.server.eheap.ajp13.enabled` user property of the J2EE server, so that the object for communication with the redirector is not placed in the Explicit heap.

5.13.2 Memory size used by the object for communicating with redirector

The memory size of Explicit heap used by the object for communicating with redirector is estimated by the following formula:

```
Memory-size-used-by-the-object-for-communicating-with-redirector
=memory-size-used-in-one-connection#×number-of-connections-with-the-redirect
or
```

#

The memory size used in one connection differs depending on the use of the automatic allocation functionality of the Explicit Memory Management functionality. The following is the memory size used in one connection depending on the use of the automatic allocation functionality of the Explicit Memory Management functionality.

Table 5–24: Memory size used in one connection depending on the use of the automatic allocation functionality of the Explicit Memory Management functionality

Item number	Whether the automatic allocation functionality of the Explicit Memory Management functionality is used	Memory size used in one connection
1	Y	144 kilobytes
2	--	128 kilobytes

Legend:

Y: Automatic allocation functionality of the Explicit Memory Management functionality is used.

--: Automatic allocation functionality of the Explicit Memory Management functionality is not used.

Use the maximum connections setup in the Web server as *number-of-connections-with-the-redirector* when system configuration of the Web server and Web container is arranged in a 1-to-1 ratio.

Location for setting maximum connections differs depending on the Web server and the type of the OS used. The following table describes the setup locations:

Table 5–25: Setup locations of maximum connections

Web server	OS	Setup location
Cosminexus HTTP Server	Windows	`ThreadsPerChild` directive of `httpsd.conf` (Cosminexus HTTP Server definition file)
	UNIX	`MaxClients` directive of `httpsd.conf` (Cosminexus HTTP Server definition file)
Microsoft IIS	Windows	'Maximum connections' set in the `<performance>` tag of the Web site property.

5.13.3 How to estimate using statistical information

You can use the statistical information to check the actual usage of Explicit heap with a J2EE server after J2EE server starts operating and for implementing the J2EE server tests. This section describes the procedure for checking the usage of Explicit heap using statistical information.

For details about the output contents of the statistical information and the settings to output the statistical information, and output destination of the statistics file, see *3.3 Statistics File Output Functionality* in the *uCosminexus Application Server Operation, Monitoring, and Linkage Guide*.

(1) Concept of estimations using statistical information

When estimating using statistical information, the memory size of the Explicit heap area required in the system is as follows:

1. Memory size of the Explicit heap area used in HTTP session
2. Memory size of the Explicit heap area used in container excluding the area mentioned in point 1.

3. Memory size of the Explicit heap area used in applications and JavaVM
4. Memory size (*size-of-the-Survivor-area-of-Java-heap* `×2`) of the Explicit heap area used to manage Explicit memory block by JavaVM

You can confirm the memory size of points 1 to 3 from statistical information. In point 4, you can use the memory size of *size-of-the-Survivor-area-of-Java-heap*`×2` when the automatic release function of Explicit Memory Management is enabled.

The following table describes the examples using the Explicit heap area mentioned in the points 1 to 3. The points 1 to 3 correspond to the item numbers 1 to 3 in the table:

Table 5–26: Concrete example of items using the Explicit heap area

No.	Explicit heap area	Concrete example of using the Explicit heap area
1	The Explicit heap area used in a HTTP session	HTTP session
2	The Explicit heap area used in a container	• Objects used for communicating with the redirector • Objects used for managing HTTP sessions
3	The Explicit heap area used in applications and JavaVM	• Application • JavaVM

(2) Precautions when acquiring statistical information used in estimation

Acquire the statistical information used for estimations in the actual environment or an environment same as the actual environment.

When the following items differ from the actual environment, you cannot estimate the appropriate memory size using the statistical information:

- Properties set up in each definition file and values specified in options
- Number of Web applications registered on a server
- Number of connections with the redirector
- Size of the data processed by business applications
- Data processed at regular intervals

Furthermore, specify an option to set a maximum value for the size of the Explicit heap area to avoid completely used status of the Explicit heap area when acquiring the statistical information for estimation.

When you acquire statistical information in a status where maximum size of the Explicit heap area is insufficient, the Explicit heap area would be in a completely used state. You cannot estimate properly if the statistical information is acquired in a state whereby the Explicit area is completed used. You can confirm whether the Explicit heap area is in a completed used status by checking that the value of `EHeapSize.HighWaterMark` of statistical information is same as the value of maximum size of the Explicit heap area. The Explicit heap area is in completed used status when the value of `EHeapSize.HighWaterMark` of statistical information is same as the value of maximum size of the Explicit heap area.

(3) How to estimate

The following points describe how to estimate based on the statistical information:

(a) Memory size of the Explicit heap area used by HTTP session

To calculate the memory size used by the HTTP session, you can use the format for calculating the memory size of the Explicit heap used by the HTTP session described in *7.11.2 Memory size used by the object related to the HTTP session* in the manual *uCosminexus Application Server System Design Guide*. Here, you can confirm the "Memory size used on one session" included in formula from the statistical information.

Memory size used in one session corresponds to the "Maximum size of Explicit memory block" output to statistical information. In the "Maximum size of Explicit memory block", the size of maximum items used is output from the Explicit memory block released in the statistical information collection interval. Therefore, round-out in unit of 64 kilobytes and estimate the Explicit heap. Additionally, when using the automatic allocation functionality of the Explicit Memory Management functionality, add 16 kilobytes, and then estimate the Explicit heap.

When estimating, please see the following values. Also, the number of sessions required in the system correspond to "Number of Explicit memory blocks":

- Maximum size of Explicit memory blocks acquired in HTTP session (Value of `HTTPSessionEMemoryBlockMaxSize.HighWaterMark`)
- Number of Explicit memory blocks acquired in HTTP session (Value of `HTTPSessionEMemoryBlockCount.HighWaterMark`)

(b) Memory size of the Explicit heap area used in container

The memory size of the Explicit heap area used in the container corresponds to "Explicit heap size used in container" of statistical information. Use the maximum value (value of `ContainerEHeapSize.HighWaterMark`) from the acquired value for the statistical information used for estimation.

(c) Memory size of the Explicit heap area used in applications and JavaVM

The memory size of the Explicit heap area used in applications and JavaVM corresponds to "Explicit heap size used in applications" value of statistical information. Use the maximum value (`ApplicationEHeapSize.HighWaterMark`) from the acquired value for the statistical information used for estimation.

(4) Procedure to confirm the statistical information

This point describes the procedure to confirm the statistical information. This point also describes the confirmation procedures with estimation formula of statistical information in (3) as an example. For details about the output contents of the statistical information file, see *3.3 Statistics File Output Functionality* in the *uCosminexus Application Server Operation, Monitoring, and Linkage Guide*.

Estimation formula

```
memory-size-of-the-required-Explicit-heap-area
=(value-where-HTTPSessionEMemoryBlockMaxSize.HighWaterMark-is-rounded-out-in
-64-kilobytes-unit
×HTTPSessionEMemoryBlockCount.HighWaterMark)
+ ContainerEHeapSize.HighWaterMark
+ ApplicationEHeapSize.HighWaterMark
+ size-of-the-Survivor-area-of-Java-heap
×2(only-when-explicit-memory-management-automatic-release-function-is-enable
d)
```

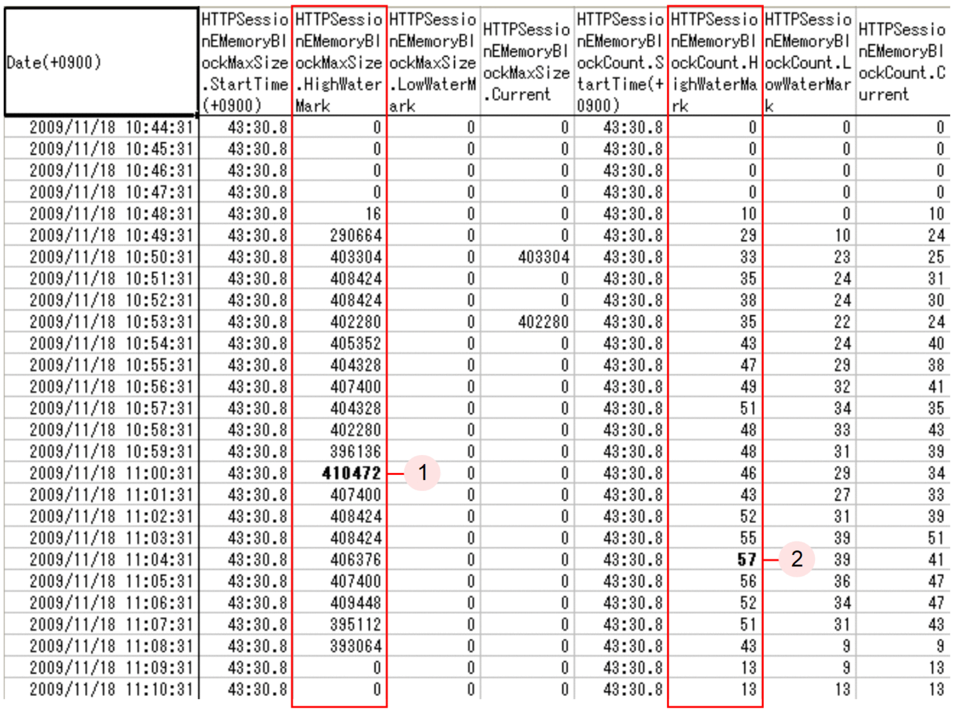
The following points describe how to confirm the respective values:

(a) Memory size of the Explicit heap area used by HTTP session

Confirm the memory size of the Explicit heap area used by
HTTP session from the `HTTPSessionEMemoryBlockMaxSize.HighWaterMark` and `HTTPSessionEMemoryBlockCount.HighWaterMark` values output to statistical information file of JavaVM.

The following figure shows an example to output statistical information of memory size of the Explicit heap area used by HTTP sessions:

Figure 5–29: Example to output statistical information of memory size of the Explicit heap area used by HTTP sessions



Date(+0900)	HTTPSessionEMemoryBlockMaxSize.StartTime(+0900)	HTTPSessionEMemoryBlockMaxSize.HighWaterMark	HTTPSessionEMemoryBlockMaxSize.LowWaterMark	HTTPSessionEMemoryBlockMaxSize.Current	HTTPSessionEMemoryBlockCount.StartTime(+0900)	HTTPSessionEMemoryBlockCount.HighWaterMark	HTTPSessionEMemoryBlockCount.LowWaterMark	HTTPSessionEMemoryBlockCount.Current
2009/11/18 10:44:31	43:30.8	0	0	0	43:30.8	0	0	0
2009/11/18 10:45:31	43:30.8	0	0	0	43:30.8	0	0	0
2009/11/18 10:46:31	43:30.8	0	0	0	43:30.8	0	0	0
2009/11/18 10:47:31	43:30.8	0	0	0	43:30.8	0	0	0
2009/11/18 10:48:31	43:30.8	16	0	0	43:30.8	10	0	10
2009/11/18 10:49:31	43:30.8	290664	0	0	43:30.8	29	10	24
2009/11/18 10:50:31	43:30.8	403304	0	403304	43:30.8	33	23	25
2009/11/18 10:51:31	43:30.8	408424	0	0	43:30.8	35	24	31
2009/11/18 10:52:31	43:30.8	408424	0	0	43:30.8	38	24	30
2009/11/18 10:53:31	43:30.8	402280	0	402280	43:30.8	35	22	24
2009/11/18 10:54:31	43:30.8	405352	0	0	43:30.8	43	24	40
2009/11/18 10:55:31	43:30.8	404328	0	0	43:30.8	47	29	38
2009/11/18 10:56:31	43:30.8	407400	0	0	43:30.8	49	32	41
2009/11/18 10:57:31	43:30.8	404328	0	0	43:30.8	51	34	35
2009/11/18 10:58:31	43:30.8	402280	0	0	43:30.8	48	33	43
2009/11/18 10:59:31	43:30.8	396136	0	0	43:30.8	48	31	39
2009/11/18 11:00:31	43:30.8	**410472**	0	0	43:30.8	46	29	34
2009/11/18 11:01:31	43:30.8	407400	0	0	43:30.8	43	27	33
2009/11/18 11:02:31	43:30.8	408424	0	0	43:30.8	52	31	39
2009/11/18 11:03:31	43:30.8	408424	0	0	43:30.8	55	39	51
2009/11/18 11:04:31	43:30.8	406376	0	0	43:30.8	**57**	39	41
2009/11/18 11:05:31	43:30.8	407400	0	0	43:30.8	56	36	47
2009/11/18 11:06:31	43:30.8	409448	0	0	43:30.8	52	34	47
2009/11/18 11:07:31	43:30.8	395112	0	0	43:30.8	51	31	43
2009/11/18 11:08:31	43:30.8	393064	0	0	43:30.8	43	9	9
2009/11/18 11:09:31	43:30.8	0	0	0	43:30.8	13	9	13
2009/11/18 11:10:31	43:30.8	0	0	0	43:30.8	13	13	13

The maximum value of `HTTPSessionEMemoryBlockMaxSize.HighWaterMark` is 410472 bytes (400.85 kilobytes) acquired at 11:00:31 in the 1. in above figure.

If you round-out this value in unit of 64 kilobytes, the value would be 448 kilobytes. The maximum value of `HTTPSessionEMemoryBlockCount.HighWaterMark` is 57 acquired at 11:04:31 in the 2. in above figure.

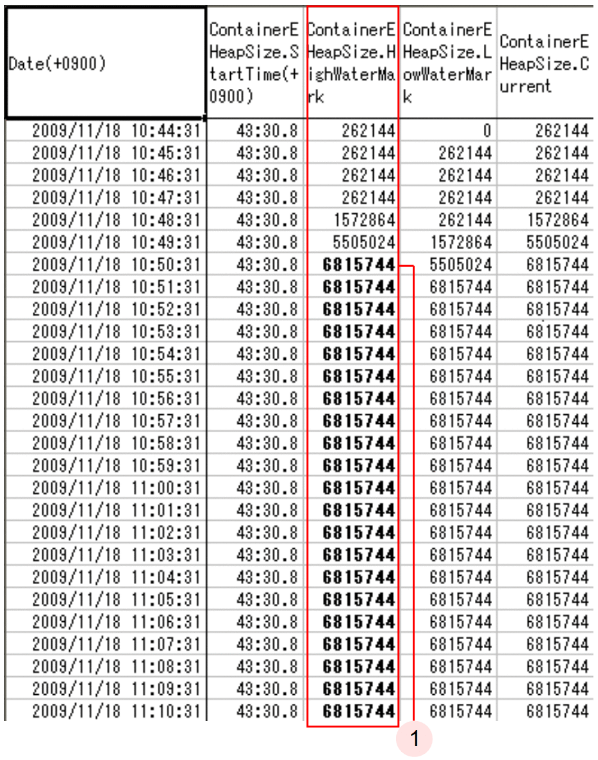
The value acquired by multiplying these two values would be the memory size of the Explicit heap area used by HTTP session.

(b) Memory size of the Explicit heap area used in container

Confirm the memory size of the Explicit heap area used by the container from `ContainerEHeapSize.HighWaterMark` value output to statistical information file of JavaVM.

The following figure shows an example to output statistical information of memory size of the Explicit heap area used in container:

Figure 5–30: Example to output statistical information of memory size of the Explicit heap area used in container



Date(+0900)	ContainerEHeapSize.StartTime(+0900)	ContainerEHeapSize.HighWaterMark	ContainerEHeapSize.LowWaterMark	ContainerEHeapSize.Current
2009/11/18 10:44:31	43:30.8	262144	0	262144
2009/11/18 10:45:31	43:30.8	262144	262144	262144
2009/11/18 10:46:31	43:30.8	262144	262144	262144
2009/11/18 10:47:31	43:30.8	262144	262144	262144
2009/11/18 10:48:31	43:30.8	1572864	262144	1572864
2009/11/18 10:49:31	43:30.8	5505024	1572864	5505024
2009/11/18 10:50:31	43:30.8	**6815744**	5505024	6815744
2009/11/18 10:51:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 10:52:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 10:53:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 10:54:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 10:55:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 10:56:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 10:57:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 10:58:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 10:59:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 11:00:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 11:01:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 11:02:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 11:03:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 11:04:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 11:05:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 11:06:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 11:07:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 11:08:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 11:09:31	43:30.8	**6815744**	6815744	6815744
2009/11/18 11:10:31	43:30.8	**6815744**	6815744	6815744

The maximum value of `ContainerEHeapSize.HighWaterMark` is 6815744 bytes (6656 kilobytes) acquired after 10:50:31 in the 1. in above figure.

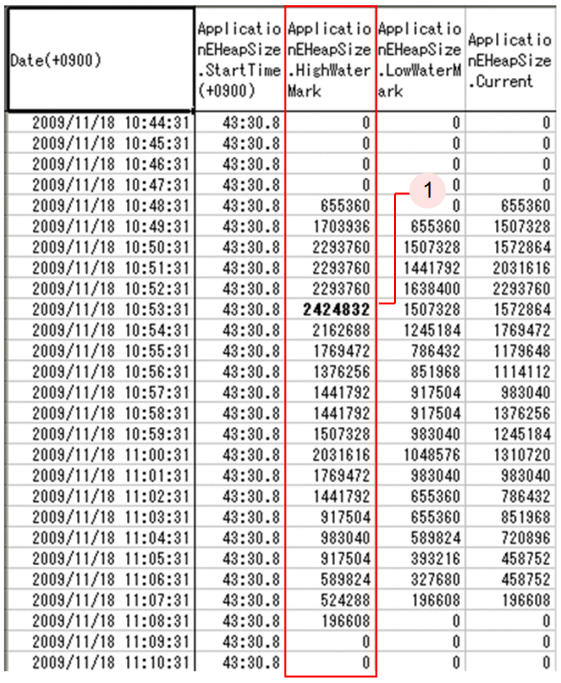
This is the memory size of the Explicit heap area used in container.

(c) Memory size of the Explicit heap area used by applications and JavaVM

Confirm the memory size of the Explicit heap area used by applications and JavaVM from `ApplicationEHeapSize.HighWaterMark` value output to statistical information file of JavaVM.

The following figure shows an example to output statistical information of memory size of the Explicit heap area used by applications and JavaVM:

Figure 5–31: Example to output statistical information of memory size of the Explicit heap area used by applications and JavaVM



Date(+0900)	ApplicationEHeapSize.StartTime(+0900)	ApplicationEHeapSize.HighWaterMark	ApplicationEHeapSize.LowWaterMark	ApplicationEHeapSize.Current
2009/11/18 10:44:31	43:30.8	0	0	0
2009/11/18 10:45:31	43:30.8	0	0	0
2009/11/18 10:46:31	43:30.8	0	0	0
2009/11/18 10:47:31	43:30.8	0	0	0
2009/11/18 10:48:31	43:30.8	655360	0	655360
2009/11/18 10:49:31	43:30.8	1703936	655360	1507328
2009/11/18 10:50:31	43:30.8	2293760	1507328	1572864
2009/11/18 10:51:31	43:30.8	2293760	1441792	2031616
2009/11/18 10:52:31	43:30.8	2293760	1638400	2293760
2009/11/18 10:53:31	43:30.8	**2424832**	1507328	1572864
2009/11/18 10:54:31	43:30.8	2162688	1245184	1769472
2009/11/18 10:55:31	43:30.8	1769472	786432	1179648
2009/11/18 10:56:31	43:30.8	1376256	851968	1114112
2009/11/18 10:57:31	43:30.8	1441792	917504	983040
2009/11/18 10:58:31	43:30.8	1441792	917504	1376256
2009/11/18 10:59:31	43:30.8	1507328	983040	1245184
2009/11/18 11:00:31	43:30.8	2031616	1048576	1310720
2009/11/18 11:01:31	43:30.8	1769472	983040	983040
2009/11/18 11:02:31	43:30.8	1441792	655360	786432
2009/11/18 11:03:31	43:30.8	917504	655360	851968
2009/11/18 11:04:31	43:30.8	983040	589824	720896
2009/11/18 11:05:31	43:30.8	917504	393216	458752
2009/11/18 11:06:31	43:30.8	589824	327680	458752
2009/11/18 11:07:31	43:30.8	524288	196608	196608
2009/11/18 11:08:31	43:30.8	196608	0	0
2009/11/18 11:09:31	43:30.8	0	0	0
2009/11/18 11:10:31	43:30.8	0	0	0

The maximum value of `ApplicationEHeapSize.HighWaterMark` is 2424832 bytes (2368 kilobytes) acquired at 10:53:31 in the 1. in above figure.

(d) Memory size of the required Explicit heap area

The following is the memory size of the required Explicit heap area acquired from the statistical information described in (a) to (c):

```
448 (kilobytes)× 57 + 6656 (kilobytes) + 2368 (kilobytes)=34560 (kilobytes)
≒ 34 megabytes
```

When the auto-release functionality of Explicit Memory Management is enabled, a value with "*Survivor-area-size-of-Java-heap* × 2" added becomes the final estimated size of the Explicit heap area.

6 In-Process HTTP Server

This chapter describes the settings for the in-process HTTP server functionality.

6.1 Organization of this chapter

Application Server provides an in-process HTTP server as Web server functionality. The *in-process HTTP server* is Web server functionality provided in the J2EE server processes. Since the J2EE server processing receives the HTTP request directly without passing through the Web server, you can use the Web server functionality with better processing performance than during the Web server integration.

The following table lists the functionality and the reference sections corresponding to the functionality of the in-process HTTP server:

Table 6–1: Functionality and reference sections corresponding to each functionality of in-process HTTP server

Functionality	Reference
Overview of in-process HTTP server	*6.2*
Controlling the number of connections from the Web client	*6.3*
Controlling the number of request processing threads	*6.4*
Controlling the flow of requests by controlling the number of concurrent connections from the Web client	6.5
Controlling the flow of requests by controlling the number of concurrently executing threads	*6.6*
Request distribution with the redirector	*6.7*
Controlling the communication with the Web client by Persistent Connection	*6.8*
Communication timeout (In-process HTTP server)	*6.9*
Specifying the IP address (In-process HTTP server)	*6.10*
Controlling access by limiting the hosts that are allowed access	*6.11*
Controlling access by limiting the request data size	*6.12*
Controlling access by limiting the HTTP-enabled methods	*6.13*
Customizing responses to the Web client using HTTP responses	6.14
Error page customization (in-process HTTP server)	*6.15*
Notifying the gateway information to the Web container	*6.16*
Output of log and trace	*6.17*
URI decode functionality	*6.18*
Settings for acquiring the in-process HTTP server log	*6.19*
`cjtracesync` (synchronize trace file information for in-process HTTP server)	*6.20*
Precautions when operating SOAP applications	*6.21*

6.2 Overview of in-process HTTP server

This section provides an overview of the in-process HTTP server.

The following table describes the organization of this section.

Table 6–2: Organization of this section (Overview of in-process HTTP server)

Category	Title	Reference
Description	Using the in-process HTTP server	*6.2.1*
	Functionality available in the in-process HTTP server	*6.2.2*
Settings	Execution environment settings (J2EE server settings)	*6.2.3*

Note:

There is no specific description of *Implementation*, *Operations*, and *Notes* for this functionality.

6.2.1 Using the in-process HTTP server

The *in-process HTTP server* is the Web server functionality provided in the J2EE server processes.

As the J2EE server processing receives the HTTP request directly without passing through the Web server, you can use the Web server functionality with even better processing performance than during the Web server integration. Therefore, for a system that emphasizes performance, Hitachi recommends that you use the in-process HTTP server.

However, there are comparative differences in the functionality provided in Cosminexus HTTP Server and Microsoft IIS, you check the differences in the functionality, and then determine whether to use the in-process HTTP server. Furthermore, you cannot use the in-process HTTP server and the Web server integration functionality simultaneously. When you design the system, you must choose which functionality you will use, in advance.

To use the in-process HTTP server, the prerequisites are as follows:

- The in-process HTTP server must be deployed within the internal network in failover instead of deploying the server in DMZ that is accessible from the external networks to which unauthorized access is assumed. In a system accessed from external networks such as Internet, you must build a system in which a proxy server is deployed on DMZ and forwarded to the in-process HTTP server within the internal network.
- In the in-process HTTP server, only HTTP is supported. HTTPS is not supported. To use HTTPS, the SSL accelerator or reverse proxy of Cosminexus HTTP Server is a prerequisite.
- You can use the in-process HTTP server to access only the Web applications deployed on the J2EE server. Note that you cannot deploy static contents alone, but only when you execute request distribution with the redirector and error page customization, you can specify static contents that are not included in the Web application.

Take note of the following when using the in-process HTTP server:

- If you stop the J2EE server by executing the `cjstopsv` command, during the TCP connection of the Web client and the in-process HTTP server, the J2EE server does not stop, until the Web client disconnects the TCP connection of the in-process HTTP server or the timeout specified in the `webserver.connector.inprocess_http.persistent_connection.timeout` key of `usrconf.properties` (user property file for the J2EE server) occurs. If you want to stop the J2EE server regardless of the disconnection of the TCP connection from the Web client or the timeout occurrence, forcibly stop the J2EE server by specifying the `-f` option in the `cjstopsv` command.

Customize the J2EE server properties to specify the settings for the in-process HTTP server. For details on customizing the operation settings for the J2EE server, see *6.2.3 Execution environment settings (J2EE server settings)*.

Tip

The in-process HTTP server is not the default server.

6.2.2 Functionality available in the in-process HTTP server

The following table lists the functionality available in the in-process HTTP server and the reference section of each functionality:

Table 6–3: Functionality available in the in-process HTTP server and references

Functionality name		Reference
Controlling the number of connections from the Web client		*6.3*
Controlling the number of request processing threads from the Web client		*6.4*
Controlling the flow of requests	Controlling the number of concurrent connections from the Web client	*6.5*
	Controlling the number of concurrently executing threads#	*6.6*
Request distribution with the redirector		*6.7*
Controlling communication with the Web client	Controlling communication by Persistent Connection	*6.8*
	Communication timeout that can be set in the in-process HTTP server	*6.9*
	IP address specification used in the in-process HTTP server#	*6.10*
Controlling access from the Web client	Limiting the hosts that are allowed access	*6.11*
	Limiting the request data size	*6.12*
	Limiting the HTTP-enabled methods	*6.13*
Customizing the responses to the Web client	Customizing the HTTP response header	*6.14*
	Customizing the error page	*6.15*
Notifying the gateway information to the Web container#		*6.16*
Output of log and trace		*6.17*

#
The functionality is not different when the in-process HTTP server is not used (when the Web server integration functionality is used).

6.2.3 Execution environment settings (J2EE server settings)

This section describes how to set up the in-process HTTP server.

To receive HTTP requests by using the Web server functionality provided in the J2EE server processes, the system must be built with a configuration using the in-process HTTP server functionality.

Procedure:

1. Enabling the in-process HTTP server functionality.

   Define the specifications for the in-process HTTP server by specifying `true` in the `webserver.connector.inprocess_http.enabled` parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file. By default, `false` is specified. For details on the Easy Setup definition file and parameters to specify, see *Part 3 Reference (V9 Compatibility Mode)*.

2. Specify settings for controlling the number of connections from the Web client and controlling the number of request processing threads.

   By adjusting the number of request processing threads according to the performance of the host that operates the server and the status of access from the client, you can improve the performance of the in-process HTTP server. For details on settings, see *6.3 Controlling the number of connections from the Web client* and *6.4 Controlling the number of request processing threads*.

   Note the following when you set up the in-process HTTP server:

   - If a large number of requests need to be processed immediately after the server starts, specify a big value as the number of request processing threads to be created when the server starts.
   - Note that if you increase the maximum number of spare threads, you can promptly support sudden increase in access, but a lot of resources will be consumed.

3. Specify settings for controlling access from the Web client.

   By enhancing the security for connections and requests sent from the client, you can prevent unauthorized access and attacks on the server from outside. For details on settings, see *6.11 Controlling access by limiting the hosts that are allowed access*, *6.12 Controlling access by limiting the request data size*, and *6.13 Controlling access by limiting the HTTP-enabled methods*.

4. As and when required, specify settings for the functionality that can be used in the in-process HTTP server.

   For details on the functionality available in the in-process HTTP server, see *6.2.2 Functionality available in the in-process HTTP server*.

6.3 Controlling the number of connections from the Web client

By controlling the number of connections and number of request processing threads from the Web client and by optimizing the number of request processing threads, you can constantly control the load on the J2EE server and maintain a stable and high throughput. For details on controlling the number of request processing threads, see *6.4 Controlling the number of request processing threads*.

This section describes the controlling of the number of connections from the Web client.

The following table describes the organization of this section.

Table 6–4: Organization of this section (Controlling the number of connections from the Web client)

Category	Title	Reference
Description	Overview of controlling the number of connections from the Web client	*6.3.1*
Settings	Execution environment settings (J2EE server settings)	*6.3.2*

Note:

There is no specific description of *Implementation*, *Operations*, and *Notes* for this functionality.

In the in-process HTTP server, you can control the number of request processing threads created by the in-process HTTP server by setting the number of Web clients that can connect simultaneously. Also, by pooling the constant number of request processing threads that are not running as the spare threads, the processing required for adding and deleting the request processing threads is restrained to the minimum.

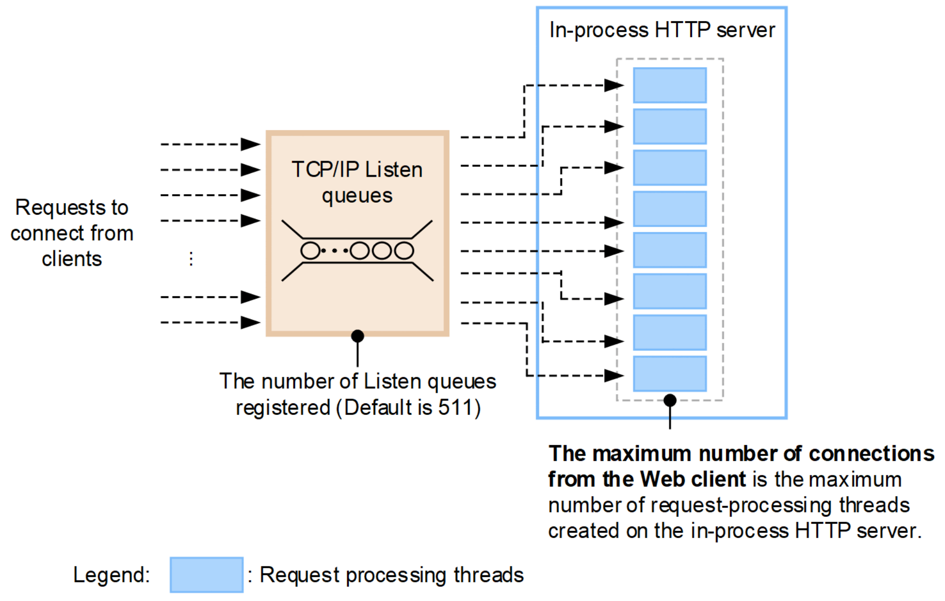
6.3.1 Overview of controlling the number of connections from the Web client

In the in-process HTTP server, you set the maximum number of Web clients and proxy servers connecting simultaneously and control the number of request processing threads. The in-process HTTP server creates the request processing threads for the number of connections from the Web client, and therefore, the maximum number of connections from the Web client becomes the upper limit for the number of request processing threads that are created by the in-process HTTP server.

Note that the connection requests from the client are registered in the TCP/IP Listen queue and are passed to the request processing threads. The connection requests from the client exceeding the upper limit for the number of connections are accumulated in the Listen queue. When the connection requests from the client accumulated in the Listen queue exceed the specified maximum value, the client fails to connect to the server.

The following figure shows an overview of controlling the number of connections from the Web client:

Figure 6–1: Overview of controlling the number of connections from the Web client


In-process HTTP server
TCP/IP Listen queues
Requests to connect from clients
The number of Listen queues registered (Default is 511)
The maximum number of connections from the Web client is the maximum number of request-processing threads created on the in-process HTTP server.
Legend:
: Request processing threads

6.3.2 Execution environment settings (J2EE server settings)

Specify the definition for controlling the number of connections from the Web client in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file.

The following table lists the definition in the Easy Setup definition file for controlling the number of connections from the Web client:

Table 6–5: Definition in the Easy Setup definition file for controlling the number of connections from the Web client

Parameter to be specified	Setting contents
`webserver.connector.inprocess_http.max_connections`	Specifies the maximum number of connections with the Web client and proxy server. The in-process HTTP server creates the request processing threads for the number of connections from the Web client, and therefore, the value specified here becomes the upper limit for the number of request processing threads.
`webserver.connector.inprocess_http.backlog`	The HTTP requests exceeding the upper limit for the number of connections from the Web client are accumulated in the Listen queue. Specify the maximum number of registrations in the Listen queue in this parameter.

For details on the Easy Setup definition file and parameters to specify, see *Part 3 Reference (V9 Compatibility Mode)*.

6.4 Controlling the number of request processing threads

By controlling the number of connections and number of request processing threads from the Web client and by optimizing the number of request processing threads, you can constantly control the load on the J2EE server and maintain a stable and high throughput. For details on controlling the number of connections from the Web client, see *6.3 Controlling the number of connections from the Web client*.

This section describes the controlling of the number of request processing threads. The following table describes the organization of this section.

Table 6–6: Organization of this section (Controlling the number of request processing threads)

Category	Title	Reference
Description	Overview of controlling the number of request processing threads	*6.4.1*
Settings	Execution environment settings (J2EE server settings)	*6.4.2*

Note:

There is no specific description of *Implementation*, *Operations*, and *Notes* for this functionality.

6.4.1 Overview of controlling the number of request processing threads

After the request processing threads are created when the in-process HTTP server starts, the status of the request processing threads and the number of threads are monitored periodically. When requests are concentrated in the in-process HTTP server, the request processing threads are added and the adequate number of spare threads is pooled in advance. When there are few requests, the extra pooled spare threads will be deleted.

The controlling of the number of request processing threads is executed as follows:

1. When the J2EE server starts, the specified number of request processing threads is created.
2. While the J2EE server is running, the number of request processing threads is monitored.
3. During monitoring, if the number of spare threads is smaller than the specified minimum value, the request processing threads are added and pooled as the spare threads. Also, if the number of spare threads is greater than the specified maximum value, the extra spare threads are deleted.

   Note that you can also maintain the number of threads created when the J2EE server starts. When maintaining the number of threads created at startup, if the total number of request processing threads and spare threads is less than the number of threads created when the Web server starts, even if the number of spare threads exceeds the maximum value, the spare threads are not deleted. For example, if the number of threads created when the Web server starts is 8 and the maximum number of spare threads is 5, the spare threads are not deleted in the case when the number of request processing threads is 2 and the number of spare threads is 6.

The transition of the number of request processing threads is explained with the following examples:

- **Transition example 1**

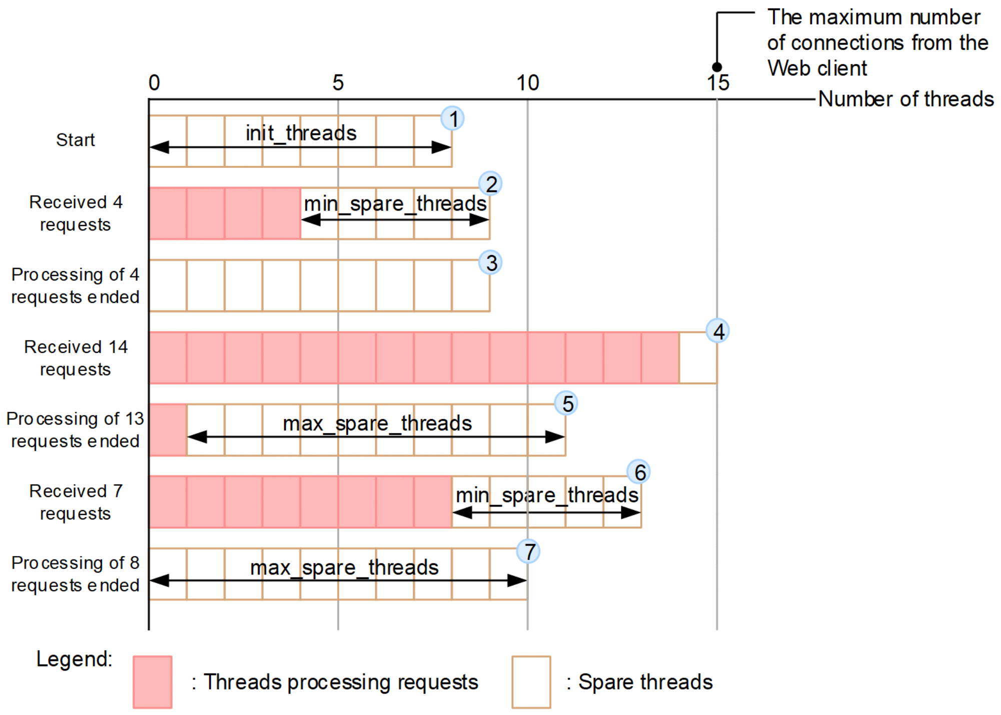
  Assume the following settings:

  - Maximum number of connections from the Web client: 15
  - Number of registrations in the Listen queue: 100
  - Number of request processing threads created when the J2EE server starts: 8
  - Minimum number of spare threads: 5

- Maximum number of spare threads: 10
- Maintenance of the number of threads created when the J2EE server starts: Disabled

The following figure shows the transition example for the number of request processing threads:

Figure 6–2: Transition example for the number of request processing threads



Stages 1. to 7. of the figure are explained below:

1. When the J2EE server starts, the specified number (8 threads) of request processing threads is created.
2. When 4 requests are received, the number of spare threads becomes 4 and since this number is less than the minimum value, 1 thread is added.
3. When the processing of the 4 requests ends, the number of spare threads becomes 9. Since this number is less than the maximum value and more than the minimum value, the current state is maintained.
4. When 14 requests are received, the number of spare threads is less than the minimum value, but the maximum number of connections from the Web client is reached, therefore, the spare threads add only 1 thread.
5. When the processing of 13 requests ends, the number of spare threads exceeds the maximum value, so 4 threads are deleted.
6. When 7 requests are received, the number of spare threads is less than the minimum value, so 2 threads are added.
7. When the processing of 8 requests ends, the number of spare threads exceeds the maximum value, so 3 threads are deleted.

- **Transition example 2**

  By setting the maximum number of spare threads equal to the maximum number of connections from the Web client, you can continue to use the request processing threads created once without deleting them.

  The transition example for the number of request processing threads when the maximum number of spare threads is equal to the maximum number of connections from the Web client is as follows:

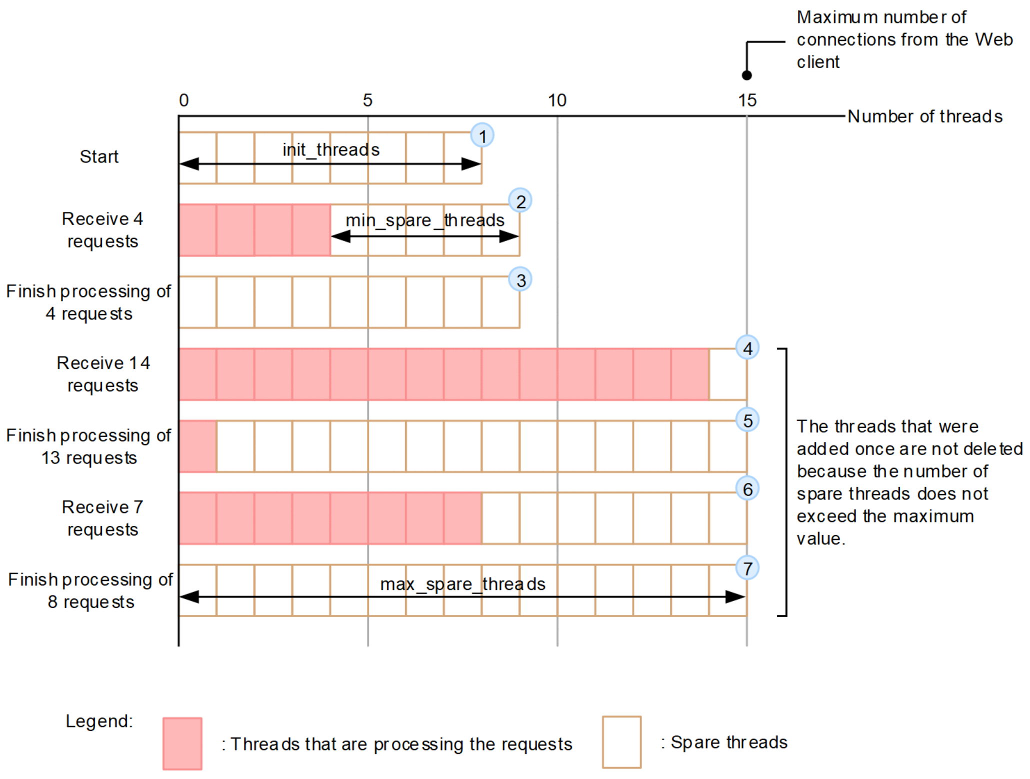
  Assume the following settings:

  - Maximum number of connections from the Web client: 15
  - Maximum number of registrations in the Listen queue: 100

- Number of request processing threads created when the J2EE server starts: 8
- Minimum number of spare threads: 5
- Maximum number of spare threads: 15
- Maintenance of the number of threads created when the J2EE server starts: Disabled

The following figure shows the transition example for the number of request processing threads, when the maximum number of spare threads is equal to the maximum number of connections from the Web client:

Figure 6–3: Transition example for the number of request processing threads when the maximum number of spare threads is equal to the maximum number of connections from the Web client


Maximum number of connections from the Web client
0
5
10
15
Number of threads
Start
init_threads
1
Receive 4 requests
min_spare_threads
2
Finish processing of 4 requests
3
Receive 14 requests
4
Finish processing of 13 requests
5
Receive 7 requests
6
Finish processing of 8 requests
max_spare_threads
7
The threads that were added once are not deleted because the number of spare threads does not exceed the maximum value.
Legend:
: Threads that are processing the requests
: Spare threads

Stages 1. to 7. of the figure are explained below:

1. When the J2EE server starts, the specified number (8 threads) of request processing threads is created.
2. When 4 requests are received, the number of spare threads becomes 4 and since this number is less than the minimum value, 1 thread is added.
3. When the processing of the 4 requests ends, the number of spare threads becomes 9. Since this number is less than the maximum value and more than the minimum value, the current state is maintained.
4. When 14 requests are received, the number of spare threads is less than the minimum value, but so that the total number of request processing threads does not exceed the maximum number of connections from the Web client, spare threads add only 1 thread.
5. When the processing of 13 requests ends, the number of spare threads becomes 14, but this number is less than the maximum value and more than the minimum value, so the current state is maintained.
6. When 7 requests are received, the number of spare threads becomes 7, but this number is less than the maximum value and more than the minimum value, so the current state is maintained.
7. When the processing of 8 requests ends, the number of spare threads becomes 15, but this number is less than the maximum value and more than the minimum value, so the current state is maintained.

- **Setup example 3**

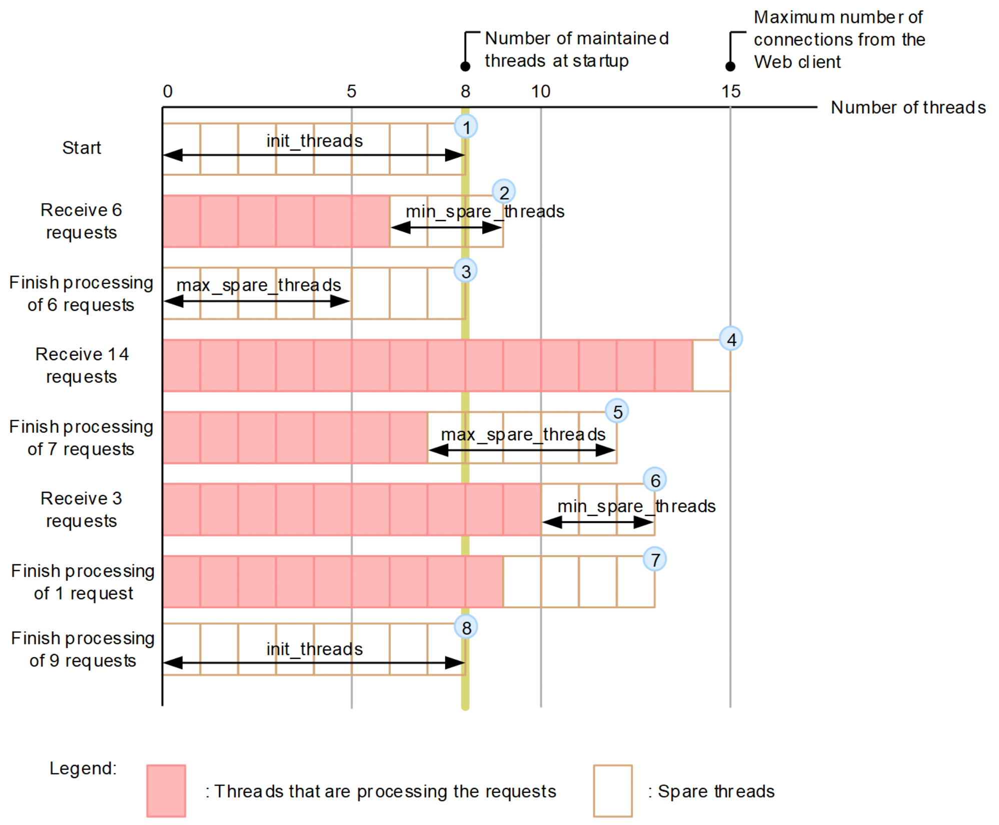
The following is a transition example for the number of request processing threads when the number of threads created at server startup is maintained:

Assume the following settings:

- Maximum number of connections from the Web client: 15
- Maximum number of registrations in the Listen queue: 100
- Number of request processing threads created when the J2EE server starts: 8
- Minimum number of spare threads: 3
- Maximum number of spare threads: 5
- Maintenance of the number of threads created when the J2EE server starts: Enabled

The following figure shows the transition example for the number of request processing threads when the number of threads created at server startup is maintained:

Figure 6–4: Transition example for the number of request processing threads when the number of threads created at J2EE server startup is maintained



Number of maintained threads at startup
Maximum number of connections from the Web client
0
5
8
10
15
Number of threads
Start
init_threads
1
Receive 6 requests
min_spare_threads
2
Finish processing of 6 requests
max_spare_threads
3
Receive 14 requests
4
Finish processing of 7 requests
max_spare_threads
5
Receive 3 requests
min_spare_threads
6
Finish processing of 1 request
7
Finish processing of 9 requests
init_threads
8
Legend:
: Threads that are processing the requests
: Spare threads

Stages 1. to 8. of the figure are explained below:

1. When the J2EE server starts, the specified number (8 threads) of request processing threads is created.
2. When 6 requests are received, the number of spare threads becomes 2. Since this number is less than the minimum value, 1 thread is added.
3. When the processing of the 6 requests ends, the number of spare threads becomes 9 and the maximum value is exceeded, but in order to maintain the number of threads created at server startup, only 1 thread is deleted.
4. When 14 requests are received, the number of spare threads is less than the minimum value, but so that the total number of request processing threads does not exceed the maximum number of connections from the Web client, spare threads add only 1 thread.

5. When the processing of 7 requests ends, the number of spare threads becomes 8 and exceeds the maximum value, so 3 threads are deleted.
6. When 3 requests are received, the number of spare threads becomes 2 and is less than the minimum value, so 1 thread is added.
7. When the processing of 1 request ends, the number of spare threads becomes 4, but this number is less than the maximum value and more than the minimum value, so the current status is maintained.
8. When the processing of 9 requests ends, the number of spare threads becomes 13. This number is more than the maximum value, but in order to maintain the number of threads at J2EE server startup, only 5 threads are deleted.

6.4.2 Execution environment settings (J2EE server settings)

Specify the definition for controlling the number of request processing threads in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file.

The following table lists the definitions in the Easy Setup definition file for controlling the number of request processing threads:

Table 6–7: Definitions in the Easy Setup definition file for controlling the number of request processing threads

Parameter to be specified	Setting contents
`webserver.connector.inprocess_http.init_threads`	Specify the number of request processing threads created when the J2EE server starts.
`webserver.connector.inprocess_http.min_spare_threads`	Specify the minimum number of spare threads. If the number of spare threads is less than the specified minimum value, the request processing threads are added and are pooled as the spare threads.
`webserver.connector.inprocess_http.max_spare_threads`	Specify the maximum number of spare threads. If the number of spare threads is more than the specified maximum value, the surplus spare threads are deleted. By setting the maximum number of spare threads equal to the maximum number of connections from the Web client, you can continue to use the request processing threads created once without deleting them.
`webserver.connector.inprocess_http.keep_start_threads`	Specify whether or not to maintain the request processing threads created when the J2EE server starts.

For details on the Easy Setup definition file and parameters to specify, see *Part 3 Reference (V9 Compatibility Mode)*.

6.5 Controlling the flow of requests by controlling the number of concurrent connections from the Web client

This section describes the controlling of the flow of requests by controlling the number of concurrent connections from the Web client.

The following table describes the organization of this section.

Table 6–8: Organization of this section (Controlling the flow of requests by controlling the number of concurrent connections from the Web client)

Category	Title	Reference
Description	Controlling the number of concurrent connections from the Web client	*6.5.1*
Settings	Execution environment settings (J2EE server settings)	*6.5.2*

Note:
There is no specific description of *Implementation*, *Operations*, and *Notes* for this functionality.

6.5.1 Controlling the number of concurrent connections from the Web client

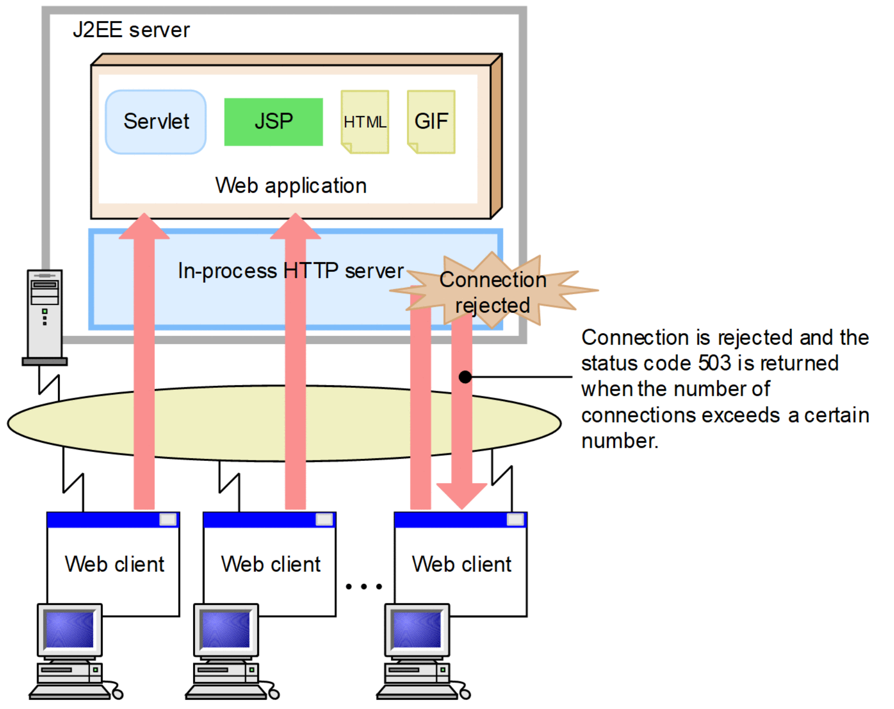
In the in-process HTTP server, you control the number of concurrent connections from the Web client by specifying the maximum number of connections from the Web client along with the number of requests for which connection is rejected.

When the number of connections from the Web client increases or if the load on the J2EE server increases with the effect of the J2EE applications, the Web client can instantly receive a response by rejecting the receipt of requests from the Web client and by returning an error immediately. As a result, the load on the J2EE server is controlled constantly and the response time for the request can be maintained.

The value obtained by subtracting the number of requests for which the connection is rejected from the maximum number of connections by the Web client is the number of requests for which connection is approved by the Web client.

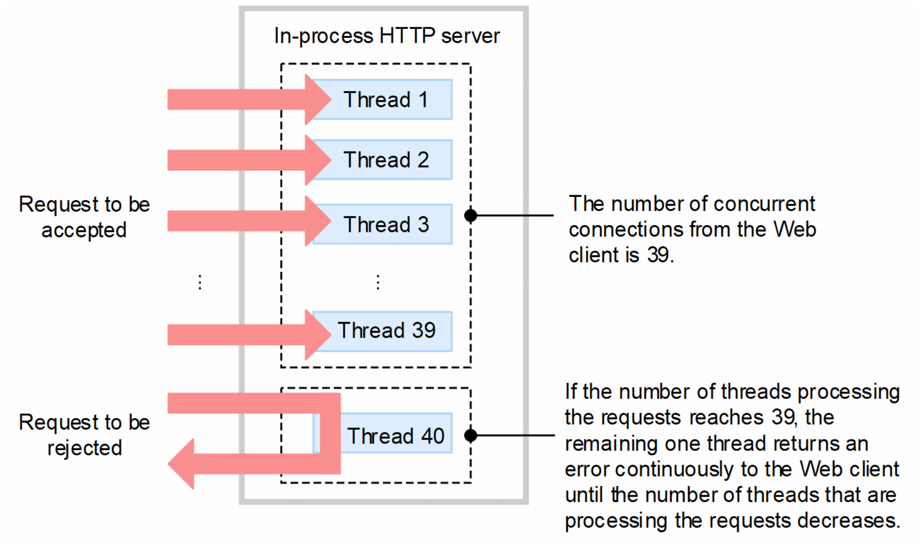
The following figure shows an overview of controlling the number of concurrent connections from the Web client:

Figure 6–5: Overview of controlling the number of concurrent connections from the Web client


J2EE server
Servlet
JSP
HTML
GIF
Web application
In-process HTTP server
Connection rejected
Connection is rejected and the status code 503 is returned when the number of connections exceeds a certain number.
Web client
Web client
Web client

For example, assuming the maximum number of connections from the Web client is 40 and the number of requests for which connection is rejected is 1, the number of Web clients that can concurrently process the requests by connecting to the in-process HTTP server is 39. If the number of threads that are processing the requests becomes 39, the remaining 1 thread continues to return an error for the received requests until the number of request processing threads (under processing) will reduce. The following figure shows an example of controlling the number of concurrent connections from the Web client:

Figure 6–6: Example of controlling the number of concurrent connections from the Web client


In-process HTTP server
Thread 1
Thread 2
Thread 3
Thread 39
Thread 40
Request to be accepted
Request to be rejected
The number of concurrent connections from the Web client is 39.
If the number of threads processing the requests reaches 39, the remaining one thread returns an error continuously to the Web client until the number of threads that are processing the requests decreases.

By controlling the number of concurrent connections from the Web client, an error with the status code 503 is returned to the Web client for the request for which connection is rejected. At this time, if you customize the error page returned to the client, you can customize the response message or redirect to another server.

For details on the settings for customizing the responses to the Web client (in the in-process HTTP server), see *6.14 Customizing responses to the Web client using HTTP responses* and *6.15 Error page customization (In-process HTTP server)*.

Important note

When displaying a page with a frame or a page with an inserted image, the multiple requests might be received from the Web client. In this case, the pages displayed by controlling the number of concurrent connections from the Web client might result in partial errors.

6.5.2 Execution environment settings (J2EE server settings)

To control the number of concurrent connections from the Web client, you must set a J2EE server.

The setting method and example of controlling the number of concurrent connections from the Web client are described here.

(1) How to set

Specify the definition for controlling the number of concurrent connections from the Web client in the following parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server), in the Easy Setup definition file:

`webserver.connector.inprocess_http.rejection_threads`

Specifies the number of requests for which connection is rejected.

For details on the Easy Setup definition file and parameters to specify, see *Part 3 Reference (V9 Compatibility Mode)*.

Important note

When displaying a page with a frame or a page with an image inserted, multiple requests might be received from the Web client. In this case, the pages displayed by the controlling of the number of concurrent connections from the Web client might result in partial errors.

(2) Example settings

The example of settings for controlling the number of concurrent connections from the Web client is described here.

The following is an example of settings wherein the upper limit of number of request processing threads is 40 and the number of request processing threads for which the connection is rejected is 1:

```
...
<param>
  <param-name>webserver.connector.inprocess_http.max_connections</param-nam
e>
  <param-value>40</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.rejection_threads</param-na
me>
  <param-value>1</param-value>
```

```
</param>
...
```

In this example, the number of Web clients that can process requests concurrently after connection is 39. If the number of threads that are processing requests reaches 39, the remaining 1 thread keeps returning error to the Web client.

By controlling the number of concurrent connections from the Web client, an error of the status code 503 (Service Unavailable) is returned to the Web client for the requests for which connection is rejected. At this time if you customize the error page returned to the client, you can customize the response message or redirect to another server. The following are the setting examples for each of these cases. For details on customizing the error page, see *6.15 Error page customization (In-process HTTP server)*.

- **To customize the response message**

  The following is an example of settings for returning a specific file as the response body to the Web client when the connection from the Web client is rejected:

  ```
  ...
  <param>
    <param-name>webserver.connector.inprocess_http.rejection_threads</param-
  name>
    <param-value>3</param-value>
  </param>
  <param>
    <param-name>webserver.connector.inprocess_http.error_custom.list</param-
  name>
    <param-value>REJECTION_1</param-value>
  </param>
  <param>
    <param-name>webserver.connector.inprocess_http.error_custom.REJECTION_1.
  status</param-name>
    <param-value>503</param-value>
  </param>
  <param>
    <param-name>webserver.connector.inprocess_http.error_custom.REJECTION_1.
  file</param-name>
    <param-value>C:/data/busy.html</param-value>
  </param>
  <param>
    <param-name>webserver.connector.inprocess_http.error_custom.REJECTION_1.
  file.content_type=text/html; charset</param-name>
    <param-value>ISO-8859-1</param-value>
  </param>
  <param>
    <param-name>webserver.connector.inprocess_http.error_custom.REJECTION_1.
  request_url</param-name>
    <param-value>/*</param-value>
  </param>
  ...
  ```

  In this example, `503` is specified in the error status code, `C:/data/busy.html` is specified in the corresponding error page file, '`text/html; charset=ISO-8859-1` (Media-Type is `text/html` and `ISO-8859-1` character set is used)' is specified in the Content-Type header of the response, and `/*` is specified in the URL pattern. Therefore, when error status code '503' occurs, regardless of the request URI, the contents of `C:/data/busy.html` file are returned as a response.

- **To redirect a request to another server**

The following is an example of settings for redirecting a request for which connection is rejected to another server:

```
...
<param>
  <param-name>webserver.connector.inprocess_http.rejection_threads</param-
name>
  <param-value>3</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.error_custom.list</param-
name>
  <param-value>REJECTION_1</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.error_custom.REJECTION_1.
status</param-name>
  <param-value>503</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.error_custom.REJECTION_1.
redirect_url</param-name>
  <param-value>http://host1/busy.html</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.error_custom.REJECTION_1.
request_url</param-name>
  <param-value>/*</param-value>
</param>
<param>
...
```

In this example, `503` is specified in error status code, `http://host1/busy.html` is specified in the corresponding error page file, and `/*` is specified in the URL pattern. Therefore, when error status code '503' occurs, regardless of the request URI, all the requests are redirected to the `http://host1/busy.html` URL.

6.6 Controlling the flow of requests by controlling the number of concurrently executing threads

This section describes the controlling the flow of requests by controlling the number of concurrently executing threads.

The following table describes the organization of this section.

Table 6–9: Organization of this section (Controlling the flow of requests by controlling the number of concurrently executing threads)

Category	Title	Reference
Description	Overview of controlling the flow of requests by controlling the number of concurrently executing threads	*6.6.1*
Settings	Execution environment settings (J2EE server settings)	*6.6.2*

Note:

There is no specific description of *Implementation*, *Operations*, and *Notes* for this functionality.

6.6.1 Overview of controlling the flow of requests by controlling the number of concurrently executing threads

In the Web container, the servlet requests are processed in multi-threads. At this time, you can set an upper limit on the number of threads that can be executed concurrently for avoiding the decrease in performance due to slashing. If you set an appropriate number of threads, you can tune the performance as per the access status.

6.6.2 Execution environment settings (J2EE server settings)

To control the number of concurrently executing threads in the Web container, you must set up a J2EE server.

This section describes the settings for controlling the number of concurrently executing threads in the Web container.

The methods for controlling the number of concurrently executing threads include the controlling in Web container, in Web application, and in URL group.

(1) Controlling in the Web container

To control the number of concurrently executing threads in the Web container, set the maximum number of threads that can be executed concurrently in the entire Web container. The number of threads set here is shared in all the Web applications deployed on the Web container.

Specify the definition for controlling the number of concurrently executing threads in the Web container in the following parameter within the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file.

`webserver.connector.inprocess_http.max_execute_threads`

Set the maximum number of threads that you can concurrently execute in the entire Web container.

For details on the Easy Setup definition file and parameters to specify, see *Part 3 Reference (V9 Compatibility Mode)*.

(2) Controlling in the Web application

Note that when you control the number of concurrently executing threads in the Web application, you also need to simultaneously specify the settings to control the number of threads in the Web container.

Specify settings for controlling the number of concurrently executing threads in the Web application using the Easy Setup definition file and the server management commands. The method of setup is the same as is used for the Web server integration functionality. For details on how to set the number of concurrently executing threads for each Web application when the Web server integration functionality is used, see the manual *uCosminexus Application Server Web Container Functionality Guide*.

(3) Controlling in the URL group

To control the number of concurrently executing threads in the URL group, you must also specify settings for controlling the number of concurrently executing threads in the Web application and the settings for controlling the number of threads in the Web container simultaneously.

Specify settings for controlling the number of concurrently executing threads in the URL group by using the server management commands. The method of setup is the same as is used for the Web server integration functionality. For details on how to set the number of concurrently executing threads for each URL group when the Web server integration functionality is used, see the manual *uCosminexus Application Server Web Container Functionality Guide*.

6.7 Request distribution with the redirector

This section explains the distribution of requests with the redirector.

In the in-process HTTP server, you can distribute the requests by the URL patterns included in an HTTP request. You can also customize the responses for the distributed requests and return them to the client.

This section describes the distribution of requests by the URL pattern and the process of customizing the responses. The section also provides an overview of the settings for request distribution with the redirector.

The following table describes the organization of this section.

Table 6–10: Organization of this section (Request distribution with the redirector)

Category	Title	Reference
Description	Distributing requests by URL pattern	*6.7.1*
	Response customization	*6.7.2*
Settings	Execution environment settings (J2EE server settings)	*6.7.3*
Notes	Precautions related to request distribution with the redirector	*6.7.4*

Note:

There is no specific description of *Implementation* and *Operations* for this functionality.

6.7.1 Distributing requests by URL pattern

In the in-process HTTP server, of the HTTP requests for the in-process HTTP server, you can distribute and process requests for a specific URL to a specified Web container. As a result, when the Web application is moved to another J2EE server due to the reasons such as changes in the system configuration, the request to the old URL can be forwarded to the new URL.

Also, in the distribution of requests by redirecting to the in-process HTTP server, the requests for specific Web applications and specific servlets and JSPs in the Web application can be temporarily redirected to another Web server. In the in-process HTTP server, the requests are redirected regardless of whether the resources for the requested servlets and JSPs actually exist. The redirection is given a higher priority than the servlets and JSPs. Therefore, when the requests to the servlets and JSPs match with the redirected URL, the servlets and JSPs are not executed.

6.7.2 Response customization

You can also customize a specific file to return as a response for the requests to a specific URL. If the status code of the response for the request to a redirected URL is 300 to 307, the response body is auto-generated and the response is returned to the client. You can also use a specified file as the response body. When you specify the file, specify the Content-Type header of the response as well.

For details on the status code for which the response is auto-generated and for the settings for request distribution with the redirector (in the in-process HTTP server), see *6.7.3 Execution environment settings (J2EE server settings)*.

6.7.3 Execution environment settings (J2EE server settings)

This section describes the settings for distributing requests with the redirector.

(1) Overview

In the in-process HTTP server, you can distribute requests by the URL pattern included in the HTTP request. You can also customize the response for the distributed request and return a specific file to the client. When the response status code for the request to a redirected URL is 300, the response body is auto-generated and the response is returned to the client. Also, you can use a specified file as a response body. When you specify the file, also specify the Content-Type header of the response.

The auto-generated response body is as follows:

```
<HTML><HEAD>
<TITLE>Status code and explanation</TITLE>
</HEAD><BODY>
<H1>Status code and explanation</H1>
</BODY></HTML>
```

The status code for which the response body is auto-generated and the description is as follows:

- 300 Multiple Choices
- 301 Moved Permanently
- 302 Found
- 303 See Other
- 305 Use Proxy
- 307 Temporary Redirect

When the reading of the file used as the response body during request processing fails, if 300 is specified as the status code, the response body is auto-generated and returned to the client. If 200 is specified as the status code, status 500 error is returned to the client.

(2) How to set

Specify the definition for distributing requests with the redirector in the `<configuration>` tag of the logical J2EE server (j2ee-server), in the Easy Setup definition file.

The following table lists the definitions in the Easy Setup definition file for distributing requests with the redirector:

Table 6–11: Definitions in the Easy Setup definition file for distributing requests with the redirector

Parameter to be specified	Setting contents
`webserver.connector.inprocess_http.redirect.list`	Specifies the redirect definition name.
`webserver.connector.inprocess_http.redirect.`*redirect-difinition-name*`.request_url`	Specifies the URL of the redirected request as an absolute path beginning with a forward slash (`/`).
`webserver.connector.inprocess_http.redirect.`*redirect-difinition-name*`.redirect_url`	Specifies the URL for redirecting the request. Note that when 200 is specified as the status code, the URL cannot be specified.

Parameter to be specified	Setting contents
`webserver.connector.inprocess_http.redirect.`*redirect-difinition-name*`.status`	Specifies the response status code used when redirection is executed.
`webserver.connector.inprocess_http.redirect.`*redirect-difinition-name*`.file`	Specifies the file to be used as the response body when a specific file is returned to the client as a response. Note that when 200 is specified as the status code, the file to be used must be specified.
`webserver.connector.inprocess_http.redirect.`*redirect-difinition-name*`.file.content_type`	Specifies the Content-Type header of the file to be used as the response body when a specific file is returned as a response to the client.

For details on the Easy Setup definition file and parameters to specify, see *Part 3 Reference (V9 Compatibility Mode)*.

Important note

- When the HTTP response is an HTML file, if another file (such as an image file) is being referenced from that HTML, the file might not be properly displayed in the browser.
- If the value specified in the redirect URL matches with the value specified for the request URL, note that the client keeps redirecting the requests.
- When a session is managed by URL rewriting, the session cannot be inherited even if the request is redirected to the same Web application as the request URL.

(3) Example settings

An example of settings for request distribution with the redirector is as follows:

- **Setup example 1**

  An example of request distribution with the redirector is as follows:

```
...
<param>
  <param-name>webserver.connector.inprocess_http.redirect.list</param-nam
e>
  <param-value>REDIRECT_1,REDIRECT_2</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.redirect.REDIRECT_1.reque
st_url</param-name>
  <param-value>/index.html</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.redirect.REDIRECT_1.redir
ect_url</param-name>
  <param-value>http://host1/new_dir/index.html</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.redirect.REDIRECT_1.statu
s</param-name>
  <param-value>302</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.redirect.REDIRECT_2.reque
st_url</param-name>
  <param-value>/old_dir/*</param-value>
```

```
</param>
<param>
  <param-name>webserver.connector.inprocess_http.redirect.REDIRECT_2.redir
ect_url</param-name>
  <param-value>http://host1/new_dir/</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.redirect.REDIRECT_2.statu
s</param-name>
  <param-value>301</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.redirect.REDIRECT_2.file<
/param-name>
  <param-value>C:/data/301.html</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.redirect.REDIRECT_2.file.
content_type</param-name>
  <param-value>text/html; charset=ISO-8859-1</param-value>
</param>
...
```

In this example, `REDIRECT_1` and `REDIRECT_2` are used as the redirect definition names. In `REDIRECT_1`, the request to `/index.html` is redirected to `http://host1/new_dir/index.html` with status code '302'. In `REDIRECT_2`, the requests to `/old_dir/` are redirected under `http://host1/new_dir/` with status '301'. Also, `C:/data/301.html` is used as the response body and `text/html; charset=ISO-8859-1` is used as the Content-Type header.

- **Setup example 2**

  When a wild card is used as a request URL and a value ending with a forward slash (`/`) is specified in the redirect URL, the value set in the Location header of the response becomes 'Value specified in the redirect URL' + 'Actual path from the wild card of the request URL'.

```
...
<param>
  <param-name>webserver.connector.inprocess_http.redirect.list</param-nam
e>
  <param-value>REDIRECT_3</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.redirect.REDIRECT_3.reque
st_url</param-name>
  <param-value>/dir1/*</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.redirect.REDIRECT_3.redir
ect_url</param-name>
  <param-value>http://host/dir2/</param-value>
</param>
...
```

In this example, when the actual request URL is `/dir1/subdir1/index.html`, `http://host/dir2/subdir1/index.html` is set in the Location header. Note that even when a wild card is used in the request URL, if the redirect URL does not end with a forward slash (`/`), the value of the Location header is the same as the redirect URL.

When redirect is executed, if a query string is added to the actual request URL, the value set in the Location header is a value wherein the query string is added to the value specified for the redirect URL. You can also specify a value with the query string added in the redirect URL. In this case, if a query string is also added to the actual request URL, the value set in the Location header is a value wherein the request query string is added behind the value specified for the redirect URL.

6.7.4 Precautions related to request distribution with the redirector

The precautions related to request distribution with the redirector are as follows:

- When the HTTP response is an HTML file, if another file (such as an image file) is being referenced from that HTML, the file might not be properly displayed in the browser.
- If the value specified in the redirect URL matches with the value specified for the request URL, note that the client keeps redirecting the requests.
- When a session is managed by URL rewriting, the session cannot be inherited even if the request is redirected within the Web application.

6.8 Controlling the communication with the Web client by persistent connection

This section describes the controlling of communication with the Web client by Persistent Connection.

The following table describes the organization of this section.

Table 6–12: Organization of this section (Controlling communication with the Web client by Persistent Connection)

Category	Title	Reference
Description	Controlling communication by Persistent Connection	*6.8.1*
Settings	Execution environment settings (J2EE server settings)	*6.8.2*

Note:

There is no specific description of *Implementation* and *Operations* for this functionality.

6.8.1 Controlling communication by Persistent Connection

The *persistent connection* is functionality used for connecting the TCP connection established between the Web client and the in-process HTTP server and keep using the TCP connection between multiple HTTP requests. By using the persistent connection, the time required for establishing a connection between the Web client and Web server is reduced and an attempt is made to reduce the processing time and lessen the communication traffic.

In the in-process HTTP server, you set up the following items to control communication by the Persistent Connection:

- **Upper limit for the number of persistent connections**

  By setting the upper limit for the number of persistent connections, you control the number of Web clients that can continuously process requests with one TCP connection. If the number of TCP connections exceeds the specified upper limit, the connection disconnects after the processing of the request ends. As a result, you can secure the threads for processing the new requests and prevent the request processing threads from being used exclusively by a specific client.

- **Upper limit for the request processing frequency of persistent connection**

  By setting the maximum value for the request processing frequency of the persistent connection, you can control the processing when there are continuous requests from the same Web client.

  When the request processing frequency of the persistent connection exceeds the specified upper limit, the connection disconnects after the processing of the request ends. As a result, you can prevent the request processing threads from being used exclusively by a specific client.

- **Persistent connection timeout**

  By setting a timeout for the request waiting time of the persistent connection, you control the request waiting time of the persistent connection. If there is no request to process requests until the specified timeout period expires, the TCP connection disconnects. As a result, you can prevent the TCP connection from being continuously occupied in an unused state. Also, even if you specify 0 for the request waiting time of Persistent Connection so that a timeout does not occur, if the number of requests exceeds the upper limit on the number of requests that can be processed, the connection is disconnected.

Note that a disconnected Web client will try to connect and send requests again.

6.8.2 Execution environment settings (J2EE server settings)

To use controlling of communication by the persistent connection, you must set up the J2EE server.

This section describes the settings and examples for controlling communication by the persistent connection.

(1) Setting up the J2EE server

Implement the J2EE server settings in the Easy Setup definition file. Specify the definition for controlling communication by the persistent connection in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file.

The following table lists the definitions in the Easy Setup definition file for controlling communication by the persistent connection.

Table 6–13: Definitions in the Easy Setup definition file for controlling communication by persistent connection

Parameter to be specified	Setting contents
`webserver.connector.inprocess_http.persistent_connection.max_connections`	Specifies the upper limit for the number of persistent connections to control the number of Web clients that can process requests continuously with one TCP connection.
`webserver.connector.inprocess_http.persistent_connection.max_requests`	Specifies the upper limit for the request processing frequency of the persistent connection to control the processing when there are continuous requests from the same Web client.
`webserver.connector.inprocess_http.persistent_connection.timeout`	Specifies the timeout value for persistent connection to control the request waiting time of the persistent connection.

For details on the Easy Setup definition file and parameters to specify, see *Part 3 Reference (V9 Compatibility Mode)*.

(2) Example settings

The following is an example of settings for controlling communication by the persistent connection:

```
...
<param>
  <param-name>webserver.connector.inprocess_http.persistent_connection.max_c
onnections</param-name>
  <param-value>5</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.persistent_connection.max_r
equests</param-name>
  <param-value>100</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.persistent_connection.timeo
ut</param-name>
  <param-value>15</param-value>
</param>
...
```

In this example, when the number of TCP connections exceeds `5` or when the request processing frequency exceeds `100`, the TCP connection disconnects after the processing of the request ends. Also, if there is no request to process requests even after the timeout period of `15` seconds passes, the TCP connection disconnects.

6.9 Communication timeout (In-process HTTP server)

This section describes the controlling of communication with the Web client through communication timeout in the in-process HTTP server.

The following table describes the organization of this section.

Table 6–14: Organization of this section (Communication timeout (in-process HTTP server))

Category	Title	Reference
Description	Overview of the Communication Timeout	*6.9.1*
Settings	Execution environment settings (J2EE server settings)	*6.9.2*

Note:

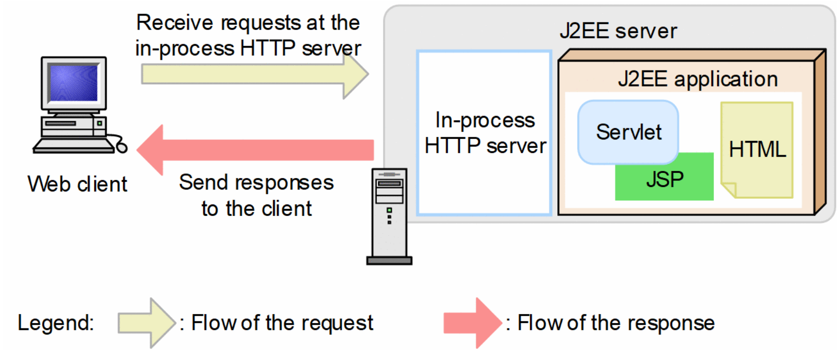
There is no specific description of *Implementation*, *Operations*, and *Notes* for this functionality.

6.9.1 Overview of the communication timeout

When using the in-process HTTP server, you can set communication timeout for receiving a request and sending a response between the Web client and the in-process HTTP server. When the response is awaited due to the network and application failure, you can detect the occurrence of failure from the occurrence of timeout, if the communication timeout is set.

When you use the in-process HTTP server, set the timeout for the communication indicated by the two arrows in the following figure.

Figure 6–7: Communication for which timeout can be set (When using the in-process HTTP server)



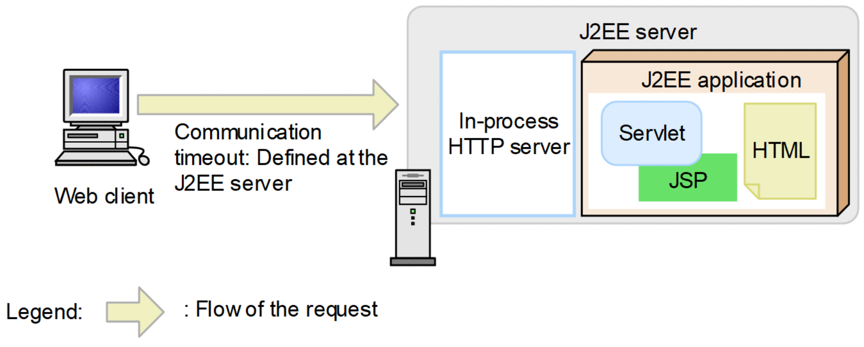
As shown in the figure, communication timeout is set for receiving requests and sending responses. The setting of communication timeout is explained separately for receiving of a request and sending of a response.

Note that when data timeout occurs while receiving a request from the Web client and sending a response to the Web client, a Web client or network failure is assumed and the connection with the Web client is disconnected, so a response is not returned.

(1) Communication timeout when receiving a request

The following figure shows the locations to set the communication timeout for receiving requests.

Figure 6–8: Locations to set the communication timeout for receiving requests (When using the in-process HTTP server)



When using the in-process HTTP server, you set a timeout for the communication between the Web client and the in-process HTTP server.

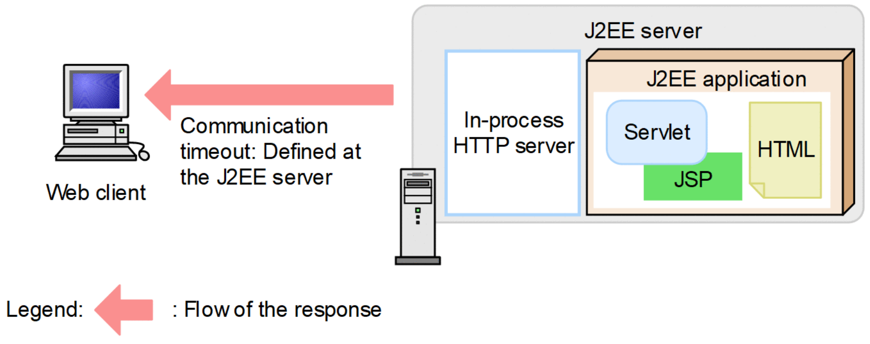
By setting a timeout for the communication between the Web client and the in-process HTTP server, you can detect the following failures in the client:

- The host on which the Web client is running is down.
- A network failure occurred between the Web client and the in-process HTTP server
- A failure occurs in client application.

(2) Communication timeout when sending a response

The following figure shows the locations to set the communication timeout for sending responses.

Figure 6–9: Locations to set the communication timeout for sending responses (When using the in-process HTTP server)



When using the in-process HTTP server, you set a timeout for the communication between the in-process HTTP server and the Web client.

By setting a timeout for the communication between the in-process HTTP server and the Web client, you can detect the following failures:

- The host on which the Web client is running is down.
- A network failure occurred between the Web client and the in-process HTTP server.
- A failure occurs in client application.

6.9.2 Execution environment settings (J2EE server settings)

To set up a communication timeout for the in-process HTTP server, you must set up the J2EE server.

This section describes the settings and examples of communication timeout in the in-process HTTP server.

You set up communication timeout when receiving requests and when sending responses. The setting of communication timeout is described separately for receiving of a request and sending of a response.

(1) Communication timeout settings for receiving requests

You set the communication timeout for receiving requests between the client and the in-process HTTP server.

Specify the communication timeout for receiving requests in the in-process HTTP server with the following parameters within the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file:

`webserver.connector.inprocess_http.receive_timeout`
: Specifies the waiting time for the process of receiving a request from the client.

For details on the Easy Setup definition file and parameters to specify, see *Part 3 Reference (V9 Compatibility Mode)*.

(2) Communication timeout settings for sending responses

You set the communication timeout for sending responses between the in-process HTTP server and the client.

Specify the communication timeout for sending responses in the in-process HTTP server with the following parameters within the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file:

`webserver.connector.inprocess_http.send_timeout`
: Specifies the waiting time for the process of sending a response to the client.

For details on the Easy Setup definition file and parameters to specify, see *Part 3 Reference (V9 Compatibility Mode)*.

(3) Example settings

An example of settings for communication timeout in the in-process HTTP server is as follows:

```
...
<param>
  <param-name>webserver.connector.inprocess_http.receive_timeout</param-nam
e>
  <param-value>300</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.send_timeout</param-name>
  <param-value>600</param-value>
</param>
...
```

In this example, `300` seconds is specified as the request receiving timeout and `600` seconds as the response sending timeout.

6.10 Specifying the IP address (In-process HTTP server)

This section describes the controlling of communication with the Web client by specifying the IP address in the in-process HTTP server.

The following table describes the organization of this section.

Table 6–15: Organization of this section (Specifying the IP address (in-process HTTP server))

Category	Title	Reference
Description	Bind address specification functionality	*6.10.1*
Settings	Execution environment settings (J2EE server settings)	*6.10.2*
Notes	Precautions related to IP address specification in the in-process HTTP server	*6.10.3*

Note:
There is no specific description of *Implementation* and *Operations* for this functionality.

Reference note

The functionality is not different when in-process HTTP server is not used (when the Web server integration functionality is used). For details on the functionality, see *5.7 Specifying the IP address (Web server integration)*.

6.10.1 Bind address specification functionality

In the Web container, you can explicitly specify the IP address to be used in the in-process HTTP server. This functionality is called the *Bind address specification functionality*. By using the bind address specification functionality, you can specify the setting so that only a single specific IP address is used for a host having multiple physical network interfaces or single physical network interface, when executing with a host.

6.10.2 Execution environment settings (J2EE server settings)

To specify the IP address of the in-process HTTP server, you must set the J2EE server.

This section describes the settings for the IP address of the in-process HTTP server.

Specify the IP address of the in-process HTTP server in the following parameters in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file:

`webserver.connector.inprocess_http.bind_host`
Specifies the host name or IP address to be used in the in-process HTTP server.

For details on the Easy Setup definition file and the parameters to be specified, see *4.3 Easy Setup definition file* in the *uCosminexus Application Server Definition Reference Guide*.

6.10.3 Precautions related to IP address specification in the in-process HTTP server

The precautions related to IP address specification in the in-process HTTP server are as follows:

- When the host name or the IP address is set, only requests for connecting to the specified IP address can be received. Instead of setting the IP address, a connection to any IP address on that host can be received, by specifying the wild card address. By default, the setting is specified to use the wild card address. When using the wild card address, note the following points:
  - If the specified host name cannot be resolved in the hosts file or DNS, start the server by using the wild card address.
  - If the specified host name or IP address is a remote host, start the server by using the wild card address.

6.11 Controlling access by limiting the hosts that are allowed access

To prevent unauthorized access of the J2EE server, you can control the hosts that can access the J2EE server. By default, access from all the hosts is allowed. By specifying the host name or the IP address of the host that is allowed access beforehand, you can allow access only from a specific host and prevent unauthorized access.

This section describes the controlling of access by limiting the hosts that are allowed access to the J2EE server.

Table 6–16: Organization of this section (Controlling access by limiting the hosts that are allowed access)

Category	Title	Reference
Description	Limiting the hosts that are allowed access	*6.11.1*
Settings	Execution environment settings (J2EE server settings)	*6.11.2*

Note:
There is no specific description of *Implementation*, *Operations*, and *Notes* for this functionality.

6.11.1 Limiting the hosts that are allowed access

To limit the hosts, set the host name or IP address of the hosts that are allowed access. In such a case, if you specify an asterisk (*) in place of the host name and the IP address, access from all hosts is allowed. When the hosts that are allowed access are specified by the host name, the name of the host is resolved on starting the J2EE server. Note that even if the local host is not explicitly specified, the access is always allowed. Normally, for the systems that are accessed from external networks, you specify the IP address of the proxy server.

The precautions when the host that is allowed access is specified in the host name are as follows:

Notes

- You need to specify the host name resolvable in the hosts file or DNS. If the host name cannot be resolved, the server is started by the default settings.
- The host name is resolved when the J2EE server is started, and hence, longer time is taken for starting the server. The changed IP address may not be applied after starting the server.

6.11.2 Execution environment settings (J2EE server settings)

To specify settings for limiting the hosts that are given the access, you must set up the J2EE server.

This section describes the settings and examples for limiting the hosts that are given the access.

(1) How to set

Specify the settings for limiting the hosts that are given the access in the following parameter within the `<configuration>` tag of the logical J2EE server (j2ee-server), in the Easy Setup definition file.

`webserver.connector.inprocess_http.permitted.hosts`
Specifies the host name or IP address of the host that is given the access.

For details on the Easy Setup definition file and the parameters to be specified, see *4.3 Easy Setup definition file* in the *uCosminexus Application Server Definition Reference Guide*.

Important note

The precautions when the host that is given the access is specified in the host name are as follows:

- You need to specify the host name resolvable in the hosts file or DNS. If the host name is not resolvable, the J2EE server is started with the default settings (access is given for all the hosts).
- The host name is resolved when the J2EE server is started, and hence, longer time is taken for starting the J2EE server. The changed IP address may not be applied after starting the server.

(2) Example settings

The following is the setting example for limiting the hosts that are given the access:

```
...
<param>
  <param-name>webserver.connector.inprocess_http.permitted.hosts</param-nam
e>
  <param-value>host1,host2</param-value>
</param>
...
```

In this example, access is given only for `host1` and `host2` and access from other hosts is not allowed.

6.12 Controlling access by limiting the request data size

In the in-process HTTP server, by receiving only the request data that is less than a constant size, you can reject the receipt of invalid request data, control the load on the server, and maintain stable operations.

This section describes the controlling of access by limiting the request data size.

The following table describes the organization of this section.

Table 6–17: Organization of this section (Controlling access by limiting the request data size)

Category	Title	Reference
Description	Limiting the request data size	*6.12.1*
Settings	Execution environment settings (J2EE server settings)	*6.12.2*

Note:

There is no specific description of *Implementation*, *Operations*, and *Notes* for this functionality.

6.12.1 Limiting the request data size

In the in-process HTTP server, by receiving only the request data that is less than a constant size, you can reject the receipt of invalid request data, control the load on the server, and maintain stable operations.

Set the following items to implement access control by limiting the request data size:

- **Limiting the length of the request line**

  Control the access by setting an upper limit on the length of the request line. The length of a request line includes the HTTP method, URI (including the query string), the HTTP version, and the linefeed (2 bytes) indicating the end of the request line.

  If the length of the received request line exceeds the upper limit, an error of status code 414 is returned to the Web client.

- **Limiting the number of HTTP headers**

  Control the access by setting the upper limit for the number of HTTP headers included in the HTTP request.

  If the number of HTTP headers included in the received HTTP request exceeds the upper limit, an error of status code 400 is returned to Web client.

- **Limiting the request header size**

  Control access by setting the upper limit for the request header size of the HTTP request.

  If the HTTP header size of the received HTTP request exceeds the upper limit, an error of status code 400 is returned to Web client.

- **Limiting the request body size**

  Control access by setting the upper limit for the body size of the HTTP request. In the in-process HTTP server, the body size of the HTTP request is determined by the value of the Content-Length header included in the request header.

  If the body size of the HTTP request exceeds the upper limit, an error of status code 413 is returned to Web client.

  When the request body is sent in a chunk format, the data up to the specified upper limit is read inside the servlet. If the data exceeds the upper limit, an exception (`IOException`) is thrown in the servlet, but the processing of the servlet continues. Based on the result of data read up to the specified upper limit in the client that sent the request, the response created by the application is returned.

Tip

If the gateway device such as SSL accelerator and load balancer exist or if the proxy server is deployed and the gateway equipment and proxy server have the functionality for controlling the request data size, you must set a value less than the value set in the control functionality.

6.12.2 Execution environment settings (J2EE server settings)

To specify the settings for limiting the request data size, you must set up a J2EE server.

This section describes the settings and examples for limiting the request data size:

(1) How to set

Implement the J2EE server settings in the Easy Setup definition file. Specify the definition for limiting the request data size in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file.

The following table lists the definitions in the Easy Setup definition file for limiting the request data size:

Table 6–18: Definitions in the Easy Setup definition file for limiting the request data size

Parameter to be specified	Setting contents
`webserver.connector.inprocess_http.limit.max_request_line`	Specifies the upper limit of the request line.
`webserver.connector.inprocess_http.limit.max_headers`	Specifies the upper limit for the number of HTTP headers included in the HTTP request.
`webserver.connector.inprocess_http.limit.max_request_header`	Specifies the upper limit for the request header size of the HTTP request.
`webserver.connector.inprocess_http.limit.max_request_body`	Specifies the upper limit for the body size of the HTTP request.

For details on the Easy Setup definition file and the parameters to be specified, see *4.3 Easy Setup definition file* in the *uCosminexus Application Server Definition Reference Guide*.

Tip

If the gateway device such as SSL accelerator and load balancer exist or if the proxy server is deployed and the gateway device and proxy server have the functionality for controlling the request data size, you must set a value less than the value set in the control functionality.

(2) Example settings

An example of settings for limiting the request data size is as follows:

```
...
<param>
  <param-name>webserver.connector.inprocess_http.limit.max_request_line</par
am-name>
  <param-value>1024</param-value>
```

```
</param>
<param>
  <param-name>webserver.connector.inprocess_http.limit.max_headers</param-na
me>
  <param-value>100</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.limit.max_request_header</p
aram-name>
  <param-value>8192</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.limit.max_request_body</par
am-name>
  <param-value>16384</param-value>
</param>
...
```

6.13 Controlling access by limiting the HTTP-enabled methods

This section describes the controlling of access by limiting the HTTP-enabled methods.

The following table describes the organization of this section:

Table 6–19: Organization of this section (Controlling access by limiting the HTTP-enabled methods)

Category	Title	Reference
Description	Limiting the HTTP-enabled methods	*6.13.1*
Settings	Execution environment settings (J2EE server settings)	*6.13.2*

Note:
There is no specific description of *Implementation*, *Operations*, and *Notes* for this functionality.

6.13.1 Limiting the HTTP-enabled methods

In the in-process HTTP server, you reject the receipt of requests containing HTTP-disabled methods by limiting the HTTP-enabled methods for the HTTP request. As a result, you can prevent the unauthorized access of the resources on the server. By default, you can use the `DELETE` method, `HEAD` method, `GET` method, `OPTIONS` method, `POST` method, and `PUT` method.

To limit the HTTP methods, specify the method names of the HTTP-enabled methods. The value defined in RFC2616 must be used for the value set as the HTTP-enabled method. However, an asterisk (`*`) cannot be used in the method name string. If an asterisk (`*`) is specified instead of the method name, all the methods can be used.

If a request containing an HTTP-disabled method is received, an error of status code 405 is returned to the Web client.

Note that if a request containing the `OPTIONS` method is sent for the static contents, a method excluding the disabled methods for the in-process HTTP server from the enabled methods (`GET` method, `POST` method, `TRACE` method, and `OPTIONS` method) is returned for the static contents in the `Allow` header included in the response by default. In the case of servlets and JSPs, limiting the HTTP-enabled methods depends on the implementation of the Web application.

6.13.2 Execution environment settings (J2EE server settings)

To specify settings for limiting the HTTP-enabled methods, you must set up the J2EE server.

This section describes the settings and examples for limiting the HTTP-enabled methods.

(1) How to set

Specify the settings for limiting the HTTP-enabled methods in the following parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file.

`webserver.connector.inprocess_http.enabled_methods`
  Specifies the method name of an HTTP-enabled method.

For details on the Easy Setup definition file and the parameters to be specified, see 4.*3 Easy Setup definition file* in the *uCosminexus Application Server Definition Reference Guide*.

Tip

If a request containing the `OPTIONS` method is sent for the static contents, a request excluding the disabled methods for the in-process HTTP server from the enabled methods (`GET` method, `POST` method, `TRACE` method, and `OPTIONS` method) is returned for the static contents by default. In the case of servlets and JSPs, limiting the HTTP-enabled methods depends on the implementation of the Web application.

(2) Example settings

The following is the setting example for limiting the HTTP-enabled methods. Note that the following example shows the default settings:

```
...
<param>
  <param-name>webserver.connector.inprocess_http.enabled_methods</param-nam
e>
  <param-value>GET,HEAD,POST,PUT,DELETE,OPTIONS</param-value>
</param>
...
```

In this example, access is allowed for the `GET` method, `HEAD` method, `POST` method, `PUT` method, `DELETE` method, and `OPTIONS` method and access is rejected for the `TRACE` method.

6.14 Customizing responses to the Web client using HTTP responses

This section describes the customizing of responses to the Web client using HTTP responses.

The following table describes the organization of this section:

Table 6–20: Organization of this section (Customizing responses to the Web client using HTTP responses)

Category	Title	Reference
Description	Customizing the HTTP response header	*6.14.1*
Settings	Execution environment settings (J2EE server settings)	*6.14.2*

Note:
There is no specific description of *Implementation*, *Operations*, and *Notes* for this functionality.

6.14.1 Customizing the HTTP response header

This section describes the customizing of the HTTP response header.

In the in-process HTTP server, you can customize the information that is automatically set up in the Server header of the HTTP response. By default, `CosminexusComponentContainer` is automatically setup.

The value defined in RFC2616 must be used for the value that is automatically set up in the Server header. If the use of Server header is specified in the servlets and JSPs, that setting is given priority.

6.14.2 Execution environment settings (J2EE server settings)

To specify settings for customizing the HTTP response header, you must set up the J2EE server.

This section describes the settings and examples for customizing the HTTP response header.

(1) How to set

Specify the settings for customizing the HTTP response header in the following parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file:

`webserver.connector.inprocess_http.response.header.server`
Specifies the string that is automatically set up in the Server header of the HTTP response.

For details on the Easy Setup definition file and the parameters to be specified, see *4.3 Easy Setup definition file* in the *uCosminexus Application Server Definition Reference Guide*.

(2) Example settings

The following is the setting example for customizing the HTTP response header:

```
...
<param>
  <param-name>webserver.connector.inprocess_http.response.header.server</par
```

```
am-name>
  <param-value>GyoumuServer/1.0</param-value>
</param>
...
```

In this example, `GyoumuServer/1.0` is specified as the Server header value.

6.15 Error page customization (In-process HTTP server)

If the client access non-existent resources the in-process HTTP server returns the error status code and error page, and the error page generated by the in-process HTTP server is displayed to the client. In the in-process HTTP server, a user-created page can be displayed to the client instead of this error page. This is called *error page customization*.

This section describes the customization of the responses to the Web client with the error page.

The following table describes the organization of this section:

Table 6–21: Organization of this section (Error page customization (in-process HTTP server))

Category	Title	Reference
Description	Error page that can be customized	*6.15.1*
Implementation	Implementation required for customizing the error page	*6.15.2*
Settings	Execution environment settings (J2EE server settings)	*6.15.3*
Notes	Precautions related to error page customization	*6.15.4*

Note:

There is no specific description of *Operations* for this functionality.

6.15.1 Error page that can be customized

When an error such as the access to non-existent resources occurs, a user-created error page can be displayed to the client instead of the error page displaying the error status code.

By using the error page customization with the in-process HTTP server, you can control error page customization corresponding to a specific status code and error page customization corresponding to a request URL in the Web Container at the same time. You can also customize the error page even when you cannot customize the error page with the Web applications in the following cases:

- When the context corresponding to the request does not exist (status code 404)
- When an attempt is made to process the request with a context that is in the process of stopping (status code 503)
- When the in-process HTTP server returns an error status code

For details on the error status codes returned by the in-process HTTP server, see *Appendix C.3 Error status codes returned by the in-process HTTP server*.

In the in-process HTTP server, you can customize the following error pages:

- **Error page customization corresponding to the status codes**
  You can customize the error pages corresponding to the status codes 400 and 500.
  By customizing the error pages corresponding to the status code, you can send the files corresponding to the status code and execute redirection corresponding to the status code.
  - Sending the files corresponding to the status code
    You can return to the client a specific file as a response body for the customized status code. In this case, specify the Content-Type header value of the response.
    Note that if the reading of the file fails during request processing, the default error page is used.
  - Redirection corresponding to the status code

You can redirect to a specific URL for the customized status code. In the case of redirection, specify 302 as the response status code and the redirect URL in the Location header.

When sending a file corresponding to the status code, redirection cannot be executed.

- **Error page customization corresponding to the request URLs**

  You can specify a request URL and customize the error page for a specific URL. When the request URL is specified, the customized error page is returned to the client only when an error occurs in the request processing matching with the specified URL.

6.15.2 Implementation required for customizing the error page

To customize the error page with the in-process HTTP server, use the `sendError` method of the `javax.servlet.http.HttpServletResponse` interface, and set the response status code. Note that if you use the `setStatus` method (such as when the `setStatus` method is used in JSP), customization might not be executed by the in-process HTTP server. However, even if you use the `sendError` method, if the Web application fulfills one of the following conditions, the error page is not customized by the in-process HTTP server:

- If an exception is thrown during the execution of the `sendError` method
- When error page customization is specified in the Web application and when an error occurs, the execution of the error page as per the settings terminates normally#

#

Normal termination of error page execution implies the satisfaction of the following conditions:

- An exception that cannot be caught in the error page does not occur.
- The status code ends with a value other than 400 to 599.

6.15.3 Execution environment settings (J2EE server settings)

To customize the error pages, you must set up the J2EE server.

This section describes the settings and examples for error page customization.

(1) How to set

Implement the J2EE server settings in the Easy Setup definition file. Specify the definition for error page customization in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file.

The following table lists the definitions in the Easy Setup definition file for the error page customization:

Table 6–22: Definitions in the Easy Setup definition file for error page customization

Parameter to be specified	Setting contents
`webserver.connector.inprocess_http.error_custom.list`	Specifies the definition name for error page customization.
`webserver.connector.inprocess_http.error_custom.`*error-page-customising-definition-name*`.status`	Specifies the error status code that customizes the error page to customize the error page in association with the error status code.

Parameter to be specified	Setting contents
`webserver.connector.inprocess_http.error_custom.`*error-page-customising-definition-name*`.file`	Specifies the file to be returned to the client as a response body to send a file corresponding to the error status code.
`webserver.connector.inprocess_http.error_custom.`*error-page-customising-definition-name*`.file.content_type`	Specifies the value of the Content-Type header when a file specified in the `webserver.connector.inprocess_http.error_custom.`*error-page-customizing-definition-name*`.file` parameter is sent to the client as a response body.
`webserver.connector.inprocess_http.error_custom.`*error-page-customising-definition-name*`.redirect_url`	Specifies the redirection destination URL when redirecting in compliance with the error status code.
`webserver.connector.inprocess_http.error_custom.`*error-page-customising-definition-name*`.request_url`	Specifies the request URL that applies error page customization when customizing the error page in association with the request URL.

For details on the Easy Setup definition file and the parameters to be specified, see *4.3 Easy Setup definition file* in the *uCosminexus Application Server Definition Reference Guide*.

(2) Example of settings

The following is the setting example of settings for the error page customization:

```
...
<param>
  <param-name>webserver.connector.inprocess_http.error_custom.list</param-na
me>
  <param-value>ERR_CUSTOM_1,ERR_CUSTOM_2</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.error_custom.ERR_CUSTOM_1.s
tatus</param-name>
  <param-value>404</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.error_custom.ERR_CUSTOM_1.f
ile</param-name>
  <param-value>C:/data/404.html</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.error_custom.ERR_CUSTOM_1.f
ile.content_type</param-name>
  <param-value>text/html; charset=ISO-8859-1</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.error_custom.ERR_CUSTOM_2.s
tatus</param-name>
  <param-value>503</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.error_custom.ERR_CUSTOM_2.r
edirect_url</param-name>
  <param-value>http://host1/503.html</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.error_custom.ERR_CUSTOM_2.r
```

```
equest_url</param-name>
  <param-value>/dir1/*</param-value>
</param>
...
```

In this example, `ERR_CUSTOM_1` and `ERR_CUSTOM_2` are used as the error page customizing definition names. In `ERR_CUSTOM_1`, when the response status code is '404', `C:/data/404.html` is returned to the client. `text/html; charset=ISO-8859-1` is used as the Content-Type header value. In `ERR_CUSTOM_2`, when the request is a URL beginning with `/dir1/` and the response status code is '503', the request is redirected to `http://host1/503.html`.

6.15.4 Precautions related to error page customization

The precautions related to error page customization by the in-process HTTP server are as follows:

- When the HTTP response is an HTML file, if another file (such as an image file) is being referenced from that HTML, the error page might not be properly displayed.
- Note that depending on the browser settings, if the status code indicates an error and if the size of the response body is small, the response body MIGHT be replaced by a browser-specific message. Note that in the default error page displayed when an error page is not specifically customized, the size of the response body is small.
- If the value specified in the redirect URL matches with the value specified for the request URL and if the same error status occurs after redirection, note that the client keeps redirecting the requests.
- When redirect corresponding to the status code is implemented, the query string added to the request URL when an error occurs, is not added to the redirect URL. Also, when a session is managed by URL rewriting, the session cannot be inherited even if the request is redirected to the same Web application as the error page.

6.16 Notification of gateway information to a Web container

This section describes the reporting of the gateway information to the Web container.

The following table describes the organization of this section:

Table 6–23: Organization of this section (Reporting of the gateway information to the Web container)

Category	Title	Reference
Description	Gateway specification functionality	*6.16.1*
Settings	Execution environment settings (J2EE server settings)	*6.16.2*
Notes	Precautions related to reporting the gateway information to the Web Container	*6.16.3*

Note:

There is no specific description of *Implementation* and *Operations* for this functionality.

Reference note

The functionality is not different when in-process HTTP server is not used (when the Web server integration functionality is used). For details on the functionality, see *5.10 Notification of gateway information to a Web container*.

6.16.1 Gateway specification functionality

If a gateway such as an SSL accelerator or a load balancer is placed between a client and an in-process HTTP server, when the Web container automatically redirects to a welcome file or the Form authentication window, the Web container may not properly create a forwarding URL because the container cannot acquire the information about the gateway.

To avoid this problem, you can use the *gateway specification functionality*. This functionality notifies a Web container of gateway information so that the Web container can properly redirect to a welcome file or Form authentication window.

The gateway specification functionality is used in the following case:

- **When an SSL accelerator is placed between a client and in-process HTTP server:**

  Even if a client accesses an SSL accelerator via HTTPS, the SSL accelerator accesses a Web server via HTTP, which causes the Web container to assume that the access uses HTTP. For this reason, HTTP is used for the URL scheme for the welcome file or Form authentication window that is the redirection destination.

  In this situation, by using the gateway specification function to specify that the scheme be always considered as HTTPS, you can ensure that accesses are properly redirected.

- **When a request without a Host header needs to be redirected away from the in-process HTTP server that received the request**

  When redirecting a request without a Host header, the host name and port number of the redirection destination URL becomes the host name and port number of the Web server that receives the request.

  Use the gateway specification functionality when the host name and port number of the URL accessed by the client is different from the Web server or in-process HTTP server that receives the request, such as when a load balancer is deployed before the Web server or in-process HTTP server. As a result, the host name and port number accessed from the client are specified, so the request can be redirected properly.

Note that when using the in-process HTTP server, gateway specification functionality cannot be used if multiple different routes are used for accessing one Web container (when HTTP requests are forwarded to the Web container from multiple gateways). To use the gateway specification functionality in the in-process HTTP server, use a configuration in which there is one access route to the Web container.

6.16.2 Execution environment settings (J2EE server settings)

To specify settings for reporting the gateway information to the Web Container, you must set up a J2EE server.

This section describes the settings and examples for reporting the gateway information to the Web container.

(1) How to set

Implement the J2EE server settings in the Easy Setup definition file. Define the settings for reporting the gateway information to the Web container in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file.

The following table lists the definitions in the Easy Setup definition file for reporting the gateway information to the Web container:

Table 6–24: Definitions in the Easy Setup definition file for reporting the gateway information to the Web container

Parameter to be specified	Setting contents
`webserver.connector.inprocess_http.gateway.https_scheme`	Specifies the scheme of the redirection destination URL.
`webserver.connector.inprocess_http.gateway.host`	Specifies the gateway host name.
`webserver.connector.inprocess_http.gateway.port`	Specifies the gateway port number.

For details on the Easy Setup definition file and the parameters to be specified, see *4.3 Easy Setup definition file* in the *uCosminexus Application Server Definition Reference Guide*.

(2) Example settings

An example of settings for the gateway specification functionality is as follows:

```
...
<param>
  <param-name>webserver.connector.inprocess_http.gateway.host</param-name>
  <param-value>host1</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.gateway.port</param-name>
  <param-value>4443</param-value>
</param>
<param>
  <param-name>webserver.connector.inprocess_http.gateway.https_scheme</param
-name>
  <param-value>true</param-value>
```

```
</param>
...
```

In this example, the gateway specification functionality is used to specify settings so that the scheme is always considered to be HTTPS.

6.16.3 Precautions related to reporting the gateway information to the Web container

The following are cautionary notes on using the gateway specification functionality:

Specifying the host name and port number of an URL where an access is redirected:

A browser usually sends a request with the Host header appended, so it is not necessary to specify the host name or port number for an URL where access is to be redirected.

Note that you can check whether or not the request has the Host header by calling the `getHeader` method of the `javax.servlet.http.HttpServletRequest` interface, with the Host argument specified.

Servlet API behavior:

Using the gateway specification functionality causes some servlet API functions to behave differently. Take care when using API functions with a Web application.

The following describes the precautions on servlet APIs when using the gateway specification functionality for each method to be used:

- The `sendRedirect` method of the `javax.servlet.http.HttpServletResponse` class

  When you specify a relative URL for the argument, and if the request does not have the Host header, the host name and port number of the URL of the redirection destination are the values specified by the gateway specification functionality. When you specify a relative URL for the argument and use the gateway specification functionality to specify that the scheme is to be considered as `https`, the scheme of the URL of the redirection destination is always `https`.

- The `getRequestURL` method of the `javax.servlet.ServletRequest` interface

  When you use the gateway specification functionality to specify that the scheme is to be considered as `https`, the return value is always a URL starting with `https://`.

- The `getServerName` method of the `javax.servlet.ServletRequest` interface

  When you use the gateway specification functionality to specify the host name of the URL of the redirection destination, and if the request does not have the Host header, the return value is the value you specified.

- The `getServerPort` method of the `javax.servlet.ServletRequest` interface

  When you use the gateway specification functionality to specify the port number of the URL of the redirection destination, and if the request does not have the Host header, the return value is the value you specified.

  When you use the gateway specification functionality to specify the host name of the URL of the redirection destination, and if the port number is omitted, the return value is `80` when the request scheme is `http`, and `443` when the request scheme is `https`.

- The `getScheme` method of the `javax.servlet.ServletRequest` interface

  When you use the gateway specification functionality to specify that the scheme is to be considered as `https`, the return value is always `https`.

- The `isSecure` method of the `javax.servlet.ServletRequest` interface

  When you use the gateway specification functionality to specify that the scheme is to be considered as `https`, the return value is always `true`.

- The `getAttribute` method of the `javax.servlet.ServletRequest` interface

  The following attributes cannot be obtained even when you used the gateway specification functionality to specify that a scheme is to be considered as `https`:

  - `javax.servlet.request.cipher_suite` (When Microsoft IIS is used for the Web server, this attribute cannot be obtained regardless of whether the gateway specification functionality is used.)

  - `javax.servlet.request.key_size`

  - `javax.servlet.request.X509Certificate`

The `<transport-guarantee>` tag in `web.xml`:

When you use the gateway specification functionality to specify that a scheme is to be considered as HTTPS, a request to a Web server will be considered to use HTTPS even if the request actually uses HTTP. Note that this prevents an access from being redirected to an URL that uses HTTPS, even if you specify `INTEGRAL` or `CONFIDENTIAL` in the `<transport-guarantee>` tag in `web.xml`.

The Secure attribute for cookies:

When you use the gateway specification functionality to specify that a scheme is to be considered as HTTPS, when a session ID generated by a Web container is returned to the client by the session cookie, the Secure attribute is appended to the cookie.

6.17 Output of log and trace

This section describes the log and trace output by the in-process HTTP server.

The following table describes the organization of this section:

Table 6–25: Organization of this section (Output of log and trace)

Category	Title	Reference
Description	Log and trace output by the in-process HTTP server	*6.17.1*
Settings	Customizing the access log of the in-process HTTP server	*6.17.2*

Note:

There is no specific description of *Implementation*, *Operations*, and *Notes* for this functionality.

6.17.1 Log and trace output by the in-process HTTP server

The in-process HTTP server outputs the log and trace described in the following table to support application development, for performance analysis during operations, and for troubleshooting during a failure:

Table 6–26: Log and trace output by the in-process HTTP server

Types of log and trace	Description
Access log	Outputs the result of request and response processing from the Web client. This log is used for analyzing the communication with the Web client. By analyzing the access log, you can analyze the files requested from the Web client and the performance information and session tracking information of the in-process HTTP server.
Performance analysis trace	Outputs the performance analysis information of the sending and receiving of requests and the information for troubleshooting in the case of failure in the in-process HTTP server. The performance analysis trace is converted to a CSV format and can be used to analyze the bottlenecks in the entire system in combination with the performance analysis information output by the functionality of other J2EE servers. For details on the performance analysis trace, see *15.1 Overview of performance analysis traces*.
Thread trace	Trace for maintenance.
Communication trace	Trace for maintenance.

6.17.2 Customizing the access log of the in-process HTTP server

The in-process HTTP server outputs the access log, performance analysis trace, thread trace, and communication trace for supporting application development, for performance analysis during operations, and for troubleshooting during failure. You can change the number and size of these files. You can also customize the log output format in the access log.

This section describes the customization of the access log output format in the in-process HTTP server. For details on changing the number and size of the access log and trace files in the in-process HTTP server, see *16.2 Settings for acquiring the in-process HTTP server log*.

(1) Customization procedure

To customize the access log output format:

1. Define the format name of the access log.

   To create a new format, add the new format name in the `webserver.logger.access_log.format_list` parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file.

   Example settings

   ```
   ...
   <param>
     <param-name>webserver.logger.access_log.format_list</param-name>
     <param-value>formatA</param-value>
   </param>
   ...
   ```

   For creating a new format, reference the access log format provided by default. For details on the format, see *(2) Access log format*.

2. Define the output format for the access log.

   Define the access log output format in the format specified in *format-name* with the `webserver.logger.access_log.`*format-name* parameter within the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file. Specify the argument for the access log format in the output format definition.

   Example settings

   ```
   ...
   <param>
     <param-name>webserver.logger.access_log.formatA</param-name>
     <param-value>%h %u %t &quot;%r&quot; %&gt;s HostHeader=&quot;%{host}i
   &quot;</param-value>
   </param>
   ...
   ```

   For details on the specifiable format arguments, see *(3) Arguments of the access log format*.

3. Specify the format that will be used to output the access log.

   Specify the format name that will be used to output the access log with the `webserver.logger.access_log.inprocess_http.usage_format` parameter within the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file. An access log is output in the format specified here:

   Example settings

   ```
   ...
   <param>
     <param-name>webserver.logger.access_log.inprocess_http.usage_format</
   param-name>
     <param-value>formatA</param-value>
   </param>
   ...
   ```

(2) Access log format

Application Server provides two types of formats namely *common* (default format) and *combined* (extended format) as the access log formats of the in-process HTTP server. When creating a new format, reference these formats.

(a) Output format

The output format of access log is described below. Note that Δ indicates one-byte space. Also, for the convenience of expression, the log is output across multiple lines, but the log is actually output in one line.

The output format of a default format is as follows:

```
Host-name-or-IP-address-of-Web-clientΔRemote-log-nameΔAuthentication-user-na
me
ΔStart-time-of-Web-client-request-processingΔRequest-line
ΔFinal-status-codeΔNumber-of-bytes-sent-excluding-the-HTTP-header
```

The output format of an extended format is described below:

```
Host-name-or-IP-address-of-Web-clientΔRemote-log-nameΔAuthentication-user-na
me
ΔStart-time-of-Web-client-request-processingΔRequest-line
ΔFinal-status-codeΔNumber-of-bytes-sent-excluding-the-HTTP-header
Δ"Referer-header-contents"Δ"User-Agent-header-contents"
```

The underlined part is the difference between the default format and extended format. In the extended format, *Referer header contents* and *User-Agent header contents* is output in addition to the output contents of the default format.

(b) Example of output

An example of access log output in the default format is as follows:

```
10.20.30.40 - user [20/Dec/2004:15:45:01 +0900] "GET /index.html HTTP/1.1" 2
00 8358
10.20.30.40 - user [20/Dec/2004:15:45:01 +0900] "GET /left.html HTTP/1.1" 20
0 2358
10.20.30.40 - user [20/Dec/2004:15:45:01 +0900] "GET /right.html HTTP/1.1" 2
00 4358
```

An example of access log output in the extended format is as follows:

```
10.20.30.40 - - [18/Jan/2005:13:06:10 +0900] "GET / HTTP/1.0" 200 38 "-" "Mo
zilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
10.20.30.40 - - [18/Jan/2005:13:06:25 +0900] "GET /demo/ HTTP/1.0" 500 684 "
-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
```

(3) Arguments of the access log format

The following table lists the arguments of the access log format that are specified when you define the output format of the format:

Table 6–27: List of arguments of the access log format

Format arguments	Output contents	Example of output
`%%`	% Sign	`%`
`%a`	IP address of the Web client	`10.20.30.40`

Format arguments	Output contents	Example of output
`%A`	IP address of the J2EE server	`10.20.30.100`
`%b`	Number of bytes sent excluding the HTTP header ('-' in the case of 0 bytes)	`2048`
`%B`	Number of bytes sent excluding the HTTP header ('`0`' in the case of 0 bytes)	`1024`
`%h`	Host name or IP address of the Web client (IP address when the host name cannot be acquired)	`10.20.30.40`
`%H`	Request protocol	`HTTP/1.1`
`%l`	Remote log name#1 (Always '-')	`-`
`%m`	Request method	`GET`
`%p`	Port number that receives the request from the Web client	`80`
`%q`	Query string (Begins with '`?`'. If the query string does not exist, null character)	`?id=100&page=15`
`%r`	Request line	`GET /index.html HTTP/1.1`
`%>s`	Final status code (Internally redirected value is not output)	`200`
`%S`#2	User's session ID ('-' if the session ID does not exist)	`00455AFE4DA4E7B7789F247B8F E5D605`
`%t`	Start time of the Web client request processing (Unit: seconds, output format: *dd*/*MMM*/*YYYY*:*HH*:*mm*:*ss* *Z*)	`[18/Jan/ 2005:13:06:10 +0900]`
`%T`	Time required for processing the Web client request (Unit: seconds)	`2`
`%d`	Start time of the Web client request processing (Unit: milliseconds, output format: *dd*/*MMM*/*YYYY*:*HH*:*mm*:*ss*.*nnn* *Z* (*nnn* indicates milliseconds))	`[18/Jan/ 2005:13:06:10.152 +0900]`
`%D`	Time required for processing the Web client request (Unit: milliseconds)	`2000`
`%u`	Basic authentication user name, Form authentication user name ('-' when the authentication user name does not exist)	user
`%U`	Request file path	`/index.html`
`%v`	Local host name of the J2EE server	server
`%{foo}I`#3	Contents of request header `foo` ('-' when foo header does not exist)	In the case of `%{Host}i`, `www.example.com:8888`
`%{foo}c`	Of the Cookie information sent by the Web client, contents of Cookie name `foo` ('-' when the Cookie name does not include `foo`)	In the case of `%{JSESSIONID}c`, `00455AFE4DA4E7B7789F247B8F E5D605`
`%{foo}o`#3	Contents of response header `foo` ('-' when the foo header does not exist)	In the case of `%{Server}o`, `CosminexusComponentContain er`

#1

The remote log name is the user name in the Web client that can be acquired with the Identification protocol defined in RFC 1413.

#2

The session ID displayed in `%S` is the value of the Cookie name 'JSESSIONID'.

#3

The same header name might be sent multiple times in one HTTP request or HTTP response. In this case, the contents of the header read first will be output.

Important note

If there is an error in the specification of the format argument, the default format is used. The example of the use of the default format is as follows:

- When strings not existing in the list of format arguments (example: `%G`) is specified
- When `0` characters (such as `%{}i`) are specified in the request header contents, response header contents, and Cookie name.

Reference note

If the default format and extended format are coded in the format arguments, the format is as follows:

- Default format
  `%h %l %u %t "%r" %>s %b`
- Extended format
  `%h %l %u %t "%r" %>s %b "%{Referer}i" "%{User-Agent}i"`

6.18 URI decode functionality

You can use the URI decode functionality in the Web server integration and the in-process HTTP server.

This section describes the URI decode functionality.

The following table describes the organization of this section:

Table 6–28: Organization of this section (URI decode functionality)

Category	Title	Reference
Description	Overview of URI decode functionality	*6.18.1*
Settings	Execution environment settings (J2EE server settings)	*6.18.2*
Notes	Precautions for using the URI decode functionality	*6.18.3*

Note:

There is no specific description of *Implementation* and *Operations* for this functionality.

6.18.1 Overview of URI decode functionality

The URI decode functionality is used for decoding the URL-encoded strings included in the servlet path of request URIs and in the additional path information of Application Server. However, the context path is not decoded.

To execute a Web application that does not use decoded URIs, you must not use the URI decode functionality or you must manage at the Web application machine.

The following is the description of "Servlet APIs affected when URI decode functionality is used", "Functionality using decoded strings", "Character code used for decoding", and "Execution procedure for decoding and normalizing character strings":

(1) Servlet APIs affected when using the URI decode functionality

For using the URI decode functionality, a decoded URI is considered as a return value in the following methods of the `javax.servlet.http.HttpServletRequest` interface:

- `getPathInfo` method
- `getPathTranslated` method
- `getServletPath` method

However, in the `getRequestURI` and `getRequestURL` methods, a non-decoded URL is considered as a return value.

(2) Functionality using decoded strings

For using the URI decode functionality, the decoded strings are used in the following processes:

- Matching with URL pattern of servlets and JSPs
- Matching with default mapping
- Matching with static contents

- Matching with URL pattern of filter
- Matching with the `<error-page>` tag of `web.xml` or with the `errPage` attribute of the page directive of JSPs
- Matching with URL pattern for restricting access
- Determining URL for login authentication
- Forward and include request
- Matching with URL pattern for HTTP response compression filter
- Matching with URL pattern to control the number of concurrently executed threads in the URL group

However, the context path is not decoded and is handled as the original string, so the value "404 Not Found" is considered as a return value, when the context path does not match with the context root.

The matching for the decoded character string is not performed in the following functionality of Application Server:

- Error page customization functionality of the in-process HTTP server
- Request distribution functionality by redirecting the in-process HTTP server

(3) Character code used for decoding

For using the URI decode functionality, the character code used for decoding is UTF-8.

(4) Execution procedure for decoding and normalizing character strings

URLs used in the matching processes after decoding are normalized in the request URIs sent from clients.

6.18.2 Execution environment settings (J2EE server settings)

To use the URI decode functionality, you must set up a J2EE server.

Implement the J2EE server settings in the Easy Setup definition file. Specify the definition for the URI decode functionality in `webserver.http.request.uri_decode.enabled` within the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file. With this parameter, specify whether to use the URI decode functionality or not.

For details on the Easy Setup definition file and the parameters to be specified, see *4.3 Easy Setup definition file* in the *uCosminexus Application Server Definition Reference Guide*.

6.18.3 Precautions for using the URI decode functionality

This section describes the precautions for using the URI decode functionality.

(1) Execution procedure for decoding and normalizing character strings

The URIs, normalized after decoding, are used for matching the URL patterns with the servlet path.

The URIs, normalized without decoding, are used for matching the context root with the context path.

(2) Attributes of request

The decoded values are also stored in the attributes that are added in requests during the forward or include processing. The following table describes whether the stored values are decoded for each attribute specified in requests during the forward or include processing:

Processing	Attribute	Decoding the stored value
Forward	`javax.servlet.forward.request_uri`	N
	`javax.servlet.forward.context_path`	N
	`javax.servlet.forward.servlet_path`	Y
	`javax.servlet.forward.path_info`	Y
	`javax.servlet.forward.query_string`	N
Include	`javax.servlet.include.request_uri`	N
	`javax.servlet.include.context_path`	N
	`javax.servlet.include.servlet_path`	Y
	`javax.servlet.include.path_info`	Y
	`javax.servlet.include.query_string`	N

Legend:

Y: Decoded

N: Not decoded

For details on each attribute and the values stored in each attribute, see *Servlet specifications*.

(3) Inheriting HTTP session

The context path is not decoded and is handled as the original string, so the HTTP session is inherited.

6.19 Settings for acquiring the in-process HTTP server log

This section describes the items that can be set up for acquiring the in-process HTTP server log.

In the in-process HTTP server, the access log, trace based performance analysis, thread trace, and the communication trace are output for supporting the application development, for performance analysis at operation time, and for troubleshooting at failure detection time. For these files, you can change the number of files and the file size in the Easy Setup definition file.

The following table describes the settings that you can change for acquiring the in-process HTTP server log and parameters of the Easy Setup definition file corresponding to the items.

Table 6–29: Settings for acquiring the in-process HTTP server log

Log or trace	Items	Corresponding parameters of the Easy Setup definition file
Access log	Availability access log output	`webserver.logger.access_log.inprocess_http.enabled` in the `<configuration>` tag on a logical J2EE server (j2ee-server) (By default access log is output)
	Access log file name	`webserver.logger.access_log.inprocess_http.filename` in the `<configuration>` tag on a logical J2EE server (j2ee-server)
	File size of the access log	`webserver.logger.access_log.inprocess_http.filesize` in the `<configuration>` tag on a logical J2EE server (j2ee-server)
	Number of files of access log	`webserver.logger.access_log.inprocess_http.filenum` in the `<configuration>` tag on a logical J2EE server (j2ee-server)
	Format name of the access log	`webserver.logger.access_log.format_list` in the `<configuration>` tag on a logical J2EE server (j2ee-server)#
	Output format of access log	`webserver.logger.access_log.`*format-name* in the `<configuration>` tag on a logical J2EE server (j2ee-server)#
	Format when the access log is output	`webserver.logger.access_log.inprocess_http.usage_format` in the `<configuration>` tag on a logical J2EE server (j2ee-server)#
Trace based performance analysis	--	Specify acquisition condition, when you want to execute the `cprfed` command for performing a daily system operation same as for other trace based performance analysis. For details on acquiring the performance analysis trace file, see *15. Performance Analysis Trace*.
Thread trace	Number of files of thread trace	`webserver.logger.thread_trace.inprocess_http.filenum` in the `<configuration>` tag on a logical J2EE server (j2ee-server)
	File size of thread trace	File size of the thread trace = (((A+B) × 32,786)+ 32,914) byte A= Value of `webserver.connector.inprocess_http.max_connections` parameter in the `<configuration>` tag on a logical J2EE server (j2ee-server) B=For `0`, the value of the `webserver.connector.inprocess_http.send_timeout` parameter is 0, and when other than 0, the value is `1` in the `<configuration>` tag on a logical J2EE server (j2ee-server)

Log or trace	Items	Corresponding parameters of the Easy Setup definition file
Communication trace	Number of files of the communication trace	`webserver.logger.communication_trace.inprocess_http.filenum` in the `<configuration>` tag on a logical J2EE server (j2ee-server).
	File size of the communication trace	File size of the communication trace=(((A+B) × 172,050)+128) byte A=Value of `webserver.connector.inprocess_http.max_connections` parameter in the `<configuration>` tag on a logical J2EE server (j2ee-server) B=For `0`, the value of the `webserver.connector.inprocess_http.send_timeout` parameter is 0, and when other than 0, the value is `1` in the `<configuration>` tag on a logical J2EE server (j2ee-server).

Legend:

--: Not applicable

#

In the access log, you can customize the log output format by defining the format with the above-mentioned keys. For customization of the access log of the in-process HTTP server, see *6.17.2 Customizing the access log of the in-process HTTP server*.

6.20 cjtracesync (synchronize trace file information for in-process HTTP server)

This section describes the `cjtracesync` command, which synchronizes the trace files used by the in-process HTTP server.

Format

```
cjtracesync [-h] [-thr|-comm] server-name
```

Function

When the in-process HTTP server is used, this command synchronizes the thread trace and communication trace information with the information in the shared memory. When the in-process HTTP server is being used and you want to obtain failure information without using Management Server functionality, execute this command immediately before obtaining the information.

Arguments

`-h`

Displays the command's usage.

`-thr`

Specifies that thread trace information is to be retrieved. If the `-thr` option and the `-comm` option are both omitted, the command updates the information for both the thread trace and the communication trace.

`-comm`

Specifies that communication trace information is to be retrieved. If the `-thr` option and the `-comm` option are both omitted, the command updates the information for both the thread trace and the communication trace.

server-name

Specifies the name of the J2EE server for which trace information is to be retrieved. A server name must be specified.

Return values

`0:`

The command terminated normally.

`1:`

The command terminated abnormally.

`9:`

The command could not be executed because there are no administrator privileges (in Windows).

6.21 Precautions when operating SOAP applications

The following describes precautions when operating SOAP applications.

6.21.1 Precautions when stopping the J2EE server using the in-process HTTP server

If you try to stop a J2EE server while an in-process HTTP server is being used, the J2EE server might not stop or might take a long time to stop.

This problem occurs when all of the following conditions are met:

- An attempt is made to stop the J2EE sever.
- An in-process HTTP server is being used on the Web server on the SAOP service side.
- Either a large value or `0` is specified as the value of `webserver.connector.inprocess_http.persistent_connection.timeout`.
- A SOAP client whose connection pooling is enabled either is connected with a SOAP service that is running or has connected with such a SOAP service at least once.

The following describes the symptoms that might occur with this problem, by specification and method of stopping:

- If a large value is specified for `webserver.connector.inprocess_http.persistent_connection.timeout` for the in-process HTTP server

  When the `cjstopsv` command is used to stop the J2EE server, KDJE39514-W is output, and it might take up to the number of seconds specified for this property.
- If `0` is specified for `webserver.connector.inprocess_http.persistent_connection.timeout` for the in-process HTTP server

  The J2EE server might not stop.
- If Management Server is used to stop the server

  The KEOS20011-E message might be output to indicate that the attempt to stop the logical server failed, and the server might be forcibly stopped.

When you execute the `cjstopsv` command, if KDJE39514-W is output and it might take a long time for the J2EE server to stop, stop the J2EE server by re-executing the `cjstopsv` command with the `-f` option specified.

When using Management Server to stop the J2EE server, and if you have set a stop monitoring time, the processing is forcibly stopped when the stop monitoring time has passed. If you did not set a stop monitoring time, stop the J2EE server by performing the following steps:

1. On the management portal, click the **Start/Stop Logical Server** anchor.
2. In the **Server View** tab, from **Logical J2EE server**, click **J2EE Server** and then **J2EE Server Name**.
3. Click the **Start/Stop** tab.
4. Click the **Forcibly Stop** button.

7 Cosminexus JAX-RS Engine (JAX-RS 1.1)

This chapter describes the Cosminexus JAX-RS engine in V9 compatibility mode.

7.1 Cosminexus JAX-RS engine in V9 compatibility mode (JAX-RS 1.1)

Cosminexus JAX-RS engine (JAX-RS 1.1) is the only JAX-RS functionality that can be used in V9 compatibility mode. For details on the Cosminexus JAX-RS engine (JAX-RS 1.1), see the manual *uCosminexus Application Server Web Service Development Guide*.

8 How to Use JPA with Application Server

This chapter gives an overview of the JPA and describes how to use JPA with Application Server.

8.1 Organization of this chapter

The JPA is a specification aimed at simplifying the designing and coding of the database-related processing and is used for mapping Java objects and relational database (O/R mapping). By using the JPA, you can operate the database information as a Java object (entity), and set up a system efficiently.

This chapter gives an overview of JPA and describes the JPA usage with Application Server. The following table describes the organization of this chapter.

Table 8–1: Organization of this chapter (Using JPA with Application Server)

Category	Title	Reference location
Description	Features of JPA	*8.2*
	JPA functionality that can be used with Application Server	*8.3*
	EntityManager	*8.4*
	Persistence context	*8.5*
Implementation	How to obtain a container-managed EntityManager	*8.6*
	How to obtain an application-managed EntityManager	*8.7*
	Definitions in persistence.xml	*8.8*
	Allocating persistence.xml	*8.9*
	JPA interfaces	*8.10*
Notes	Notes on setting up applications	*8.11*

Note:

There is no specific description of *Settings* and *Operations* for this functionality.

8.2 Features of JPA

This section describes features of the applications using JPA.

8.2.1 Advantages of applications using JPA

The following contents can be realized, if you use an application using JPA:

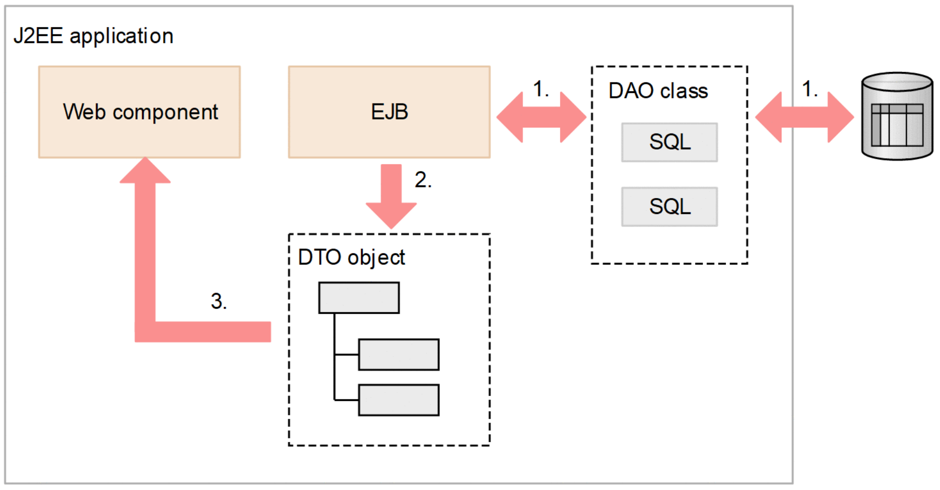
- By concealing the O/R mapping and database access processing, the user programming becomes easy.
- By using annotations, the cost of creating the definition file is reduced and coding is implemented using POJO (Plain Old Java Object).
- By setting up the default values, the amount of coding by the user can be reduced.
- Cosminexus JPA provider conforms to the JPA specifications, and therefore, you can merge the applications used with other JPA providers.

This section compares the data access models when the JPA is not used and when the JPA is used, and describe the advantages of applications using the JPA.

(1) Data access model when the JPA is not used

To access a database from a J2EE application when the JPA is not used, you can use the data access model shown in the following figure to create an application.

Figure 8–1: Database access model when JPA is not used



This section describes the above figure.

With the data access model shown in the figure, you create a class called *DAO* corresponding to the table to hide the SQL statement from the business logic. With the DAO class, you create a process to issue an SQL statement using the JDBC interface. The flow of processing shown in the figure is as follows:

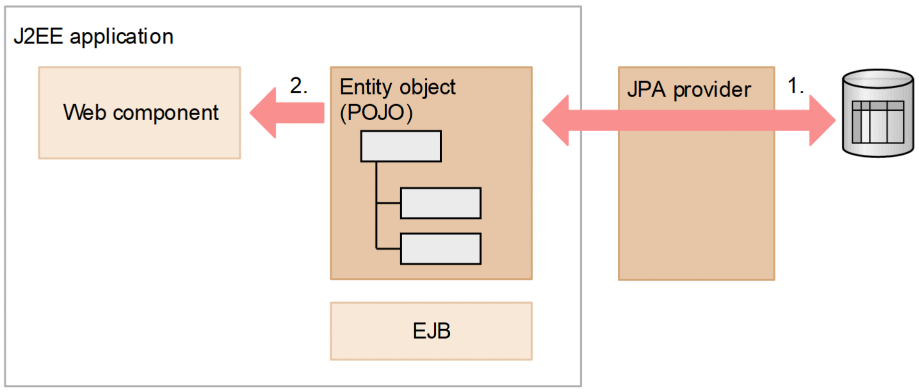
1. With an EJB where the business logic is coded, DAO is used to read the data from a database.
2. The acquired data is stored in an object called *DTO*.
3. The DTO object returns to the Web component. With the Web component, the data acquired from the database is output to a Web page.

With such a data access model, if the data model of the database becomes more complex, the number of DAO, SQL, and DTO classes that must be created also increases. The creation of DAO, SQL, and DTO involves monotonous manual work, so this decreases the productivity of an application development.

(2) Data access model when the JPA is used

The following figure shows a data access model when the JPA is used.

Figure 8–2: Database access model when the JPA is used



When the JPA is used, you create a class corresponding to the lines in the database table. This class is called an *entity class*. With an EJB where the business logic is coded, you can code the processing as if the object of this entity class is directly stored in the database.

The description of the figure is as follows:

1. The JPA engine called the JPA provider issues an SQL statement for the database. Also, the JPA provider automatically synchronizes the status of the entity object and the database table.
2. The entity object can pass the obtained data to the Web component as is.

If you use the JPA, you need not create DTO. Furthermore, the entity class can also be automatically generated from the database table schema by using a development tool such as Eclipse. Because you do not need to create the DAO, SQL, and DTO classes that were the reason of decreased productivity in the past data access models, you can further improve the productivity of the applications.

For details about the entity classes and the JPA providers, see *8.2.2 Entity class* and *8.2.3 JPA provider*.

8.2.2 Entity class

When you use the JPA, you create a class that forms a data container with applications. This class is called the *entity class*. Normally, you create the entity class so that one object of the entity class corresponds to one line in the database table. You create an entity class using the normal Java class (POJO). Special interfaces need not be implemented.

You specify the mapping for the values of the entity class fields and the columns of the database table where the values are stored, using annotations in the items such as the entity class fields. However, the CoC concept has been introduced in JPA in order to improve the easy development. If you do not specify mapping explicitly, the default mapping rules are applied. For example, if the mapping of fields is not explicitly specified, the corresponding column is presumed from the field name and mapped.

8.2.3 JPA provider

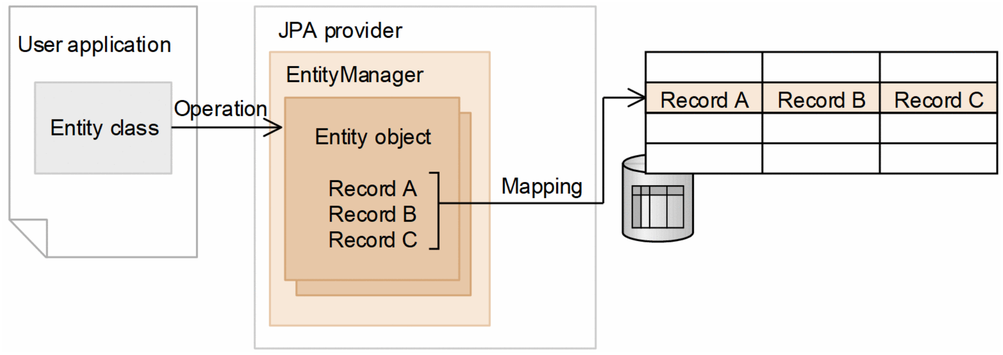
The *JPA provider* is a JPA implementation that provides the following mapping functionality, API, and query language. The following functionality is provided with the JPA provider:

- Functionality for mapping Java objects and a database
- API encapsulating the database processing
- Query language that can be commonly used in the JPA specifications

The advantage of using the JPA provider is that you can design an application without knowing the processing related to database exchanges. Also, by using the query language JPQL available with the JPA provider, you can also send a query even if you are unfamiliar with the database.

The following figure shows the mapping functionality provided by JPA providers.

Figure 8–3: Overview of the mapping functionality provided by JPA providers



This section describes the above figure.

An entity class is prepared in the application and an entity object is generated from the entity class. By changing the contents of the entity object, the user changes the database contents. As a result, the user can update the database contents without knowing the database processing.

The JPA provider maps the entity objects to the database records. The JPA provider also executes the search, insert, delete, or update processing implemented by the user for the database.

The generated entity object is managed by EntityManager. If the value of an entity object field is changed, EntityManager automatically detects the change and applies the change to the database table.

During the following processing, EntityManager is invoked:

- To add the data of the entity class object in the database table.
- To search the data already stored in the database and to extract the data as an entity class object.

For details on EntityManager, see *8.4 EntityManager*.

8.3 JPA functionality that can be used with Application Server

This section describes the JPA providers, components, and resource adapters that can be used with Application Server, when the JPA is used.

8.3.1 Available JPA providers

The *JPA provider* is an engine that provides the EntityManager functionality. The JPA providers that can be used with Application Server include Cosminexus JPA provider and the JPA providers provided by other vendors. The following points describe the usage of each of these JPA providers:

(1) When Cosminexus JPA provider is used

Cosminexus JPA provider is a JPA provider provided with Application Server. Cosminexus JPA provider provides the Cosminexus JPA provider-specific functionality in addition to the functionality based on the JPA 1.0 specifications. For an overview of Cosminexus JPA provider, notes on application implementation, and the usage methods of Cosminexus JPA provider, see *9. Cosminexus JPA Provider*.

(2) When the JPA providers from other vendors are used

The JPA providers are provided by other vendors. The JPA specifications clearly specify the interfaces between JPA providers and Application Server, and therefore you can also use JPA providers provided by other vendors and conforming to the JPA 1.0 specifications with Application Server.

When you use the other JPA providers from Application Server, you must specify the following settings:

- **Specifying JAR files**
  You use one of the following methods to specify the JAR file containing the JPA provider implementation:
  - Specify the JAR file under the `<configuration>` tag of the logical J2EE server (`j2ee-server`) in the Easy Setup definition file. To specify a JAR file, you specify `add.class.path` in the `<param-name>` tag and the JAR file in the `<param-value>` tag. For details on the Easy Setup definition file and parameters to specify, see *12.2.1 Parameters used for setting up the user properties for the J2EE server*.
  - Include JAR files in J2EE applications as a library.
- **Definitions in persistence.xml**
  In the `<provider>` tag of `persistence.xml`, specify the implementation class name of `javax.persistence.PersistenceProvider` provided by the used JPA provider. For details, see *8.8.2(2) <provider> tag*.

To use the functionality for monitoring the J2EE application execution time provided with Application Server, you must add the JPA provider classes and entity classes into the protected area list. For details on how to add a class into the protected area list, see *2.2.5 criticalList.cfg (Protected areas list file)* in the *uCosminexus Application Server Definition Reference Guide*.

Reference note

With the execution of applications using the JPA, you can use the trace based performance analysis functionality provided with Application Server.

- When Cosminexus JPA provider is used as the JPA provider
  The trace based performance analysis can be output with both Application Server and Cosminexus JPA provider.
- When a JPA provider from other vendors is used
  You can only use the trace based performance analysis output by Application Server.

Note that with Application Server, the trace based performance analysis is output by the EntityManagerFactory, EntityManager, EntityTransaction, and Query APIs of the `javax.persistence` package. Furthermore, the trace based performance analysis related to the entity life cycle callback is output with the JPA provider.

For an overview of the trace based performance analysis, see *15.2 Overview of the trace based performance analysis of Application Server*. For key points on output of the performance analysis trace, see *15. Performance Analysis Trace*.

8.3.2 Available components

With Application Server, you can use the JPA with EJBs and Web applications. You can also use the JPA when user threads are used from Web applications. Note that you cannot use the JPA in the following environments or libraries:

- EJB client application environment
- J2EE application client environment
- Container extension library

The following table lists the components that can use the JPA.

Table 8–2: Components that can use the JPA

Components		JPA usage
EJB	Stateless Session Bean (EJB 3.0 and later)#1	Y
	Stateful Session Bean (EJB 3.0 and later)#1	Y
	Stateless Session Bean (earlier than EJB 3.0)	N
	Stateful Session Bean (earlier than EJB 3.0)	N
	Interceptor	Y
	Message-driven Bean	N
	Entity Bean	N
Web application	Servlet, filter, event listener (Servlet 2.5 and later)	Y
	JSP, JSP tag handler, JSP event listener, JSP tag library event listener #2 (Servlet 2.5 and later)	Y
	Servlet, filter, event listener (earlier than Servlet 2.5)	N
	JSP, JSP tag handler, JSP event listener, JSP tag library event listener (earlier than Servlet 2.5)	N

Legend:
Y: Available
N: Not available

#1

With Application Server, you cannot specify the JPA definition using EJB 3.0 `ejb-jar.xml`. Therefore, you use annotations such as `@PersistenceUnit` and `@PersistenceContext` to define the references for the persistence units and persistence contexts.

#2

With Application Server, you cannot use annotations in the JSP tag library event listener. To use the JPA functionality in the JSP tag library event listener, you use `<persistence-unit-ref>` tag and the `<persistence-context-ref>` tag of `web.xml` to define the references for the persistence units and persistence contexts.

Important note

If `true` is specified in the `metadata-complete` attribute of `web.xml` in Servlet 2.5 and later, the Web component annotations are not read. Therefore, you cannot use annotations to define the references for the persistence contexts or persistence units. However, you can define the references in `web.xml`.

8.3.3 Supported application formats

A J2EE application using the JPA is deployed on Application Server in one of the following formats:

- **Archive-format J2EE applications**
- **Exploded-archive format J2EE applications**

You can also replace a J2EE application that uses a deployed JPA. For the archive format, use the redeploy functionality and for the exploded archive format, use the reload functionality.

Note that when you use the reload functionality, updates are not detected for the O/R mapping file. However, the O/R mapping file is re-read when the file is reloaded. The following table describes the targets for update detection and the re-reading during reload.

Table 8–3: Targets for update detection and re-reading when reload is executed

Target classes and files	Update detection	Re-reading
Entity class	Y	Y
Mapped super class	Y	Y
Embedded class	Y	Y
`persistence.xml`	Y	Y
O/R mapping file (When `orm.xml` is allocated under `META-INF`)	N	Y
O/R mapping file (When `orm.xml` is allocated to the location specified in the `<mapping-file>` tag of `persistence.xml`)	Y	Y

Legend:

Y: Target

N: Not a target

For details on archive-format J2EE applications and exploded archive-format J2EE applications, see the manual *uCosminexus Application Server Common Container Functionality Guide*.

Important note

When you use an application with the JPA in the exploded archive format, do not delete the class or library JAR while the application is running. If the class or library JAR is deleted, Application Server and JPA provider might perform unexpected operations.

8.3.4 Supported class loader configuration

The J2EE applications using the JPA support the class loaders listed in the following table.

Table 8–4: Class loaders supported in a J2EE application using the JPA

Class loaders	Support
Default class loader configuration	Y
Class loader configuration used when local call optimization is performed #	Y
Class loader configuration for downward compatibility	N

Legend:

Y: Supported

N: Not supported

Note:

The class loader configuration for the downward compatibility is used only in the basic mode and therefore, not supported.

#

Indicates the following specification in the `<configuration>` tag of the logical J2EE server (`j2ee-server`) in the Easy Setup definition file:

```
<param-name>ejbserver.rmi.localinvocation.scope</param-name>
<param-value>all</param-value>
```

8.3.5 available resource adapters

When you execute a J2EE application that uses the JPA, with Application Server, you can use a resource adapter with the connection factory interface `javax.sql.DataSource`. You can use DB Connector as the resource adapter provided with Application Server.

Also, you must deploy the resource adapter you want to use as the J2EE resource adapter. In the case of the J2EE applications using the JPA, you cannot include and deploy the resource adapter in the J2EE application.

The following table lists the resource adapters available with Application Server.

Table 8–5: Resource adapters available with Application Server

Connection factory interface	Deploy format of the resource adapter	Usage from the JPA
`javax.sql.DataSource`	Deployed as a J2EE resource adapter	Y
	Included and deployed in the J2EE application	N
Other than `javax.sql.DataSource`	--	N

Legend:

Y: Available

N: Not available

--: Not applicable

For details on the resource adapters available when you use Cosminexus JPA provider, see *9.2.3(3) Available DB Connectors*.

8.4 EntityManager

The *EntityManager* is an object that has an interface for registering and deleting entities for the database. This section gives an overview of EntityManager.

8.4.1 Methods provided with EntityManager

The EntityManager provides methods. The typical methods are as follows:

- **persist method (equivalent to SQL INSERT)**

  The method used to add an entity object that executes `new` in the application, into the database.
- **find method (equivalent to SQL SELECT)**

  The method used to search an entity object from the database.
- **remove method (equivalent to SQL DELETE)**

  The method used to delete an entity object from the database.

The entity objects searched using the `find` method from EntityManager and the entity objects passed to EntityManager using the `persist` method are managed by EntityManager. If a field value of an entity object managed by EntityManager is changed, EntityManager automatically detects the change and applies the change to the database table.

With the JPA, an entity object managed by EntityManager is called a *managed entity*. By default, when a transaction is concluded, the entity is no longer managed by EntityManager. An entity that is no longer managed by EntityManager is called a *detached entity*.

8.4.2 Types of EntityManager

The types of EntityManager include the container-managed EntityManager and the application-managed EntityManager. The following is a description of each type:

(1) Container-managed EntityManager

The method entrusts the creation and destruction of EntityManager to the container. If you use a container-managed EntityManager, you can code an application without being aware of the generation and destruction of EntityManager. The following points describe how to obtain and how to destroy the container-managed EntityManager.

- **How to obtain the container-managed EntityManager**

  To obtain the container-managed EntityManager, you use the DI or JNDI lookup in the application. EntityManager obtained using this method is EntityManager created by the container. You can use EntityManager obtained from a container as it is during the application coding.

  For details on how to obtain the container-managed EntityManager from an application, see *8.6 How to obtain the container-managed EntityManager*.
- **How to destroy the container-managed EntityManager**

  The creation and destruction of EntityManager need not be coded in the application.

(2) Application-managed EntityManager

In this method, the application explicitly creates and destroys EntityManager. The life cycle is managed explicitly by application coding. The following points describe how to obtain and how to destroy the application-managed EntityManager.

- **How to obtain the application-managed EntityManager**

  You use EntityManagerFactory to create EntityManager in the application. To obtain EntityManagerFactory, you use the DI or JNDI lookup in the application.

  For details on how to obtain the application-managed EntityManager, see *8.7 How to obtain the application-managed EntityManager*.
- **How to destroy the application-managed EntityManager**

  You invoke the `close` method of EntityManager to destroy EntityManager.

8.4.3 Transaction control and EntityManager

The following two types of EntityManager are available depending on how the transactions are controlled:

- **JTA entity manager**

  EntityManager in which the transactions are controlled by the JTA.
- **Resource local entity manager**

  EntityManager in which the transactions are controlled by the EntityTransaction API.

The following table describes the relationship between the types of EntityManager and the transaction control methods.

Table 8–6: Relationship between types of EntityManager and transaction control methods

Types of EntityManager	Transaction control method	
	JTA	Resource local
Container-managed EntityManager	Y[#1]	N
Application-managed EntityManager [#2]	Y	Y

Legend:

Y: Transaction can be controlled

N: Transaction cannot be controlled

JTA: JTA entity manager

Resource local: Resource local entity manager

#1

The transaction is necessarily controlled by the JTA.

#2

You can select whether to control the transaction using the JTA or whether the application controls the transaction, by explicitly using the `EntityTransaction` API. When the `EntityTransaction` API is used, the transaction becomes a resource local transaction. Even if the JTA transaction exists, the transaction is controlled regardless of the JTA transaction.

You specify whether you want to use the JTA entity manager or the resource local entity manager in the definition of the persistence unit. For details on how to specify definitions in the persistence unit, see *8.8.1(2) transaction-type attribute*.

8.4.4 Persistence unit

You must define the following information, when the JPA is used from the application:

- Information about the entity classes in the application
- Information about the mapping between the entity classes and the database tables
- Information about the data source for the JPA provider to obtain the database connection

The unit that defines this information is called the *persistence unit*.

You define the persistence unit in `persistence.xml`. When the JPA is used in the Java EE environment, `persistence.xml` is allocated in the determined location in the EJB-JAR, WAR, or EAR files when the user packages the application.

You can include multiple persistence unit definitions in the `persistence.xml` file. You can also include multiple `persistence.xml` files in one application. As a result, you can define multiple persistence units in one application. When multiple persistence units are defined in an application, you specify the persistence unit to be used by the application in the `unitName` attribute of `@PersistenceContext`. Note that when the persistence unit to be used can be identified uniquely, such as when only one persistence unit is defined in the application, you can omit the `unitName` attribute.

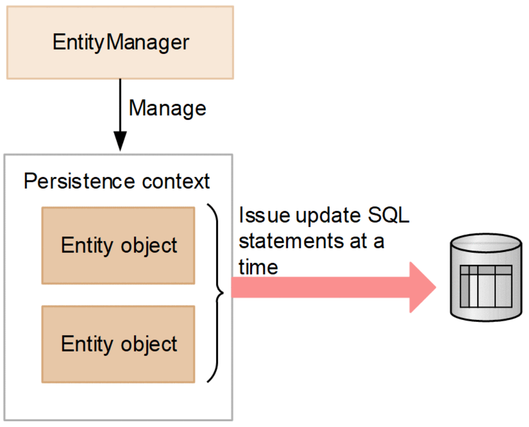
8.5 Persistence context

The EntityManager caches the entity objects to be updated and the searched entity objects. The persistence context is the cache of the entity objects cached by EntityManager.

8.5.1 EntityManager and persistence context

The following figure shows a relationship between EntityManager and persistence context.

Figure 8–4: Relationship between EntityManager and persistence context



The EntityManager inserts the managed entity object in the persistence context and manages the entity objects. When the application passes the entity object to the `persist` method or updates the field value of the managed entity object, the status of the entity object in the persistence context is changed. EntityManager synchronizes the statuses of the entity objects in the persistence context and the database table just before the transaction is committed. In order to apply the statuses of the entity objects within the persistence context to the database table, the update SQL statements are issued collectively at this time. As a result, the database locking time is shortened, and therefore you can improve the concurrent executability and update the data efficiently.

(1) Types of EntityManager

You decide whether to use the container-managed EntityManager or the application-managed EntityManager depending on the relationship between the persistence context and transaction.

- **When the persistence context needs to be automatically propagated along with the JTA transaction**
  You use the container-managed EntityManager. If you use the container-managed EntityManager, the persistence context is automatically propagated along with the JTA transaction. Therefore, when multiple components are invoked in one JTA transaction, EntityManager used in the same JTA transaction can be associated with the same persistence context.
  As a result, the application need not pass the EntityManager references to the arguments used when the references are invoked from one component in another component.
- **When the application needs to use the persistence context independently from the JTA transaction**
  You use the application-managed EntityManager. If you use the application-managed EntityManager, when you use another EntityManager in the same JTA transaction also, this EntityManager does not share the persistence context and has an independent persistence context.

(2) Types of persistence context

The persistence contexts are of two types depending on the lifetime:

- **Transaction scope persistence context**
- **Extended persistence context**

For the container-managed EntityManager, you can choose the type of persistence context. You specify the type of persistence context in the `type` attribute of `@PersistenceContext`. The default type is the transaction scope persistence context.

Note that the extended persistence context is always used for the application-managed EntityManager. You cannot choose the type of persistence context.

8.5.2 Persistence context when the container-managed EntityManager is used

When you use the container-managed EntityManager, the life cycle of the persistence context is managed by the container and the persistence context is automatically propagated along with the JTA transaction. You can choose the transaction scope persistence context or the extended persistence context as the type of the persistence context life cycle.

The following sections describe the respective persistence contexts:

(1) Transaction scope persistence context

With the JPA specifications, the persistence context having the same life cycle as the transaction is called the *transaction scope persistence context*.

The EntityManager has the same life cycle as that of the transaction by default. Therefore, the updates cached in the persistence context of EntityManager are applied to the database when the transaction is committed.

(a) Life cycle of persistence context

The life cycle of the transaction scope persistence context is as follows:

- **Creating the persistence context**
  The transaction scope persistence context is created when the container-managed EntityManager is first invoked in a JTA transaction.
  The created persistence context is associated with the JTA transaction.
  After that, when the container-managed EntityManager is used in the same JTA transaction, this persistence context is used.
- **Destroying the persistence context**
  When the JTA transaction is committed or rolled back, the transaction scope persistence context is destroyed.

If the container-managed EntityManager is invoked outside the transaction, all the entities loaded from the database are immediately detached when the invocation of the EntityManager method ends.

(2) Extended persistence context

With the Java EE environment, if EntityManager is used from the Stateful Session Bean, you can set the same persistence context lifetime as the Stateful Session Bean lifetime. In this case, the updates are applied to the database every time the transaction is committed, but the entity objects managed in the persistence context are stored in the managed status as are across multiple transactions. The persistence context with the same life cycle as the Stateful Session Bean is called the *extended persistence context* in the JPA.

The extended persistence context is created simultaneously when the Stateful Session Bean is created and is associated with that Stateful Session Bean. Thereafter, the extended persistence context is destroyed simultaneously when the Stateful Session Bean is destroyed.

When the Stateful Session Bean creates another Stateful Session Bean and when the definition is such that the creating Stateful Session Bean and the created Stateful Session Bean both use the extended persistence context, the persistence context on the creating side is inherited on the created machine. The persistence context is inherited regardless of whether the transaction is active when the Stateful Session Bean is created. If the persistence context is inherited when the Stateful Session Bean is created, the persistence context is destroyed when all the Stateful Session Beans sharing that persistence context are destroyed.

(3) Extended persistence context and transactions

The extended persistence context exists from the time the EntityManager instance is created until the instance is closed. The extended persistence context supports multiple transactions and the invocation outside the EntityManager transaction.

The relationship with transactions is as follows:

- If EntityManager is invoked within the transaction scope or if the stateful session beans bound by the persistence context are invoked in the transaction scope, the entities managed by EntityManager participate in the transaction.
- Regardless of whether the transaction is running, the `persist`, `remove`, `merge`, and `refresh` operations might be performed. In this case, EntityManager participates in the transaction and the updates are applied in the database when the transaction is committed.
- Even after the transaction is committed, the references to the entity object are stored. The entity object is managed by EntityManager and is updated as an object managed between transactions (managed entity).

(4) Propagation of persistence context

When the container-managed EntityManager is used, the persistence context is propagated using the JTA transaction and might be associated with multiple EntityManager. However, the persistence context is only propagated with the same Application Server. The persistence context is not propagated in a remote Application Server.

The following points separately describe the propagation of the persistence context for the states when a component is invoked:

If the JTA transaction does not exist or the persistence context is not associated with the JTA transaction when the component is invoked

The persistence context is not propagated. The operations when EntityManager is invoked from this component are as follows:

- When EntityManager using the transaction scope persistence context is invoked, a new persistence context is created.
- When EntityManager using the extended persistence context is invoked, the extended persistence context associated with the invoked Stateful Session Bean is used.
- If the JTA transaction exists when EntityManager is invoked, the persistence context is associated with the JTA transaction.

If the JTA transaction is propagated and the persistence context is associated with the JTA transaction when the component is invoked

The operations when EntityManager is invoked from this component are as follows:

- If another persistence context is associated with the JTA transaction although the component is the Stateful Session Bean that already has the extended persistence context, the container throws `EJBException`.
- When EntityManager using the transaction scope persistence context is invoked, the persistence context associated with the propagated JTA transaction is used.

8.5.3 Persistence context when the application-managed EntityManager is used

When you use the application-managed EntityManager, the application directly invokes EntityManagerFactory of the JPA provider and manages the EntityManager life cycle and the creation and destruction of the persistence context. The life cycle of the application-managed persistence context can manage life cycles across multiple transactions.

(1) Managing the EntityManager life cycle

With the application, you use the `close` and `isOpen` methods of EntityManager to manage the life cycle of the application-managed EntityManager. If you invoke the `close` method of EntityManager, EntityManager, persistence context associated with EntityManager, and the other resources are released. After the `close` method is invoked, do not invoke methods other than EntityManager `getTransaction` method and `isOpen` method with the application. If other methods are invoked, `IllegalStateException` is thrown. If the `close` method is invoked when the transaction is active, the persistence context remains stored until the transaction is concluded. The `isOpen` method of EntityManager returns `true` until EntityManager is closed and `false` after EntityManager is closed.

(2) Life cycle of persistence context

The life cycle of the application-managed persistence context is as follows:

- **Creating the persistence context**
  The persistence context is created when the `createEntityManager` method of EntityManagerFactory is invoked.
- **Destroying the persistence context**
  The persistence context is destroyed when the `close` method of EntityManager is invoked.

The application-managed persistence context is independent of the transaction. Therefore, this persistence context is not propagated along with the JTA transaction.

(3) Notes on using the JTA entity manager

When the JTA entity manager is used in the application-managed EntityManager, the invocation of `joinTransaction` of EntityManager when the application creates EntityManager outside the JTA transaction scope is the responsibility of the application. The application must report EntityManager that the transaction has started, so invoke the `joinTransaction` method of EntityManager after the transaction starts.

8.6 How to obtain the container-managed EntityManager

To obtain the container-managed EntityManager from the J2EE application, use one of the following methods:

- **Method of injecting EntityManager in the application field and setter method by using the DI**
- **Method of looking up EntityManager by using the JNDI from the application**

The following sections describe each of the methods.

8.6.1 Method of injecting EntityManager in the application

The following two more methods are available for injecting EntityManager in the application fields and in the `setter` method:

1. Method of adding `@PersistenceContext` in the field or method at the inject destination
2. Method of specifying definitions in the `<persistence-context-ref>` tag of a DD (`web.xml`)

However, you cannot specify the JPA definition using EJB 3.0 `ejb-jar.xml` with Application Server. Therefore, if you want to use the JPA in an EJB, use the method specified in point 1.

The following is a description of the methods:

(1) Method of using @PersistenceContext

When you use `@PersistenceContext` to inject EntityManager, you add `@PersistenceContext` in the field and `setter` method where EntityManager is to be injected. The attributes that can be specified in `@PersistenceContext` are as follows:

(a) unitName attribute

In the `unitName` attribute, you specify the name of the persistence unit defined in `persistence.xml`. However, when the persistence unit to be used can be uniquely identified, such as when only one persistence unit is defined in the EJB-JAR and WAR or EAR file, you can omit the `unitName` attribute. For details on the persistence unit used when the `unitName` attribute is omitted, see *8.11.2 Reference scope of the persistence unit name*.

(b) type attribute

In the `type` attribute, you specify the life cycle type of the persistence context. You can specify `PersistenceContextType.TRANSACTION` or `PersistenceContextType.EXTENDED`.

- **When PersistenceContextType.TRANSACTION is specified**
  The transaction scope persistence context is used and the transaction lifetime and the persistence context lifetime becomes the same.
- **When PersistenceContextType.EXTENDED is specified**
  The extended persistence context is used and the Stateful Session Bean lifetime and the persistence context lifetime becomes the same.
  Note that `@PersistenceContext` in which `PersistenceContextType.EXTENDED` is specified in the `type` attribute can only be added in the field or method of the Stateful Session Bean.

When the `type` attribute is omitted, the default value is `PersistenceContextType.TRANSACTION`.

(c) properties attribute

In the `properties` attribute, you can specify the properties for the JPA provider used for setting up a persistence unit. The properties specified here are passed to the JPA provider when EntityManager is obtained from the JPA provider.

(d) name attribute

When you use injection, normally you need not specify the `name` attribute, but if specified, EntityManager is registered in the JNDI Namespace (`java:comp/env`) with the name specified in the `name` attribute. An example of using `@PersistenceContext` to inject EntityManager is as follows:

```
@Stateless
public class InventoryManagerBean implements InventoryManager {
 @PersistenceContext(unitName="myUnit")
 private EntityManager em;
 ...
}
```

(2) Method of using the <persistence-context-ref> tag of the DD

When you use a DD to inject EntityManager, you define the following tags in the `<persistence-context-ref>` tag of the DD:

(a) <description> tag

In the `<description>` tag, the user can freely code the explanation for the EntityManager references to be defined. Even if this tag is specified, the specified contents do not affect the operations of the application. You can also omit this tag.

(b) <persistence-context-ref-name> tag

In the `<persistence-context-ref-name>` tag, specify the name with which EntityManager is registered in the JNDI Namespace. The specified name is the relative path from `java:comp/env`. The JNDI registration name of EntityManager is not mandatory, but JPA specifications recommend that the name be set under `java:comp/env/persistence`.

(c) <persistence-unit-name> tag

In the `<persistence-unit-name>` tag, you specify the name of the persistence unit defined in `persistence.xml`. However, when the persistence unit to be used can be uniquely identified, such as when only one persistence unit is defined in the EJB-JAR and WAR or EAR, you can omit the `<persistence-unit-name>` tag. For details on the persistence unit used when the `<persistence-unit-name>` tag is omitted, see *8.11.2 Reference scope of the persistence unit name*.

(d) <persistence-context-type> tag

In the `<persistence-context-type>` tag, you specify the life cycle type of the persistence context. You specify `Transaction` or `Extended`.

- **When Transaction is specified**

  The transaction scope persistence context is used and the transaction lifetime and the persistence context lifetime becomes the same.

- **When Extended is specified**

The extended persistence context is used and the Stateful Session Bean lifetime and the persistence context lifetime becomes the same.

Note that the <persistence-context-ref> tag in which Extended is specified in the <persistence-context-type> tag can only be defined for the Stateful Session Bean.

When the <persistence-context-type> tag is omitted, the default value is Transaction.

(e) <persistence-property> tag

In the <persistence-property> tag, you can specify the properties for the JPA provider used for setting up the persistence unit. The properties specified here are passed to the JPA provider when EntityManager factory is obtained from the JPA provider. You can omit this tag.

(f) <injection-target> tag

In the <injection-target-class> tag of the <injection-target> tag, you specify the inject destination class. In the <injection-target-name> tag of the <injection-target> tag, you specify the field name or setter method name at the inject destination. An example of defining the <persistence-context-ref> tag in web.xml and injecting EntityManager is as follows:

```
...
<web-app>
 ...
 <servlet>
   <display-name>InventoryManagerServlet</display-name>
   <servlet-name>InventoryManagerServlet</servlet-name>
   <servlet-class>com.hitachi.InventoryManagerServlet</servlet-class>
 </servlet>
 ...
 <persistence-context-ref>
   <description>
    Persistence context for the inventory management application.
   </description>
   <persistence-context-ref-name>persistence/InventoryAppMgr
   </persistence-context-ref-name>
   <persistence-unit-name>InventoryManagement</persistence-unit-name>
   <persistence-context-type>Transaction</persistence-context-type>
   <injection-target>
     <injection-target-class>
      com.hitachi.InventoryManagerServlet
   </injection-target-class>
   <injection-target-name>em</injection-target-name>
  </injection-target>
 </persistence-context-ref>
 ...
</web-app>
...
```

8.6.2 Method of looking up EntityManager from the application

The following two more methods are available for using the JNDI to look up EntityManager from the application:

1. Method of adding `@PersistenceContext` in the class for looking up EntityManager and defining the EntityManager references
2. Method of defining the `<persistence-context-ref>` tag in the DD (`web.xml`) and defining the EntityManager references

However, you cannot specify the JPA definition using EJB 3.0 `ejb-jar.xml` with Application Server. Therefore, if you want to use the JPA in an EJB, use the method specified in point 1.

The following sections describe the methods:

(1) Method of using @PersistenceContext

When you use `@PersistenceContext` to define the EntityManager references, add `@PersistenceContext` in the class that executes lookup.

The attributes that can be specified in `@PersistenceContext` are as follows:

(a) name attribute

In the `name` attribute, you specify the lookup name with which the application code will look up EntityManager. The specified lookup name is the relative path from `java:comp/env`. The lookup name of EntityManager is not mandatory, but the JPA specifications recommend that the name be set up under `java:comp/env/persistence`.

For `@PersistenceContext` the same attributes are used that are used as other attributes in *8.6.1 Method of injecting EntityManager in the application*. Note that only the Stateful Session Bean can look up EntityManager in the extended scope. Also, for adding multiple `@PersistenceContext` in one class, you add `@PersistenceContexts` in the class and specify the array of `@PersistenceContext` as the `value` attribute. An example of using `@PersistenceContext` to look up EntityManager from `SessionContext` is as follows:

```
@Stateless
@PersistenceContext(name="persistence/OrderEM")
public class MySessionBean implements MyInterface {
 @Resource SessionContext ctx;
 public void doSomething() {
     ...
 EntityManager em = (EntityManager)ctx.lookup("persistence/OrderEM");
     ...
 }
}
```

An example of using `@PersistenceContext` to look up EntityManager from `InitialContext` is as follows:

```
@Stateless
@PersistenceContext(name="persistence/InventoryAppMgr")
public class InventoryManagerBean implements InventoryManager {
 public void updateInventory(...) {
     ...
   Context initCtx = new InitialContext();
   EntityManager em = (EntityManager)
    initCtx.lookup("java:comp/env/persistence/InventoryAppMgr");
     ...
 }
}
```

(2) Method of using the <persistence-context-ref> tag of the DD

When you use the DD to define the EntityManager references, define the following tags in the `<persistence-context-ref>` tag of the DD:

(a) <persistence-context-ref-name> tag

In the `<persistence-context-ref-name>` tag, specify the lookup name with which the application code will look up EntityManager. The specified lookup name is the relative path from `java:comp/env`. The lookup name of EntityManager is not mandatory, but the JPA specifications recommend that the name be set under `java:comp/env/persistence`.

For the `<persistence-context-ref>` tag, the same tags are used that are used as other tags in *8.6.1 Method of injecting EntityManager in the application*. However, when EntityManager is obtained with the JNDI lookup, you cannot specify `<injection-target>`. Note that only the Stateful Session Bean can look up EntityManager in the extended scope.

An example of defining `<persistence-context-ref>` in `web.xml` is as follows:

```
...
<web-app>
 ...
 <servlet>
    <display-name>InventoryManagerServlet</display-name>
    <servlet-name>InventoryManagerServlet</servlet-name>
    <servlet-class>com.hitachi.InventoryManagerServlet</servlet-class>
 </servlet>
 ...
 <persistence-context-ref>
   <description>
    Persistence context for the inventory management application.
   </description>
   <persistence-context-ref-name>
    persistence/InventoryAppMgr
   </persistence-context-ref-name>
   <persistence-unit-name>InventoryManagement</persistence-unit-name>
   <persistence-context-type>Transaction</persistence-context-type>
 </persistence-context-ref>
 ...
</web-app>
...
```

8.6.3 Overriding the @PersistenceContext definition using the DD

If the `<persistence-context-ref>` tag is defined in the DD when `@PersistenceContext` is mentioned in the application, the contents defined in the annotation are overwritten by the contents defined in the DD.
In this case, the mapping between the annotations and DD is determined with the mapping between the `name` attribute of `@PersistenceContext` and the `<persistence-context-ref-name>` tag present under the `<persistence-context-ref>` tag of the DD. Note that even if the `name` attribute is not explicitly specified in `@PersistenceContext`, the `name` attribute contains the default value.

The precautions to be taken when the attributes specified in `@PersistenceContext` are overridden with the DD tags are as follows:

(1) <persistence-unit-name> tag and unitName attribute

The `<persistence-unit-name>` tag of the DD overrides the `unitName` attribute of `@PersistenceContext`. Normally, if you change the persistence unit name, the application stops operating, so take care when you define the DD and annotations.

(2) <persistence-context-type> tag and type attribute

The `<persistence-context-type>` tag of the DD overrides the `type` attribute of `@PersistenceContext`. Normally, if you change the life cycle type of the persistence context, the application stops operating, so take care when you define the DD and annotations.

(3) <persistence-property> tag and properties attribute

The properties specified in the `<persistence-property>` tag of the DD are added to the properties specified in the `properties` attribute of `@PersistenceContext`. However, if the property name is the same, the property value is overridden.

(4) <injection-target> tag

The injection target cannot be overridden. Note that when the `<injection-target>` tag is coded in the DD, accurately specify the fields and methods in which `@PersistenceContext` is added.

8.7 How to obtain the application-managed EntityManager

When the application-managed EntityManager is used, the application uses EntityManagerFactory to create EntityManager. There are two methods by which the application obtains EntityManagerFactory:

- Method of using the DI to inject EntityManagerFactory in the application field and `setter` method
- Method of using the JNDI from the application to look up EntityManagerFactory

The following is a description of the methods:

8.7.1 Method of injecting EntityManagerFactory in the application

The following two more methods are available for injecting EntityManagerFactory in the application fields and `setter` method:

1. Method of adding `@PersistenceUnit` in the field or method at the inject destination
2. Method of specifying a definition in the `<persistence-unit-ref>` tag of the DD (`web.xml`)

However, you cannot specify the JPA definition using EJB 3.0 `ejb-jar.xml` with Application Server. Therefore, if you want to use the JPA in an EJB, use the method specified in point 1.

The following sections describe the methods:

(1) Method of using @PersistenceUnit

When you use `@PersistenceUnit` to inject EntityManagerFactory, add `@PersistenceUnit` in the field and `setter` method where EntityManagerFactory is to be injected. The attributes that can be specified in `@PersistenceUnit` are as follows:

(a) unitName attribute

In the `unitName` attribute, you specify the name of the persistence unit defined in `persistence.xml`. However, when the persistence unit to be used can be uniquely identified, such as when only one persistence unit is defined in the EJB-JAR and WAR or EAR, you can omit the `unitName` attribute. For details on the persistence unit used when the `unitName` attribute is omitted, see *8.11.2 Reference scope of the persistence unit name*.

(b) name attribute

When you use injection, normally you need not specify the `name` attribute, but if you specify, EntityManager is registered in the JNDI Namespace (`java:comp/env`) with the name specified in the `name` attribute. An example of using `@PersistenceUnit` to inject EntityManagerFactory is as follows:

```
@Stateless
public class InventoryManagerBean implements InventoryManager {
 @PersistenceUnit(unitName="myUnit")
 private EntityManagerFactory emf;
    ...
}
```

(2) Method of using the <persistence-unit-ref> tag of the DD

When you use the DD to inject EntityManagerFactory, define the following tags in the `<persistence-unit-ref>` tag of the DD:

(a) <persistence-unit-ref-name> tag

In the `<persistence-unit-ref-name>` tag, you specify the name with which EntityManagerFactory is registered in the JNDI Namespace. The specified name is the relative path from `java:comp/env`.

The JNDI registration name of EntityManagerFactory is not mandatory, but the JPA specifications recommend that the name be set under `java:comp/env/persistence`.

(b) <description> tag

In the `<description>` tag, you can freely code the EntityManagerFactory references to be defined. Even if this element is specified, the specified contents do not affect the operations of the application. You can also omit this tag.

(c) <persistence-unit-name> tag

In the `<persistence-unit-name>` tag, you specify the name of the persistence unit defined in `persistence.xml`. However, when the persistence unit to be used can be uniquely identified, such as when only one persistence unit is defined in the EJB-JAR and WAR or EAR, you can omit the `<persistence-unit-name>` tag. For details on the persistence unit used when the `<persistence-unit-name>` tag is omitted, see *8.11.2 Reference scope of the persistence unit name*.

(d) <injection-target> tag

In the `<injection-target-class>` tag of the `<injection-target>` tag, you specify the inject destination class. In the `<injection-target-name>` tag of the `<injection-target>` tag, you specify the field name or `setter` method name at the inject destination. An example of defining the `<persistence-unit-ref>` tag in `web.xml` and injecting EntityManagerFactory is as follows:

```
...
<web-app>
 ...
 <servlet>
   <display-name>InventoryManagerServlet</display-name>
   <servlet-name>InventoryManagerServlet</servlet-name>
   <servlet-class>com.hitachi.InventoryManagerServlet</servlet-class>
 </servlet>
 ...
 <persistence-unit-ref>
   <description>
    Persistence unit for the inventory management application.
   </description>
   <persistence-unit-ref-name>persistence/InventoryAppDB
   </persistence-unit-ref-name>
   <persistence-unit-name>InventoryManagement</persistence-unit-name>
   <injection-target>
     <injection-target-class>
      com.hitachi.InventoryManagerServlet
   </injection-target-class>
   <injection-target-name>emf</injection-target-name>
  </injection-target>
 </persistence-unit-ref>
```

```
  ...
</web-app>
...
```

8.7.2 Method of looking up EntityManagerFactory from the application

The following two more methods are available for using the JNDI to look up EntityManagerFactory from the application:

1. Method of adding `@PersistenceUnit` in the class for looking up EntityManagerFactory and defining the EntityManagerFactory references
2. Method of defining the `<persistence-unit-ref>` tag in the DD (`web.xml`) and defining the EntityManagerFactory references

However, you cannot specify the JPA definition using EJB 3.0 `ejb-jar.xml` with Application Server. Therefore, if you want to use the JPA in an EJB, use the method specified in 1.

The following sections describe the methods:

(1) Method of using @PersistenceUnit

When you use `@PersistenceUnit` to define the EntityManagerFactory references, you add `@PersistenceUnit` in the class that executes lookup. The attributes that can be specified in `@PersistenceUnit` are as follows:

(a) name attribute

In the `name` attribute, you specify the lookup name with which the application code will look up EntityManagerFactory. The specified lookup name is the relative path from `java:comp/env`. The lookup name of EntityManagerFactory is not required, but the JPA specifications recommend that the name be set up under `java:comp/env/persistence`.

For `@PersistenceUnit`, the same attributes are used that are used as the other attributes in *8.7.1 Method of injecting EntityManagerFactory in the application*. Note that to add multiple `@PersistenceUnit` in one class, you add `@PersistenceUnits` in the class and specify the array of `@PersistenceUnit` as the `value` attribute. An example of using `@PersistenceUnit` to look up EntityManagerFactory from `SessionContext` is as follows:

```
@Stateless
@PersistenceUnit(name="persistence/InventoryAppDB")
public class InventoryManagerBean implements InventoryManager {
  @Resource SessionContext ctx;
  public void updateInventory(...) {
    ...
  EntityManagerFactory emf = (EntityManagerFactory)
    ctx.lookup("persistence/InventoryAppDB");
  EntityManager em = emf.createEntityManager();
    ...
 }
}
```

An example of using `@PersistenceUnit` to look up EntityManagerFactory from `InitialContext` is as follows:

```
@Stateless
@PersistenceUnit(name="persistence/InventoryAppDB")
```

```
public class InventoryManagerBean implements InventoryManager {
   public void updateInventory(...) {
   Context initCtx = new InitialContext();
   EntityManagerFactory emf = (EntityManagerFactory)
     initCtx.lookup("java:comp/env/persistence/InventoryAppDB");
   EntityManager em = emf.createEntityManager();
     ...
 }
}
```

(2) Method of using the <persistence-unit-ref> tag of the DD

When you use the DD to define the EntityManagerFactory references, define the `<persistence-unit-ref>` tag of the DD:

(a) <persistence-unit-ref-name> tag

In the `<persistence-unit-ref-name>` tag, specify the lookup name with which the application code will look up EntityManagerFactory. The specified lookup name is the relative path from `java:comp/env`. The lookup name of EntityManagerFactory is not mandatory, but the JPA specifications recommend that the name be set under `java:comp/env/persistence`.

For the `<persistence-unit-ref>` tag, the same tags are used that are used as the other tags in *8.7.1 Method of injecting EntityManagerFactory in the application*. However, when EntityManagerFactory is obtained with the JNDI lookup, you cannot specify `<injection-target>` tag. An example of defining `<persistence-unit-ref>` in `web.xml` is as follows:

```
...
<web-app>
 ...
 <servlet>
    <display-name>InventoryManagerServlet</display-name>
    <servlet-name>InventoryManagerServlet</servlet-name>
    <servlet-class>com.hitachi.InventoryManagerServlet</servlet-class>
 </servlet>
 ...
 <persistence-unit-ref>
   <description>
    Persistence unit for the inventory management application.
   </description>
   <persistence-unit-ref-name>
    persistence/InventoryAppDB
   </persistence-unit-ref-name>
 <persistence-unit-name>InventoryManagement</persistence-unit-name>
  </persistence-unit-ref>
  ...
</web-app>
...
```

8.7.3 Overriding the @PersistenceUnit definition using the DD

If the `<persistence-unit-ref>` tag is defined in the DD when `@PersistenceUnit` is mentioned in the application, the contents defined in the annotation are overwritten by the contents defined in the DD. In this case, the mapping between the annotations and DD is determined with the mapping between the `name` attribute of `@PersistenceUnit` and the `<persistence-unit-ref-name>` tag present under the `<persistence-unit-ref>` tag of the DD. Note that even if the `name` attribute is not explicitly specified in `@PersistenceUnit`, the `name` attribute contains the default value.

The precautions to be taken when the attributes specified in `@PersistenceUnit` are overridden with the DD tags are as follows:

(1) <persistence-unit-name> and unitName attribute

The `<persistence-unit-name>` tag of the DD overrides the `unitName` attribute of `@PersistenceUnit`. Normally, if you change the persistence unit name, the application stops operating, so take care when you make changes.

(2) <injection-target> tag

The injection target cannot be overridden with the DD. Note that when the `<injection-target>` tag is coded in the DD, you must accurately specify the fields and methods in which `@PersistenceContext` is added.

8.8 Definitions in persistence.xml

You define the persistence unit information by using the `<persistence-unit>` tag of `persistence.xml`. This section describes the attributes of the `<persistence-unit>` tag and the tags specified under the `<persistence-unit>` tag.

Note that the space and linefeed characters added at the beginning and end of values specified in the attributes of the `<persistence-unit>` tag and in the tags specified under the `<persistence-unit>` tag are ignored.

For details on the tags in `persistence.xml`, see *13.2 persistence.xml*.

8.8.1 Attributes specified in the <persistence-unit> tag

In the `<persistence-unit>` tag, you specify the `name` attribute and `transaction-type` attribute.

(1) name attribute

You specify the name of the persistence unit to be defined. The name specified here is referenced from the `unitName` attribute of `@PersistenceUnit` or `@PersistenceContext` in the case of annotations. Also, in the case of the DD, the name specified here is referenced from the `<persistence-unit-name>` tag under the `<persistence-context-ref>` tag or under the `<persistence-unit-ref>` tag.

You cannot omit the `name` attribute. Also, when the JPA is used with Application Server, you cannot specify null in the `name` attribute. Specify a string of at least 1 character.

(2) transaction-type attribute

In the persistence unit to be defined, you specify whether the JTA will control the transaction or whether the application will control the transaction by using `javax.persistence.EntityTransaction`.

- **When the transaction is controlled by the JTA**
  Specify `JTA` in the `transaction-type` attribute. When you specify `JTA`, you must also specify the `<jta-data-source>` tag at the same time.
- **When the application controls the transaction by using EntityTransaction**
  Specify `RESOURCE_LOCAL` in the `transaction-type` attribute. When you specify `RESOURCE_LOCAL`, you must also specify the `<non-jta-data-source>` tag at the same time.

Note that if the `transaction-type` attribute is omitted, the default value is `JTA`.

8.8.2 Tags specified under the <persistence-unit> tag

The tags listed in the following table are specified under the `<persistence-unit>` tag.

Table 8–7: Tags specified under the <persistence-unit> tag

Specified tags	Settings
`<description>` tag	Describes the persistence unit.

Specified tags	Settings
`<provider>` tag	Specifies the JPA provider used.
`<jta-data-source>` tag `<non-jta-data-source>` tag	Specifies the JTA data source or non-JTA data source used in the JPA provider.
`<mapping-file>` tag	Specifies the O/R mapping file used.
`<jar-file>` tag	Specifies the name of the JAR file containing the `entity` class, `embeddable` class, and `mappedsuper` class.
`<class>` tag	Specifies the `entity` class, `embeddable` class, and `mappedsuper` class.
`<exclude-unlisted-classes>` tag	Specifies whether to handle the class as a Persistence class.
`<properties>` tag	Specifies the JPA provider-specific properties.

The following points describe respective tags:

(1) <description> tag

The user can freely code the explanation about the persistence unit. The contents specified here do not affect the operations of the application. Note that you can omit this tag.

(2) <provider> tag

Specifies the JPA provider used in the persistence unit. You specify the implementation class name of the `javax.persistence.spi.PersistenceProvider` interface of the JPA provider with the fully qualified name containing the package name. You can omit this tag.

If this tag is omitted, the default JPA provider specified in the Easy Setup definition file is used. Also, when this tag is omitted and the default JPA provider is not specified in the Easy Setup definition file, Cosminexus JPA provider is used as the JPA provider.

If the application depends on a specific JPA provider functionality and behavior, make sure to specify the `<provider>` tag.

Tip

To specify the default JPA provider in the Easy Setup definition file, specify `ejbserver.jpa.defaultProviderClassName` in the `<param-name>` tag of the logical J2EE server and specify the name of the default JPA provider class in the `<param-value>` tag.

Note that if the `ejbserver.jpa.overrideProvider` parameter is specified in the `<param-name>` tag of the logical J2EE server in the Easy Setup definition file, the name of the JPA provider class specified in the `<param-value>` tag of the `ejbserver.jpa.overrideProvider` parameter is used on a higher priority than the value specified in the `<provider>` tag and the `ejbserver.jpa.defaultProviderClassName` parameter.

For details on the parameters to be specified in the Easy Setup definition file, see *12.2.1 Parameters used for setting up the user properties for the J2EE server*.

The following table describes the priority for determining the JPA provider used in the persistence unit.

Table 8–8: Priority for determining the JPA provider used in the persistence unit

Priority	JPA provider to be used
1	Value specified in `ejbserver.jpa.overrideProvider` property of the Easy Setup definition file
2	Value specified in the `<provider>` tag of `persistence.xml`
3	Cosminexus JPA provider (The JPA provider can also be changed by using the `ejbserver.jpa.defaultProviderClassName` parameter of the Easy Setup definition file)

(3) <jta-data-source> tag and <non-jta-data-source> tag

Specifies the JTA data source or non-JTA data source used by the JPA provider. The value specified here is product-dependent as per the JPA specifications, but define the data source references with Application Server as follows:

- **When referencing a resource adapter conforming to Connector 1.0**
  Specify the "*display-name-of-resource-adapter*" or "*optional-name-of-resource-adapter*".
- **When referencing a resource adapter conforming to Connector 1.5**
  Specify the "*display-name-of-resource-adapter* ! *connection-definition-identifier*" or "*optional-name-of-resource-adapter*".

The specified value is interpreted as "*display-name-of-resource-adapter*" or "*display-name-of-resource-adapter* ! *connection-definition-identifier*" and the relevant resource adapter is searched. If the relevant resource adapter does not exist, the specified value is interpreted as "*optional-name-of-resource-adapter*" and the relevant resource adapter is searched.

The resource adapter to be referenced must be deployed as a J2EE resource adapter (method of deploying the resource adapter as a standalone module). Start the resource adapter before starting the application containing the persistence unit.

You can omit the `<jta-data-source>` tag and `<non-jta-data-source>` tag. If the tags are omitted, the value specified in the `ejbserver.jpa.defaultJtaDsName` parameter or `ejbserver.jpa.defaultNonJtaDsName` parameter of the Easy Setup definition file is used. However, these properties do not have default values.

If a value is specified in the `ejbserver.jpa.overrideJtaDsName` parameter or `ejbserver.jpa.overrideNonJtaDsName` parameter, this value is used on higher priority than the value specified in the `<jta-data-source>` tag and `<non-jta-data-source>` tag and the value specified in the `ejbserver.jpa.defaultJtaDsName` parameter and `ejbserver.jpa.defaultNonJtaDsName` parameter.

You must specify `LocalTransaction` or `XATransaction` in the transaction support level of the resource adapter specified in the `<jta-data-source>` tag. You must also specify `NoTransaction` in the transaction support level of the resource adapter specified in `<non-jta-data-source>`.

The following table lists the priority for determining the JTA data source and non-JTA data source used in the persistence unit.

Table 8–9: Priority for determining the JTA data source and non-JTA data source used in the persistence unit

Priority	JPA provider to be used
1	Value specified in the `ejbserver.jpa.overrideJtaDsName` property or `ejbserver.jpa.overrideNonJtaDsName` property of the Easy Setup definition file
2	Value specified in the `<jta-data-source>` or `<non-jta-data-source>` element of `persistence.xml`
3	Value specified in the `ejbserver.jpa.defaultJtaDsName` property or `ejbserver.jpa.defaultNonJtaDsName` property of the Easy Setup definition file

Important note

If a resource adapter display name containing characters other than one-byte alphanumeric characters and underscore (_) is specified in the `<jta-data-source>` tag or `<non-jta-data-source>` tag of `persistence.xml`, replace that character with an underscore (_). However, if the replaced display name is duplicated with another resource adapter display name or optional name, note that the persistence unit might perform operations using an unintended data source.

(4) <mapping-file>, <jar-file>, <class>, and <exclude-unlisted-classes> tag

The following two methods are available for specifying the entity class, embeddable class, mapped superclass included in the persistence unit:

1. Method specified explicitly by using the O/R mapping file and the `<class>` tag
2. Method that is not specified explicitly and uses auto-search by JPA provider

A description of method 1 is as follows:

(a) Specifying the classes using the O/R mapping file

If an XML file named `orm.xml` is allocated under `META-INF` at the persistence unit root or under `META-INF` in another JAR file referenced with the `<jar-file>` tag from the persistence unit, the file is automatically handled as an O/R mapping file even if the file is not specified in the `<mapping-file>` tag. Furthermore, if the name of the XML file that can be loaded on the class path is specified in the `<mapping-file>` tag, that XML file is also handled as an O/R mapping file. If one persistence unit contains multiple O/R mapping files, the mapping information is read from all the O/R mapping files. However, the operations for duplicate mapping defined between multiple O/R mapping files are not provided.

(b) Specifying the JAR file that searches the persistence class

In the `<jar-file>` tag, you can specify the JAR file containing the persistence class and O/R mapping file. From the JAR file specified in the `<jar-file>` tag, the class in which `@Entity`, `@Embeddable`, and `@MappedSuperclass` are added is searched and the mapping information is obtained automatically. If `META-INF/orm.xml` exists in the specified JAR file, the mapping information is also obtained from `orm.xml`. Note that you can also use the `<jar-file>` tag and the `<mapping-file>` tag together.

The JAR files that can be specified in the `<jar-file>` tag must be included in the class path. The following JAR files can be specified:

- JAR file placed in the EAR root
- JAR file placed in the library directory of the EAR file
- EJB-JAR
- JAR file placed in `WEB-INF/lib` in the WAR

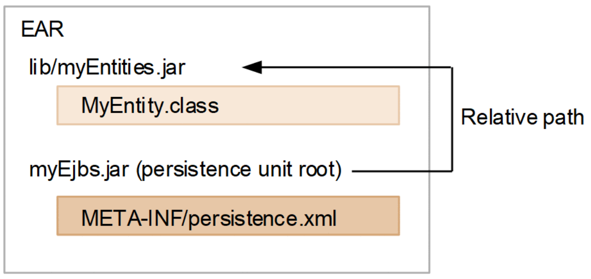
However, you cannot specify the JAR file placed in `WEB-INF/lib` in the WAR in the `<jar-file>` tag of the persistence unit defined in the EAR level or EJB-JAR level. This is because the JAR file placed in `WEB-INF/lib` in the WAR is loaded using the WEB application class loader and, therefore, can only be referenced from the components in the WAR.

Also, you can only specify the JAR file included in the same WAR from the `<jar-file>` tag of the persistence unit defined in the WAR level. This is because the classes included in the JAR file allocated to a location other than the same WAR might be loaded using the class loader before the deployment of the persistence unit defined in the WAR level. In such cases, the conversion of the byte code by the JPA provider might not be performed correctly.

In the `<jar-file>` tag, specify the relative path from the persistence unit root to the JAR file. The following is an example of specification:

Example 1

The specification of the relative path shown in the following figure is described below:

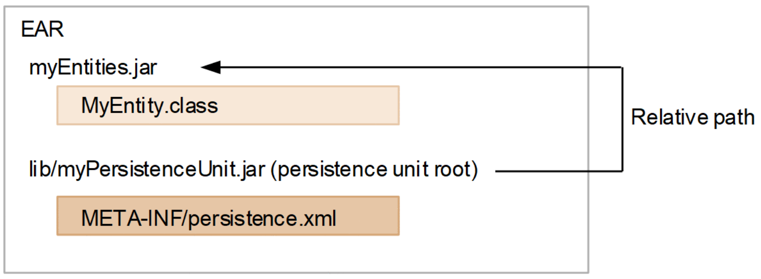


- The EAR lib contains `myEntities.jar` that stores the entity class.
- Specify `myEntities.jar` in the `<jar-file>` tag of `META-INF/persistence.xml` of EJB-JAR placed in the EAR root.

In this figure, the persistence unit root becomes EJB-JAR, so specify the relative path from EJB-JAR to `myEntities.jar`. The relative path is `lib/myEntities.jar`. Therefore, specify `lib/myEntities.jar` in the `<jar-file>` tag.

Example 2

The specification of the relative path shown in the following figure is described below:

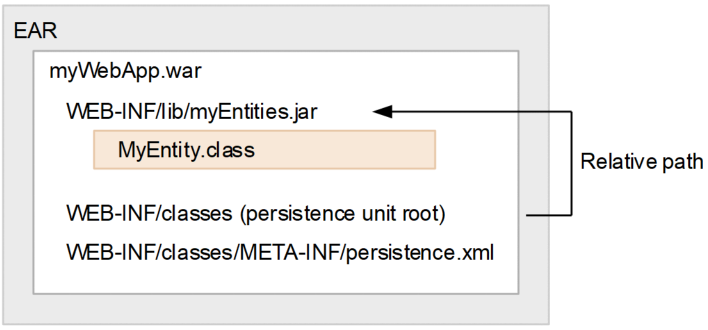


- The EAR root contains `myEntities.jar` that stores the entity class.
- Specify `myEntities.jar` in the `<jar-file>` tag of `META-INF/persistence.xml` of `lib/myPersistenceUnit.jar` in EAR.

In this figure, the persistence unit root becomes `myPersistenceUnit.jar`, so specify the relative path from `myPersistenceUnit.jar` to `myEntities.jar`. The relative path is `../myEntities.jar`. Therefore, specify `../myEntities.jar` in the `<jar-file>` tag.

Example 3

The specification of the relative path shown in the following figure is described below:


EAR
myWebApp.war
WEB-INF/lib/myEntities.jar
MyEntity.class
WEB-INF/classes (persistence unit root)
WEB-INF/classes/META-INF/persistence.xml
Relative path

- `WEB-INF/lib` of the WAR contains `myEntities.jar` that stores the entity class.
- Specify `myEntities.jar` in the `<jar-file>` tag of `WEB-INF/classes/META-INF/persistence.xml` of the WAR.

In this figure, the persistence unit root becomes `WEB-INF/classes` of the WAR, so specify the relative path from `WEB-INF/classes` to `myEntities.jar`. The relative path is `../lib/myEntities.jar`. Therefore, specify `../lib/myEntities.jar` in the `<jar-file>` tag.

(c) Specifying the persistence class list explicitly

If you use the `<class>` tag, you can explicitly specify the persistence class list. The mapping information is obtained from the annotation added in the specified class. Note that you can also use the `<class>` tag in combination with the `<mapping-file>` tag and `<jar-file>` tag.

(d) Allocating the persistence class with the added annotation to the persistence unit root

From the persistence unit root, the persistence class in which `@Entity`, `@Embeddable`, and `@MappedSuperclass` are added is automatically searched. The mapping information is obtained from the annotation added in the class. If you do not want to add the annotation-added class, which is allocated to the persistence unit root, in the persistence unit, you must specify the `<exclude-unlisted-classes>` tag beforehand.

(5) <properties> tag

You can specify the vendor-specific properties of the JPA provider. If you specify properties that cannot be understood by the JPA provider, the properties are ignored. Note that you cannot specify properties beginning with `javax.persistence` in `<properties>`.

If you specify properties with property names beginning with the prefix `ejbserver.jpa.emfprop.` as the system properties, the properties with the prefix removed are added in the persistence unit properties.

8.9 Allocating persistence.xml

You allocate `persistence.xml` in the EJB-JAR, WAR, or EAR. Make sure that you place `persistence.xml` under `META-INF`. The path with `persistence.xml` under `META-INF` is called the *persistence unit root*.

The locations in which `persistence.xml` can be allocated are as follows:

- `META-INF/persistence.xml` of EJB-JAR
- `WEB-INF/classes/META-INF/persistence.xml` of the WAR
- `META-INF/persistence.xml` in the `jar` file placed under `WEB-INF/lib` of the WAR
- `META-INF/persistence.xml` in the `jar` file placed in the EAR root directory
- `META-INF/persistence.xml` in the `jar` file placed in the EAR library directory

You can define multiple persistence units in one `persistence.xml`. You must name the persistence unit. However, you cannot define multiple persistence units with duplicated names in one EJB-JAR, WAR, or EAR.

The class managed with the persistence unit defined in EAR is loaded using the application class loader and can be referenced from all the components in the application.

Furthermore, when the same entity class is referenced from the components in different EJB-JAR and WAR, the referenced class is the same unique class even if the persistence units are different.

8.10 JPA interfaces

This section introduces the JPA interfaces and describes the `javax.persistence.EntityManager` interface and `javax.persistence.EntityManagerFactory` interface.

8.10.1 javax.persistence.EntityManager interface

This section describes the interface definition and the notes on the `javax.persistence.EntityManager` interface.

(1) Definition of the interface

```
package javax.persistence;

/**
* Interface for operating the persistence context.
*
* EntityManager instance is associated with the persistence context.
* The persistence context is a set of entity instances and
* the entity instance is unique for each
* perpetuated entity.
* The entity instances and their life cycles
* are managed in the persistence context.
* The EntityManager interface defines the methods for operating the * persis
tence context and is used for
* creating or deleting the perpetuated entity instances,
* searching the entity based on the primary key, and for executing
* the entity query.
*
* Define the set of entities that can be managed by EntityManager
* using the persistence unit.
* The persistence unit defines the group of entity classes used
* by the application and also defines the mapping between the entity
* classes and the database.
*/
public interface EntityManager {

 /**
 * The instance is set to managed and is perpetuated.
 * @param entity instance of the persistent entity
 * @throws EntityExistsException When the entity already exists
 * (When persist method is invoked, EntityExistsException is thrown
 * or EntityExistsException or another PersistenceException
 * is thrown during flush or commit)
 * @throws IllegalArgumentException When the argument is not an entity
 * @throws TransactionRequiredException When the container-managed
 * EntityManager specifying PersistenceContextType.TRANSACTION
 * is invoked when the transaction does not exist
 */
 public void persist(Object entity) ;

 /**
 * The entity status is merged with the current persistence context
 * @param entity Entity
 * @return instance where the status is merged with the persistence context
 * @throws IllegalArgumentException When the instance is not an entity
 * or is a removed entity
 * @throws TransactionRequiredException When the container-managed
 * EntityManager specifying PersistenceContextType.TRANSACTION
 * is invoked when the transaction does not exist.
 */
 public <T> T merge(T entity) ;

 /**
 * entity instance is deleted.
 * @param entity Entity
 * @throws IllegalArgumentException Instance is not an entity
 * or is a detached entity
 * @throws TransactionRequiredException When the container-managed
 * EntityManager specifying PersistenceContextType.TRANSACTION
 * is invoked when the transaction does not exist.
 */
 public void remove(Object entity) ;

 /**
 * Searches the primary key.
 * @param entityClass Entity class
 * @param primaryKey Primary key
 * @return Instance of the searched entity
 * null when the entity does not exist
 * @throws IllegalArgumentException When the entityClass argument
 * is not an entity type or if the primaryKey argument is not the
 * valid type as the primary key of that entity
 */
 public <T> T find(Class<T> entityClass, Object primaryKey) ;

 /**
 * The status obtains the delayed fetch instance.
 * When the requested entity does not exist in the database,
 * When the instance status is accessed for the first time
 * EntityNotFoundException is thrown.
 * (when getReference is invoked, the JPA provider is also allowed
 * to throw EntityNotFoundException)
 * If the application does not access the instance while
 * Entity Manager is open, do not expect that the instance status
 * can be accessed during detach
 * @param entityClass Entity class
 * @param primaryKey Primary key
 * @return Instance of the searched entity
 * @throws IllegalArgumentException When the entityClass argument
 * is not an entity type or the primaryKey argument is not a
 * valid type as the primary key of that entity
 * @throws EntityNotFoundException When the entity status cannot
 * be accessed
 */
 public <T> T getReference(Class<T> entityClass, Object primaryKey) ;

 /**
 * The persistence context status is flushed in the database.
 * @throws TransactionRequiredException When the transaction does
 * not exist
 * @throws PersistenceException When flush fails
```

```
*/
public void flush() ;

/**
* Specifies the flush mode applied to all the objects
* included in the persistence context.
* @param flushMode Flush mode
*/
public void setFlushMode(FlushModeType flushMode) ;

/**
* Obtains the flush mode applied to all the objects
* included in the persistence context.
* @return flushMode Flush mode
*/
public FlushModeType getFlushMode() ;

/**
* Sets the lock mode of the entity objects
* included in the persistence context.
* @param entity Entity
* @param lockMode Lock mode
* @throws PersistenceException When an unsupported
* lock invocation is performed
* @throws IllegalArgumentException When the instance is not an entity
* or is a detached entity
* @throws TransactionRequiredException When the transaction
* does not exist
*/
public void lock(Object entity, LockModeType lockMode) ;

/**
* Instance status is refreshed to the database status.
* If the instance status is changed, the status is
* overwritten by the database status.
* @param entity Entity
* @throws IllegalArgumentException When the argument is not an entity
* or does not have the managed status
* @throws TransactionRequiredException When the container-managed
* EntityManager specifying PersistenceContextType.TRANSACTION
* is invoked when the transaction does not exist
* @throws EntityNotFoundException When the entity already
* does not exist in the database
*/
public void refresh(Object entity) ;

/**
* Clears the persistence context and all the managed entities
* are detached.
* If the entity has changes that were not flushed in the database
* the entity is not perpetuated.
*/
public void clear() ;

/**
* Checks if the instance is included in the current
* persistence context.
* @param entity Entity
```

```
 * @return true if included
 * @throws IllegalArgumentException When the argument is not an entity
 */
 public boolean contains(Object entity) ;

 /**
 * Creates a Query instance for executing
* JPQL(Java Persistence Query language) statement
 * @param qlString Java Persistence Query statement
 * @return New Query instance
 * @throws IllegalArgumentException When the query statement is not valid
 */
 public Query createQuery(String qlString) ;

 /**
 * Creates a Query instance for executing the named query
* (JPQL or native SQL).
 * @param name Name of the query defined in the Meta data
 * @return New Query instance
 * @throws IllegalArgumentException If the query with the specified
 * name is not defined.
 */
 public Query createNamedQuery(String name) ;

 /**
 * Creates a Query instance for executing the native SQL statement
* (update statement and delete statement)
 * @param sqlString Native SQL statement
 * @return New Query instance
 */
 public Query createNativeQuery(String sqlString) ;

 /**
 * Creates a Query instance for executing the native SQL query.
 * @param sqlString Native SQL statement
 * @param resultClass Class of the instance that forms the
 * return value
 * @return New Query instance
 */
 public Query createNativeQuery(String sqlString, Class result-
 Class) ;

 /**
 * Creates a Query instance for executing the native SQL query.
 * @param sqlString Native SQL statement
 * @param resultSetMapping Result set mapping name
 * @return New Query instance
 */
 public Query createNativeQuery(String sqlString, String result-
 SetMapping) ;

 /**
 * Reports EntityManager that the JTA transaction is active.
 * This method is invoked by the application
 * to associate the application-managed JTA entity manager that was
 * created outside the transaction scope
 * with the current JTA transaction.
 * @throws TransactionRequiredException When the transaction
 * does not exist
 */
 public void joinTransaction() ;

 /**
 * Returned when the lower provider objects exist.
 * The return value of this method depends on the implementation,
 * but when the container-managed EntityManager is used in
 * Cosminexus Component Container,
 * the EntityManager object of the JPA provider is returned.
 * @return EntityManager object of the JPA provider
 * /
 public Object getDelegate() ;

 /**
 * Closes the application-managed EntityManager.
 * After the close method is invoked, all the methods other than
 * getTransaction and isOpen (returning false) of the Query object
 * obtained from EntityManager instance and EntityManager
 * throw IllegalStateException.
 * If this method is invoked when EntityManager
 * is associated with an active transaction, the persistence context
 * continues to exist until the transaction is concluded.
 * @throws IllegalStateException For the container-managed
 * EntityManager
 */
 public void close() ;

 /**
 * Returns whether EntityManager is open.
 * @return Returns true until EntityManager closes.
 */
 public boolean isOpen() ;

 /**
 * Returns the resource level transaction object.
 * Used by the EntityTransaction instance to start and commit
 * multiple transactions serially.
 * @return EntityTransaction instance
 * @throws IllegalStateException When this method is invoked with
 * the JTA entity manager
 */
 public EntityTransaction getTransaction() ;

}
```

(2) Notes

- When the transaction scope persistence context is used, the `persist`, `merge`, `remove`, and `refresh` methods must be invoked in the transaction context. If the transaction context does not exist, `javax.persistence.TransactionRequiredException` is thrown.
- The `find` and `getReference` methods can also be invoked outside the transaction context. When the transaction scope persistence context is used, the resulting entity has a detached status. When the extended persistence context is used, the resulting entity has a managed status.

- The `Query` and `EntityTransaction` objects obtained from EntityManager can be used while EntityManager is open.
- If the argument of the `createQuery` method is not a valid JPQL (Java Persistence Query Language), `IllegalArgumentException` is thrown or the execution of the query fails. If the native query is incorrect and if the definition of the result set is not compatible with the query result, `PersistenceException` is thrown when the query is executed and the execution of the query fails. When possible, `PersistenceException` wraps the database exceptions.
- If a runtime exception is thrown in the method of the `EntityManager` interface, the transaction is rolled back.
- The `close`, `isOpen`, `joinTransaction`, and `getTransaction` methods are used for managing the application-managed EntityManager.
- With the EJB specifications, the following methods can invoke the EntityManager methods in the Stateless Session Bean:
  - Business method of the business interface or component interface
  - Business method
  - Interceptor method
  - Timeout callback method

  The EntityManager methods cannot be invoked with the constructor, the `setter` method (including `setSessionContext` method) of the DI, and the life cycle callback method (`PostConstruct` and `PreDestroy`). If the EntityManager methods are invoked in the unpermitted locations, the KDJE56538-E message is displayed and `java.lang.IllegalStateException` is thrown.
- With the EJB specifications, the following methods can invoke the EntityManager methods in the Stateful Session Bean:
  - Life cycle callback method (`PostConstruct` and `PreDestroy`)
  - Business method of the business interface or component interface
  - Business method
  - Interceptor method
  - `afterBegin` and `beforeCompletion` methods of `SessionSynchronization`

  The EntityManager methods cannot be invoked with the constructor, the `setter` method of the DI, and `afterCompletion` method of `SessionSynchronization`. If the EntityManager methods are invoked in unpermitted locations, the KDJE56538-E message is displayed and `java.lang.IllegalStateException` is thrown.

Note the following points as well when you use the JPA with Application Server:

- EntityManager that the application obtains by using inject, JNDI lookup, and EntityManagerFactory becomes the proxy class of EntityManager provided by Application Server and not the `EntityManager` object provided by the JPA provider. If you use the `getDelegate()` method of this proxy class, you can obtain the `EntityManager` object provided by the JPA provider. However, when the container-managed EntityManager is used, the container manages the EntityManager life cycle, so do not invoke the `close()` method of the `EntityManager` object obtained by using the `getDelegate()` method, from the application.
- If the `getDelegate()` method of EntityManager is invoked when the transaction scope persistence context is used and when transaction does not exist, the container cannot close the returned EntityManager. In this case, you must invoke `EntityManager.close()` with the application.

8.10.2 javax.persistence.EntityManagerFactory interface

This section describes the interface definition and the notes on the `javax.persistence.EntityManagerFactory` interface.

(1) Definition of the interface

```
package javax.persistence;

/**
* The EntityManagerFactory interface is used
* to obtain the application-managed EntityManager by the application.
* When the application ends the use of EntityManagerFactory,
* the application must close EntityManagerFactory.
* After EntityManagerFactory is closed,
* all EntityManagers created from EntityManagerFactory are
* treated as closed.
*/
public interface EntityManagerFactory {
 /**
 * Creates a new EntityManager.
 * Whenever this method is invoked, a new EntityManager instance
 * is returned.
 * The isOpen method of EntityManager returned by this method returns
 * true.
 * @return New EntityManager
 */
 public EntityManager createEntityManager() ;

 /**
 * A new EntityManager is created by using the specified property
 * map.
 * Whenever this method is invoked, a new EntityManager instance
 * is returned.
 * The isOpen method of EntityManager returned by this method returns
 * true.
 * @param map Map storing the EntityManager properties
 * @return New EntityManager
 */
 public EntityManager createEntityManager(Map map) ;

 /**
 * Closes factory and releases all the resources
 * stored in factory.
 * After factory is closed, if method other than isOpen is invoked,
 * IllegalStateException is thrown. The isOpen method returns
 * false.
 * If EntityManagerFactory is closed, all EntityManagers created
 * from factory are also treated as closed.
 */
 public void close() ;

 /**
 * Returns whether factory is open.
 * @return True until factory is closed
 */
 public boolean isOpen() ;
}
```

(2) Notes

- You can include vendor-specific properties of the JPA provider in the map passed to `createEntityManager`. The properties that cannot be recognized by the JPA provider are ignored.
- With the EJB specifications, the following methods can invoke the EntityManagerFactory methods in the Stateless Session Bean:
    - Life cycle callback method (`PostConstruct` and `PreDestroy`)
    - Business method of the business interface or component interface
    - Business method
    - Interceptor method
    - Timeout callback method

  The EntityManagerFactory methods cannot be invoked with the constructor and the `setter` method (including the `setSessionContext` method) of the DI.
- With the EJB specifications, the following methods can invoke the EntityManagerFactory methods in the Stateful Session Bean:
    - Life cycle callback method (`PostConstruct` and `PreDestroy`)
    - Business method of business interface or component interface
    - Business method
    - Interceptor method
    - The `afterBegin` and `beforeCompletion` methods of `SessionSynchronization`.

  The EntityManagerFactory methods cannot be invoked with the constructor, the `setter` method of the dependency injection, and the `afterCompletion` method of `SessionSynchronization`.

8.11 Notes on setting up applications

This section describes the notes on setting up the applications running on Application Server and using the JPA.

8.11.1 Notes on allocating the entity classes

With Application Server, package the entity classes in the EARs, EJB-JARs, or WARs, at the locations decided in the JPA specifications. Do not add the entity classes in the system class path.

When an entity class is loaded with the application class loader or the Web application class loader, the byte code of the class is converted using the JPA provider in order to implement operations such as Lazy fetch. If the entity class is included in the system class path, the byte code conversion does not work because the entity class is loaded with the system class loader. Therefore, the JPA provider does not operate properly.

8.11.2 Reference scope of the persistence unit name

The components, such as the EJBs and servlets, included in the application, reference the used persistence unit by specifying the persistence unit name in the `unitName` attribute of `@PersistenceUnit` and `@PersistenceContext` and in the `<persistence-unit-name>` tag under the `<persistence-context-ref>` tag or under the `<persistence-unit-ref>` tag defined in the DD. However, the persistence unit scope that can be referenced from each component is as follows:

- The persistence units defined in the EJB-JARs or WARs can be referenced from the components included in the EJB-JARs or WARs.
- The persistence unit defined in the EAR file can be referenced from all the components included in the EAR file.

If the name of the persistence unit defined in the EAR file and the name of the persistence unit defined in the EJB-JARs or WARs are duplicated, from the components in the EJB-JAR or WAR, the persistence units defined with a narrower scope are given priority. For example, if persistence units with the same name are defined in the EAR and WAR files, from the components included in the WAR file, the persistence unit defined in the WAR file is visible on priority. The persistence unit existing in the EAR file is not visible.

(1) Persistence unit used when the persistence unit name is omitted

If you omit the persistence unit name referenced by the components when the JPA is used with Application Server, the persistence unit used is determined with the following rules:

- If only one persistence unit is defined in the EJB-JAR or WAR that contains components, that persistence unit is used.
- If persistence unit is not defined in the EJB-JAR or WAR that contains components and only one persistence unit is defined in the EAR, the persistence unit in the EAR is used.

Note that if the following conditions are fulfilled, one persistence unit cannot be identified, so the persistence unit name cannot be omitted:

- When two or more persistence units are defined in the EJB-JAR or WAR that contains components
- If persistence unit is not defined in the EJB-JAR or WAR that contains components and two or more persistence units are defined in the EAR

(2) Explicitly referencing the EAR-level persistence unit using # syntax

If the name of the persistence unit defined in the EAR and the name of the persistence unit defined in the EJB-JAR or WAR are duplicated, from the components in the EJB-JAR or WAR, the persistence units defined with a narrower scope are visible on priority. However, by using # syntax in the reference name of the persistence unit, you can explicitly reference the persistence unit defined in the EAR. When you use # syntax, specify the persistence unit name as follows:

Relative-path-from-the-EJB-JAR-or-WAR-containing-components-to-the-persistence-unit-root#*Name-of-the-persistence-unit*

For example, when the persistence unit `myPersistenceUnit` included in `lib/persistenceUnitRoot.jar` of the EAR file is referenced from the EJB included in `ejbs/myEjbs.jar` of the EAR file, the referenced persistence unit name is `../lib/persistenceUnitRoot.jar#myPersistenceUnit`.

8.11.3 Items checked when the application is deployed

The following items are checked when an application using the JPA is deployed on the J2EE server:

- Checking the persistence unit definitions
- Checking the EntityManager and EntityManagerFactory references

The following is a description of the checks:

(1) Checking of the persistence unit definition

The following points describe the contents checked in the persistence unit definition:

(a) Validation of persistence.xml

Validates whether `persistence.xml` included in the application is in accordance with the `persistence_1_0.xsd` schema. If an error is found during the validation, the error message KDJE56526-E is output and the deployment is cancelled.

(b) Checking of the persistence unit name

Checks whether the persistence unit name is null. If the persistence unit name is null, the error message KDJE56505-E is output and the deployment is cancelled.

Also, checks whether persistence units with duplicated names are defined in the application EAR files or in one EJB-JAR or WAR. If persistence units with duplicated names are defined, a warning message KDJE56500-W is output and the deployment continues. Note that in this case, only one persistence unit is actually deployed.

(c) Checking whether the data source referenced by the persistence unit exists

The contents checked differ depending on the transaction type of the persistence unit.

- **When the transaction type of the persistence unit is the JTA**
  The following table lists the contents checked and the operations to be performed when an error occurs during the check.

Table 8–10: Checked contents and operations to be performed when an error occurs during the check (for the JTA transaction type)

Checked contents	Operations when an error occurs during the check
The JTA data source (the data source specified in the `<jta-data-source>` tag of `persistence.xml` or the default value specified in the system properties) used in the persistence unit is specified.	The error message KDJE56527-E is output and the deployment is cancelled.
The resource adapter that provides the JTA data source exists.	The error message KDJE56529-E is output and the deployment is cancelled.
The resource adapter that provides the JTA data source is running.	
The transaction support level of the resource adapter is `LocalTransaction` or `XATransaction`.	The error message KDJE56531-E is output and the deployment is cancelled.
The connection factory interface of the resource adapter is `javax.sql.DataSource`.	The error message KDJE56533-E is output and the deployment is cancelled.

- **When the transaction type of the persistence unit is RESOURCE_LOCAL**
  The following table lists the contents checked and the operations to be performed when an error occurs during the check.

Table 8–11: Checked contents and operations to be performed when an error occurs during the check (for the RESOURCE_LOCAL transaction type)

Checked contents	Operations when an error occurs during the check
The non-JTA data source (the data source specified in the `<non-jta-data-source>` tag of `persistence.xml` or the default value specified in the system properties) used in the persistence unit is specified.	The error message KDJE56528-E is output and the deployment is cancelled.
The resource adapter that provides the non-JTA data source exists.	The error message KDJE56530-E is output and the deployment is cancelled.
The resource adapter that provides the non-JTA data source is running.	
The transaction support level of the resource adapter is `NoTransaction`.	The error message KDJE56532-E is output and the deployment is cancelled.
The connection factory interface of the resource adapter is `javax.sql.DataSource`.	The error message KDJE56533-E is output and the deployment is cancelled.

(d) Checking the provider class specified in the persistence unit

Checks whether the JPA provider class (class specified in the `<provider>` tag of `persistence.xml`) used by the persistence unit can be loaded from the application. If the loading of the class fails, the error message KDJE56503-E is output and the deployment is cancelled.

(e) Checking whether the JAR file referenced by the persistence unit exists

Checks whether the JAR file (JAR file specified in the `<jar-file>` tag of `persistence.xml`) referenced by the persistence unit exists. If the JAR file does not exist, the error message KDJE56506-E is output and the deployment is cancelled.

(f) Checking the contents defined in the persistence unit using the JPA provider

The JPA provider generates the persistence unit from the contents of `persistence.xml` parsed by the JPA container. If the persistence unit definition has a problem and if the JPA provider returns an error, the error message (KDJE56539-E) is displayed and the deployment is cancelled.

(2) Checking the EntityManager and EntityManagerFactory references

The following points describe the contents checked in the EntityManager and EntityManagerFactory references:

(a) Checking the existence of the persistence unit

In the EntityManager and EntityManagerFactory references defined in the EJB and Web components, check whether the specified persistence unit name is an actually referable persistence unit name. Also, if the persistence unit name is omitted, check whether the persistence unit to be used can be identified. If an error is found in this check, the error message KDJE56501-E is output and the deployment is cancelled.

(b) Checking the transaction type for the container-managed EntityManager

Check whether the transaction type of the persistence unit is the JTA when the container-managed EntityManager is used. If an error occurs in this check, the error message KDJE56534-E is output and the deployment is cancelled.

(c) Checking if the extended persistence context is used from outside the Stateful Session Bean

When `EXTENDED` is specified as the persistence context type with the EntityManager references, check whether the location where the references are defined is the Stateful Session Bean. If the references are defined in a location other than the Stateful Session Beans, the error message KDJE56535-E is output and the deployment is cancelled.

8.11.4 Notes on using the JPA with Application Server

The EntityManager type that the application can obtain is the proxy class of EntityManager provided by Application Server and not the `EntityManager` object provided by the JPA provider.

The `EntityManager` object can be obtained by using injection, JNDI lookup, or EntityManagerFactory.

Note that to use injection for obtaining EntityManager, you set up `javax.persistence.EntityManager` as the field or method argument type that injects EntityManager.

Also, you cannot cast EntityManager obtained with injection, JNDI lookup, or EntityManagerFactory in the EntityManager implementation class of the JPA provider.

If you need to obtain the `EntityManager` object of the JPA provider, use the `getDelegate` method of the EntityManager proxy object obtained with injection, JNDI lookup, or EntityManagerFactory.

8.11.5 Notes when the Cosminexus JPA functionality is not used

With the Cosminexus JPA functionality, when `persistence.xml` is included in an application, `persistence.xml` is read by default regardless of whether the Cosminexus JPA functionality is used. If `persistence.xml` cannot be interpreted with Application Server, the application fails to start.

To avoid this situation, when `persistence.xml` is included in the application and you do not want to use the Cosminexus JPA functionality, specify `true` in the `ejbserver.jpa.disable` property of the J2EE server.

As a result, `persistence.xml` is no longer read with Application Server.

Also, regardless of the value specified in the `ejbserver.jpa.disable` property, the JPA version that can be used with Application Server is JPA 1.0.

8.12 javax.persistence package

This section describes the list of annotations included in the `javax.persistence` package and the precautions to be taken when specifying annotations.

You can also specify the mapping information in an O/R mapping file instead of the annotations. For details on the correspondence between the annotations and the O/R mapping files, see *8.12.64 Correspondence between the annotations and O/R mapping*.

Precautions when specifying an annotation

- With the Cosminexus JPA, the annotations included in the `javax.persistence` package are not supported in the attributes related to the DDL output functionality.
- When specifying the same column name more than once in an annotation, arrange the upper case and lower case characters.
- If field names or method names are allocated in the column name, character strings are considered as upper case characters strings and used with Cosminexus JPA. If you want to specify a column name in the supported annotation, use upper case characters.
- The access type is decided according to the location at which the annotation is provided. However, if the access type exists in both, the field and property, the settings of the field will be enabled.
- The property name is decided as follows depending on the character string acquired by removing `get` or `set` (`is`) from the access method:

  - If the first two characters are in upper case, the string is used as it is.

  - If the first two characters are not in upper case, the first character is converted into lower case, and the string is used.

  - For a single character, the first character is converted into lower case, and the string is used.

List of annotations

Annotation classification	Annotation name	Overview
Entity annotation	`@Entity`	Indicates that the class is an entity.
Annotations related to the tables or columns	`@Column`	Specifies the mapping between the persistence field or the persistence property, and the columns of the database.
	`@JoinColumn`	Specifies the external key column for the binding table or a column of the binding-destination table that is referenced from the external key column by correlating the entity classes.
	`@JoinColumns`	Used when multiple `@JoinColumn` are coded concurrently.
	`@JoinTable`	This annotation specifies the binding table set up in the following classes: • Owner side class when `ManyToMany` relationship is specified. • Class with single-sided `OneToMany` relationship.
	`@PrimaryKeyJoinColumn`	Specifies the column used as the external key, when binding with other tables.
	`@PrimaryKeyJoinColumns`	Used when multiple `@PrimaryKeyJoinColumn` are coded concurrently.
	`@SecondaryTable`	Specifies a secondary table in the entity class.

Annotation classification	Annotation name	Overview
	@SecondaryTables	Used when multiple @SecondaryTable are coded concurrently.
	@Table	Specifies a primary table in the entity class.
	@UniqueConstraint	If you want to generate CREATE statement for the primary table or secondary table, include the unique constraints, and then specify. Note that this annotation is not supported with Cosminexus JPA provider CJPA provider.
Annotations related to the ID	@EmbeddedId	Specifies the compound primary key of a class that can be embedded.
	@GeneratedValue	Specifies the method for automatically generating and allotting a unique value to the primary key column.
	@Id	Specifies the properties or fields of the primary key of the entity class.
	@IdClass	Specifies the compound primary key class mapped to multiple fields or properties of the entity class.
	@SequenceGenerator	Specifies the settings of the sequence generator that creates the primary key.
	@TableGenerator	Specifies the settings of the generator that creates the primary key.
Lock annotation	@Version	Specifies the version field or the version property for using the optimistic lock functionality.
Annotations related to mapping	@Basic	Indicates the type of mapping to the simplest database column.
	@Embeddable	Specifies an embedded class.
	@Embedded	Specifies the persistence property or the persistence field indicating the instance value of the embedded class within the entity class at the embedding destination.
	@Enumerated	Specifies the persistence field or the persistence property as the enumeration type.
	@Lob	Specifies the persistence field or the persistence property of the large object type supported by the database.
	@MapKey	Specifies the map key used for object identification within the map, when a non-owner entity class is indicated by the java.util.Map type in the OneToMany relationship or the ManyToMany relationship.
	@OrderBy	Specifies the order in which the collection is evaluated when the entity information is acquired.
	@Temporal	Specifies in the persistence property or persistence field having the type that expresses the time (java.util.Date and java.util.Calendar).
	@Transient	Specifies the field or property of a non-persisting entity class, mapped superclass, or embedded class.

Annotation classification	Annotation name	Overview
Annotations related to the relationship	`@ManyToMany`	Indicates that the specified class has a `ManyToMany` relationship, and also specifies the multiple relationships from the owner entity class to the non-owner entity class.
	`@ManyToOne`	Indicates that the specified class has the `ManyToOne` relationship, and also specifies the relationship to the non-owner entity class.
	`@OneToMany`	Indicates that the specified class has the `OneToMany` relationship, and also specifies the multiple relationships from the owner entity class to the non-owner entity class.
	`@OneToOne`	Indicates that the specified class has the `OneToOne` relationship, and also specifies the single relationship between entity classes.
Annotations related to inheritance and overriding	`@AssociationOverride`	Overrides the settings used in the `ManyToOne` relationship or the `OneToOne` relationship specified in a mapped superclass and embedded class.
	`@AssociationOverrides`	Used when multiple `@AssociationOverride` are coded concurrently.
	`@AttributeOverride`	Overrides the following mapping information: • Properties or fields specified by `@Basic` (or applied by default) • Properties or fields specified by `@Id`
	`@AttributeOverrides`	Used when multiple `@AttributeOverride` are coded concurrently.
	`@DiscriminatorColumn`	Specifies the column used for identification in the SINGLE_TABLE strategy or JOINED strategy. This annotation is added to an entity class that becomes a superclass by inheriting an entity class.
	`@DiscriminatorValue`	Specifies the value of the column used for identification in the SINGLE_TABLE strategy or JOINED strategy.
	`@Inheritance`	Specifies the inheritance mapping strategy used in the entity class hierarchy.
	`@MappedSuperclass`	Specifies a mapped superclass.
Annotations related to queries	`@ColumnResult`	Specifies the column for mapping the query results of an SQL to the entity class.
	`@EntityResult`	Specifies the entity class in which the query results of the SQL are to be mapped.
	`@FieldResult`	Specifies the field in which the query results of the SQL are to be mapped.
	`@NamedNativeQueries`	Used when multiple `@NamedNativeQuery` are coded concurrently.
	`@NamedNativeQuery`	Specifies a named query in the SQL.
	`@NamedQueries`	Used when multiple `@NamedQuery` are coded concurrently.
	`@NamedQuery`	Specifies a named query of JPQL.

Annotation classification	Annotation name	Overview
	`@QueryHint`	Specifies a database-specific query hint.
	`@SqlResultSetMapping`	Specifies the result set mapping of an SQL query.
	`@SqlResultSetMappings`	Used when multiple `@SqlResultSetMapping` are coded concurrently.
Annotations related to event callback#	`@EntityListeners`	Specifies the callback listener class used in the entity class or mapped superclass.
	`@ExcludeDefaultListeners`	This annotation excludes the default listener for the following classes: • Entity class • Mapped superclass • Subclass of the entity class or mapped superclass
	`@ExcludeSuperclassListeners`	This annotation excludes the superclass listener for the following classes: • Entity class • Mapped superclass • Subclass of the entity class or mapped superclass
	`@PostLoad`	This annotation indicates the callback method invoked after the `SELECT` statement is issued in the database.
	`@PostPersist`	This annotation indicates the callback method invoked after the `INSERT` statement is issued in the database.
	`@PostRemove`	This annotation indicates the callback method invoked after the `DELETE` statement is issued in the database.
	`@PostUpdate`	This annotation indicates the callback method invoked after the `UPDATE` statement is issued in the database.
	`@PrePersist`	This annotation indicates the callback method invoked before the `INSERT` statement is issued in the database.
	`@PreRemove`	This annotation indicates the callback method invoked before the `DELETE` statement is issued in the database.
	`@PreUpdate`	This annotation indicates the callback method invoked before the `UPDATE` statement is issued in the database.
Annotations related to the reference of EntityManager and EntityManagerFactory	`@PersistenceContext`	Defines the container-managed EntityManager.
	`@PersistenceContexts`	Used when multiple `@PersistenceContext` are coded concurrently.
	`@PersistenceProperty`	Sets up properties in the container-managed EntityManager.
	`@PersistenceUnit`	Defines the persistence unit for the EntityManagerFactory.
	`@PersistenceUnits`	Used when multiple `@PersistenceUnit` are coded concurrently.

#

For details on the callback method, see *9.15 Procedure for specifying the callback method*.

8.12.1 @AssociationOverride

(1) Description

This annotation overrides the settings used in the `ManyToOne` relationship or the `OneToOne` relationship specified in a mapped superclass and an embedded class.

When `@AssociationOverride` is not specified, the external key column is mapped in the same way as the original mapping.

The applicable targets are class, method, and field.

(2) Element

The following table lists the elements of `@AssociationOverride`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name of the field or property having the related mapping that is to be overridden.
`joinColumns`	Required	This element specifies an array of `@JoinColumn`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

`String`

Description

This element specifies the name of the field or property having the related mapping that is to be overridden.

Default value

None

(b) joinColumns element

Type

`JoinColumn[]`

Description

This element specifies an array of `@JoinColumn`.

The definition of the mapped superclass or embedded class is applied as the mapping type.

You can specify the value within the specifiable range of the arrays of `@JoinColumn`. For details, see *8.12.24 @JoinColumn*.

Default value

None

8.12.2 @AssociationOverrides

(1) Description

This annotation is specified when multiple `@AssociationOverride` are coded concurrently.

The applicable targets are class, method, and field.

(2) Element

The following table lists the elements of `@AssociationOverrides`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@AssociationOverride`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type
    `AssociationOverride[]`

Description
    This element specifies an array of `@AssociationOverride`.
    You can specify the value within the specifiable range of the arrays of `@AssociationOverride`. For details, see *8.12.1 @AssociationOverride*.

Default value
    None

8.12.3 @AttributeOverride

(1) Description

This annotation overrides the following mapping information:

- Properties or fields specified by `@Basic` (applied by default)
- Properties or fields applied by default
- Properties or fields specified by `@Id`

To override the settings of `@Column` defined in the mapped superclass and embedded class, apply the field or property of the entity class and embedded class in which the mapped superclass is inherited.

If `@AttributeOverride` is not specified, the column is mapped with the original mapping before override.

If `@AttributeOverride` is defined in the entity class of a unit that does not have an inheritance relationship, the operation is performed; however, the operation cannot be guaranteed.

The applicable targets are class, method, and field.

(2) Element

The following table lists the `@AttributeOverride` attributes.

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name of the field or property in which the mapping is overridden.
`column`	Required	This element specifies the `@Column` to be overridden.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

`String`

Description

This element specifies the name of the field or property in which mapping is overridden.

Default value

None

(b) column element

Type

`Column`

Description

This element specifies the `@Column` to be overridden.

The definition of the embeddable class or mapped superclass is applied as the mapping type.

You can specify the value within the specifiable range of `@Column`. For details, see *8.12.6 @Column*.

Default value

None

8.12.4 @AttributeOverrides

(1) Description

The annotation to be specified when you want to code multiple `@AttributeOverride` concurrently.

Applicable elements are class, method, and field.

(2) Element

The following table lists the `@AttributeOverrides` attributes:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@AttributeOverride`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

`AttributeOverride[]`

Description

Attribute that specifies the array of `@AttributeOverride`.

Specifiable values are within the range of the specifiable values for `@AttributeOverride` array. For details, see *8.12.3 @AttributeOverride*.

Default value

None

8.12.5 @Basic

(1) Description

This annotation indicates the type of mapping to the simplest database column.

This annotation can be applied to the properties or instance variables of the following persistence types:

- Java primitive type
- Primitive type wrapper class
- `java.lang.String`
- `java.math.BigInteger`
- `java.math.BigDecimal`
- `java.util.Date`
- `java.util.Calendar`
- `java.sql.Date`
- `java.sql.Time`
- `java.sql.Timestamp`
- `byte[]`
- `Byte[]`
- `char[]`
- `Character[]`
- `enums`
- User-defined serialize type

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@Basic`:

Element name	Optional/Required	Element description
`fetch`	Optional	This element specifies the specification value of the fetch strategy.
`optional`	Optional	This element specifies whether or not a null value can be used in the field or property. Note that Cosminexus JPA provider does not support this attribute.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) fetch element

Type

`FetchType`

Description

This element specifies the specification value of the fetch strategy.

`FetchType.EAGER` or `FetchType.LAZY` can be specified.

Furthermore, the `fetch` attribute is ignored in Cosminexus JPA provider CJPA provider, and the default `FetchType.EAGER` is usually applied. For details on the `fetch` attribute, see *9.4.5 Synchronization with the database*.

Default value

`FetchType.EAGER`

8.12.6 @Column

(1) Description

This annotation specifies the mapping between the persistence field or persistence property, and the columns of the database.

Even when `@Column` is not specified explicitly in the persistence property or persistence field, the persistence property or persistence field is handled as if `@Column` were specified. In such a case, the default values will be applied in each element value of `@Column`.

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@Column`:

Element name	Optional/ Required	Element description
`name`	Optional	This element specifies the column name.
`unique`	Optional	This element specifies whether or not the property is a unique key. Note that Cosminexus JPA provider does not support this attribute.

Element name	Optional/ Required	Element description
`nullable`	Optional	This element specifies whether or not a null value can be specified in the database column. Note that Cosminexus JPA provider does not support this attribute.
`insertable`	Optional	This element specifies whether or not to include the column specified by `@Column` in the `INSERT` statement of the SQL.
`updatable`	Optional	This element specifies whether or not to include the column specified by `@Column` in the `UPDATE` statement of the SQL.
`columnDefinition`	Optional	This element is used to describe the constraints added to the column in the DDL, when the `CREATE` statement is output. Note that Cosminexus JPA provider does not support this attribute.
`table`	Optional	This element specifies the table name that includes the column.
`length`	Optional	This element specifies the length of a column. Note that Cosminexus JPA provider does not support this attribute.
`precision`	Optional	This element specifies the accuracy of a column. This element is specified when the column is `numeric` type. Note that Cosminexus JPA provider does not support this attribute.
`scale`	Optional	This element specifies the scale of a column. This element is specified when the column is `numeric` type. Note that Cosminexus JPA provider does not support this attribute.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

`String`

Description

This element specifies the column name.

The column name that can be specified depends on the database specifications.

Default value

Property name or field name in which this annotation is specified

(b) insertable element

Type

`boolean`

Description

This element specifies whether or not to include the column specified by `@Column` in the `INSERT` statement of the SQL. You can specify either `true` or `false`.

These values imply the following meaning:

`true`: The column specified by `@Column` is included in the `INSERT` statement of the SQL.

`false`: The column specified by `@Column` is not included in the `INSERT` statement of the SQL.

Default value

`true`

(c) updatable element

Type

`boolean`

Description

This element specifies whether or not to include the column specified by `@Column` in the `UPDATE` statement of the SQL. You can specify either `true` or `false`.

These values imply the following meaning:

`true`: The column specified by `@Column` is included in the `UPDATE` statement of the SQL.

`false`: The column specified by `@Column` is not included in the `UPDATE` statement of the SQL.

Default value

`true`

(d) table element

Type

`String`

Description

This element specifies the table name that includes the column.

The table name that can be specified depends on the database specifications.

Default value

Primary table name

8.12.7 @ColumnResult

(1) Description

This annotation specifies the column for mapping the query results of an SQL to the entity class

The applicable targets are the columns of `@SqlResultSetMapping`.

(2) Element

The following table lists the elements of `@ColumnResult`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name or optional name of the columns of `SELECT` clause.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

`String`

Description

This element specifies the name or optional name of the columns of `SELECT` clause.

The column name that can be specified depends on the database specifications.

Default value

None

8.12.8 @DiscriminatorColumn

(1) Description

This annotation specifies the column for identification used in the SINGLE_TABLE strategy or JOINED strategy. This annotation is added to an entity class that becomes a superclass by inheriting an entity class.

The applicable target is class.

(2) Element

The following table lists the elements of `@DiscriminatorColumn`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies the name of the column for identification.
`discriminatorType`	Optional	This element specifies the type of the column for identification.
`columnDefinition`	Optional	This element is used to describe the constraints added to the column for identification in the DDL, when the `CREATE` statement is output. Note that Cosminexus JPA provider does not support this attribute.
`length`	Optional	This element specifies the length when the column for identification is a character string. Note that Cosminexus JPA provider does not support this attribute.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

`String`

Description

This element specifies the name of the column for identification.

The column name that can be specified depends on the database specifications.

Specify the same value (matching the upper case and lower case characters) for the `name` element as that for the `name` element of `@Column`.

Default value

`"DTYPE"`

(b) discriminatorType element

Type

`DiscriminatorType`

Description

This element specifies the type of the column for identification.

You can specify the following values:

- `DiscriminatorType.STRING`
- `DiscriminatorType.CHAR`
- `DiscriminatorType.INTEGER`

Default value

`DiscriminatorType.STRING`

8.12.9 @DiscriminatorValue

(1) Description

This annotation specifies the value of the column for identification used in the SINGLE_TABLE strategy or JOINED strategy. You can specify this annotation in a superclass or subclass.

The applicable target is class.

Note the following points:

- The settings of `@DiscriminatorValue` are not inherited. `@DiscriminatorValue` must be set up in each entity class.
- The settings of `@DiscriminatorValue` must match the type specified in `discriminatorType` and length specified in `length` of `@DiscriminatorColumn`.
- If the `discriminatorType` of `@DiscriminatorColumn` is `INTEGER`, make note of the following points:
  - In `@DiscriminatorValue`, specify only an integer that does not include `0` or a blank at the beginning.
  - You cannot omit `@DiscriminatorValue`. If omitted, the operation will not be guaranteed.
- If the `discriminatorType` of `@DiscriminatorColumn` is other than `INTEGER`, you can omit `@DiscriminatorValue`. In such a case, the operation is performed by assuming that the value specified in `value` is the class name of the entity.

(2) Element

The following table lists the elements of `@DiscriminatorValue`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies the value to be set up in the column for identification.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

  `String`

Description

  This element specifies the value to be set up in the column for identification.

  The value that can be specified depends on the type of the column for identification.

Default value

  Entity name

8.12.10 @Embeddable

(1) Description

This annotation indicates an embedded class.

An embedded class is a class that can be embedded as a field within the entity class.

The applicable target is class.

(2) Element

`@Embeddable` does not have attributes.

8.12.11 @Embedded

(1) Description

This annotation specifies the persistence property or persistence field indicating the instance value of the embedded class within the entity class.

If you want to override the column mapping declared within the embedded class, use either `@AttributeOverride` or `@AttributeOverrides`.

The applicable targets are method and field.

(2) Element

`@Embedded` does not have attributes.

8.12.12 @EmbeddedId

(1) Description

This annotation specifies the compound primary key of an embedded class.

This annotation is added in the persistence property or persistence field of an embeddable class owned by the entity.

When using `@EmbeddedId`, you cannot specify multiple `@EmbeddedIds` or specify `@Id` besides `@EmbeddedId`.

When you add `@Transient` to a field of the embedded class, the compound primary key will not be applicable for that field.

The applicable targets are method and field.

(2) Element

`@EmbeddedId` does not have attributes.

8.12.13 @Entity

(1) Description

This annotation specifies that the class is an entity.

The class name of the entity class does not include the package name. Note the following points during specification:

- Make sure that the entity name is a unique name within the persistence unit.
- You cannot set up the reserved characters of JPQL. If you set up the reserved characters, the operation will not be guaranteed.

The applicable target is class.

(2) Element

The following table lists the elements of `@Entity`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies a logical name for the entity class. It becomes an abstract schema name in JPQL.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

`String`

Description

This element specifies a logical name for the entity class. It becomes an abstract schema name in JPQL.

The value that can be specified depends on the specifications of JPQL.

Default value

Class name of the class in which `@Entity` is specified

8.12.14 @EntityListeners

(1) Description

This annotation specifies the callback listener class used in the entity class or mapped superclass.

The applicable target is class.

(2) Element

The following table lists the elements of `@EntityListeners`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies the callback listener class.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type
    `Class[]`

Description
    This element specifies the callback listener class.
    The value that can be specified is class.

Default value
    None

8.12.15 @EntityResult

(1) Description

This annotation specifies the entity class in which the query results of the SQL are to be mapped.

The applicable target is the `entities` element of `@SqlResultSetMapping`.

(2) Element

The following table lists the elements of `@EntityResult`:

Element name	Optional/Required	Element description
`entityClass`	Required	This element specifies the result class.
`fields`	Optional	This element specifies the arrays of `@FieldResult`.
`discriminatorColumn`	Optional	This element specifies the name or optional name of the column for identification within the `SELECT` clause that determines the type of the entity instance.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) entityClass element

Type
: `Class`

Description
: This element specifies the result class.
: The value that can be specified is the class name.

Default value
: None

(b) fields element

Type
: `FieldResult[]`

Description
: This element specifies an array of `@FieldResult`.
: You can specify the value within the specifiable range of the arrays of `@FieldResult`. For details, see *8.12.19 @FieldResult*.

Default value
: Blank array

(c) discriminatorColumn element

Type
: `String`

Description
: This element specifies the name or optional name of the column for identification within the `SELECT` clause that determines the type of the entity instance.
: The value that can be specified is the name or optional name of the column specified in the table.

Default value
: Blank array

8.12.16 @Enumerated

(1) Description

This annotation specifies the persistence field or persistence property as the enumeration type.

This annotation can be used along with `@Basic`. You can specify `ORDINAL` (numeric type) and `STRING` (character string type) in the enumeration type.

In the following cases, `ORDINAL` (numeric type) is specified as the enumeration type:

- When the enumeration type is not specified in the `value` element
- When `@Enumerated` is not specified

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@Enumerated`:

Element name	Optional/Required	Element description
`value`	Optional	This element specifies the type used for mapping the enumeration type.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

`EnumType`

Description

This element specifies the type used for mapping the enumeration type.

You can specify either of the following values:

- `EnumType.ORDINAL`: Numeric type
- `EnumType.STRING`: Character string type

Default value

`EnumType.ORDINAL`

8.12.17 @ExcludeDefaultListeners

(1) Description

This annotation excludes the default listener for the following classes:

- Entity class
- Mapped superclass
- Subclass of the entity class or mapped superclass

Note that the default listener can be specified only in the XML descriptor.

The applicable target is class.

(2) Element

`@ExcludeDefaultListeners` does not have attributes.

8.12.18 @ExcludeSuperclassListeners

(1) Description

This annotation excludes the superclass listener for the following classes:

- Entity class
- Mapped superclass
- Subclass of the entity class or mapped superclass

The applicable target is class.

(2) Element

`@ExcludeSuperclassListeners` does not have attributes.

8.12.19 @FieldResult

(1) Description

This annotation specifies the field in which the query results of the SQL are to be mapped.

The applicable target is the `field` element of `@EntityResult`.

(2) Element

The following table lists the elements of `@FieldResult`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name of the persistence field or persistence property of the class.
`column`	Required	This element specifies the name or optional name of the column of `SELECT` clause.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type
: `String`

Description
: This element specifies the name of the persistence field or persistence property of the class.

Default value
: None

(b) column element

Type

`String`

Description

This element specifies the name or optional name of the column of `SELECT` clause.

The column name or optional name that can be specified depends on the database specifications.

Default value

None

8.12.20 @GeneratedValue

(1) Description

This annotation specifies the method for automatically generating and allotting a unique value to the primary key column. This annotation is applicable to the field or property of the primary key of entity class or mapped superclass containing `@Id`.

The primary key value is generated by the following four methods. Depending on the generation method selected, the base table and database sequence object must be prepared beforehand. For details on each of the generation methods, see the description about the *strategy element*.

- `GenerationType.AUTO`
- `GenerationType.IDENTITY`
- `GenerationType.SEQUENCE`
- `GenerationType.TABLE`

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@GeneratedValue`:

Element name	Optional/Required	Element description
`strategy`	Optional	This element specifies the method for generating the primary key value of the entity class.
`generator`	Optional	This element specifies the `name` element set up in `@SequenceGenerator` or `@TableGenerator` to be used.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) strategy element

Type

`GenerationType`

Description

This element specifies the method for generating the primary key value of the entity class.

The following four types of values can be specified:

- **GenerationType.AUTO**
  For generating the primary key value, select the most appropriate procedure in each database.
  When Oracle or HiRDB is used as the database, the processing is same as `GenerationType.TABLE`.
- **GenerationType.IDENTITY**
  The primary key value is generated using the `identity` column of the database.
  If Oracle is used as the database, the processing is same as `GenerationType.SEQUENCE`.
  If HiRDB is used as the database, the processing is same as `GenerationType.TABLE`.
- **GenerationType.SEQUENCE**
  The primary key value is generated using the database sequence object.
  If HiRDB is used as the database, the processing is same as `GenerationType.TABLE`.
- **GenerationType.TABLE**
  The primary key value is generated using a table for maintaining the primary key value.

Default value

`GenerationType.AUTO`

(b) generator element

Type

`String`

Description

This element specifies the `name` element set up in `@SequenceGenerator` or `@TableGenerator` to be used.

Default value

The following names are assumed depending on the value of the strategy element:

- **In the case of GenerationType.AUTO**
  `"SEQ_GEN"`
- **In the case of GenerationType.SEQUENCE**
  `"SEQ_GEN_SEQUENCE"`
- **In the case of GenerationType.TABLE**
  `"SEQ_GEN_TABLE"`

8.12.21 @Id

(1) Description

This annotation specifies the properties or fields of the primary key of entity class.

`@Id` is applicable in the entity class or mapped superclass.

The column of the database mapped to the field or property in which `@Id` is specified is assumed as the primary key column of the primary table. When the column name of the primary key column is not specified using `@Column`, the column name of the primary key column becomes the name of the field or property in which `@Id` is specified.

Note that if `@Version` is specified in a field in which `@Id` is specified, `@Id` becomes invalid.

The applicable targets are method and field.

(2) Element

`@Id` does not have attributes.

8.12.22 @IdClass

(1) Description

This annotation specifies the compound primary key class mapped to multiple fields or properties of the entity class.

This annotation is applicable to the mapped superclass or entity class.

The name and type of the field or property of the compound primary key class must match with that of the field or property of the primary key of entity class. The name and type specified in this annotation must correspond to the name and type of the property or field of primary key of the entity in which `@Id` is added.

The applicable target is class.

(2) Element

The following table lists the elements of `@IdClass`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies the compound primary key class.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

`Class`

Description

This element specifies the compound primary key class.

The value that can be specified is the class name.

Default value

None

8.12.23 @Inheritance

(1) Description

This annotation specifies the inheritance mapping strategy used in the inheritance hierarchy of an entity.

`@Inheritance` is specified in the parent entity class of inheritance hierarchy.

The following are two types of inheritance mapping strategy available with Cosminexus JPA provider:

- **SINGLE_TABLE** (single table for each class hierarchy)
- **JOINED (**binding subclass strategy**)**

For details on the inheritance mapping strategy, see *9.13.2 Inheritance mapping strategy*.

The applicable target is class.

(2) Element

The following table lists the elements of `@Inheritance`:

Element name	Optional/Required	Element description
`strategy`	Optional	This element specifies the type of inheritance mapping strategy.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) strategy element

Type

`InheritanceType`

Description

This element specifies the type of inheritance mapping strategy used in an entity.

The following two types of values can be specified:

- `InheritanceType.SINGLE_TABLE`: This strategy is used to map all classes in the inheritance hierarchy to a single table.
- `InheritanceType.JOINED`: This strategy is used to map the top most (parent class) of the inheritance hierarchy to a single table, and map each subclass with a subclass-specific mapping.

Default value

`InheritanceType.SINGLE_TABLE`

8.12.24 @JoinColumn

(1) Description

This annotation specifies the external key column for the binding table, or the column name of the binding-destination table that is referenced from the external key column, when the entity classes are correlated. Always specify the column that acts as the primary key of the binding-destination table.

When multiple external key columns exist, use @JoinColumns, and specify @JoinColumn for each relation. When multiple @JoinColumn are specified, specify the name element and referencedColumnName element in each annotation.

When @JoinColumn is not specified explicitly, it is assumed that a single external key column is specified in the persistence property or persistence field that specifies the relation. Also, the default value is applied to each element value of @JoinColumn.

Furthermore, if the changes made in a single column of the field and the changes made by correlating the cascade operation are performed concurrently, consistency might not be achieved. Therefore, when the column specified in the name element and the referencedColumnName element is defined in a field of the entity, the insertable element and the updatable element must be set to false. With this, only the changes made in the field will be applied to the database, but the changes made due to the correlation of the cascade operation will not be applied to the database.

The applicable targets are method and field.

(2) Element

The following table lists the elements of @JoinColumn:

Element name	Optional/Required	Element description
name	Optional	This element specifies the external key column name used to bind the target tables.
referencedColumnName	Optional	This element specifies the column name of the binding-destination table that is referenced from the external key column specified in the name element.
unique	Optional	This element specifies whether or not the property is a unique key. Note that Cosminexus JPA provider does not support this attribute.
nullable	Optional	This element specifies whether or not a null value can be specified in the database column. Note that Cosminexus JPA provider does not support this attribute.
insertable	Optional	This element specifies whether or not to include the column specified by @JoinColumn in the INSERT statement of the SQL.
updatable	Optional	This element specifies whether or not to include the column specified by @JoinColumn in the UPDATE statement of the SQL.
columnDefinition	Optional	This element is used to describe the constraints added to the external column in the DDL, when the CREATE statement is output. Note that Cosminexus JPA provider does not support this attribute.
table	Optional	This element specifies the table name that includes the external key column.

The details of attributes that are supported in Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the external key column name used to bind the target tables.

The location of existence of the external key column is different for each type of the entity relationship. The location of existence of the external key column for each type of the entity relationship is as follows:

- **In the case of OneToOne relationship or ManyToOne relationship**

Within the local entity table

- **In the case of ManyToMany relationship**

  Within the binding table of `@JoinTable`

The values that can be specified depend on the specifications of the database column name.

Default value

- **When a single external key column is specified in the local entity, and nothing is specified in the value of the name element**

  name-of-the-related-property-or-field-within-the-local-entity_name-of-the-referenced-primary-key-column

- **When the related property and field that is being referenced does not exist (example: when @JoinTable is used)**

  name-of-the-referenced-entity_name-of-the-referenced-primary-key-column

(b) referencedColumnName element

Type

`String`

Description

This element specifies the column name of the binding-destination table that is referenced by the external key column specified in the `name` element.

The column name of the binding-destination table exists at the following locations:

- **When the relationship annotation is used**

  Within the referenced table

- **When @JoinTable is used**

  Within the entity table of the owner entity

Note

When binding is defined as a part of reverse binding, the location will be within the table of the non-owner entity class.

The column names that can be specified depend on the database specifications.

Default value

Column name of the primary key of the table referenced from the external key

Note

If a single external key column is specified, the default value will be applied.

(c) insertable element

Type

`boolean`

Description

This element specifies whether or not to include the column specified by `@JoinColumn` in the `INSERT` statement of the SQL. You can specify either `true` or `false`.

These values imply the following meaning:

`true`: The column specified by `@JoinColumn` is included in the `INSERT` statement of the SQL.

`false`: The column specified by `@JoinColumn` is not included in the `INSERT` statement of the SQL.

Default value

`true`

(d) updatable element

Type

`boolean`

Description

This element specifies whether or not to include the column specified by `@JoinColumn` in the `UPDATE` statement of the SQL. You can specify either `true` or `false`.

These values imply the following meaning:

`true`: The column specified by `@JoinColumn` is included in the `UPDATE` statement of the SQL.

`false`: The column specified by `@JoinColumn` is not included in the `UPDATE` statement of the SQL.

Default value

`true`

(e) table element

Type

`String`

Description

This element specifies the table name that includes the external key column.

The table name that can be specified depends on the database specifications.

Default value

Primary table name

8.12.25 @JoinColumns

(1) Description

This annotation is specified when multiple `@JoinColumn` that indicate the same relationship are coded concurrently. `@JoinColumns` defines the mapping of the compound external key.

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@JoinColumns`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@JoinColumn`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type
: `JoinColumn[]`

Description
: This element specifies an array of `@JoinColumn`.
: The values can be specified within the specifiable range of the arrays of `@JoinColumn`.

Default value
: None

8.12.26 @JoinTable

(1) Description

This annotation specifies the binding table set up in the following classes:

- Owner class when the `ManyToMany` relationship is specified
- Class containing a single-direction `OneToMany` relationship

When the `name` element is not specified, the name of the binding table becomes as follows:

owner-table-name_non-owner-table-name

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@JoinTable`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies the name of the binding table.
`catalog`	Optional	This element specifies the catalog name of the binding table. Note that Cosminexus JPA provider does not support this attribute.
`schema`	Optional	This element specifies the schema name of the binding table.
`joinColumns`	Optional	This element specifies the external key column of the binding table that references the primary table of the owner entity.
`inverseJoinColumns`	Optional	This element specifies the external key column of the binding table that references the primary table of the non-owner entity.
`uniqueConstraints`	Optional	This element specifies the unique constraints of the table. Note that Cosminexus JPA provider does not support this attribute.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

String

Description

This element specifies the name of the binding table.

The table name that can be specified depends on the database specifications.

Default value

owner-table-name_non-owner-table-name

(b) schema element

Type

String

Description

This element specifies the schema name of the table.

The value that can be specified depends on the specifications of the database schema name.

Default value

Default schema of the database used

(c) joinColumns element

Type

JoinColumn[]

Description

This element specifies the external key column of the binding table that references the primary table of the owner entity. This element specifies an array of `@JoinColumn`. The external key column name of the binding table is specified in the `name` element, while the referenced column name of the owner is specified in the `referencedColumnName` element of `@JoinColumn`.

The column names that can be specified depend on the database specifications.

Default value

External key of `@JoinColumn`

(d) inverseJoinColumns element

Type

JoinColumn[]

Description

This element specifies the external key column of the binding table that references the primary table of the non-owner entity. This element specifies an array of `@JoinColumn`. The external key column name of the binding table is specified in the `name` element, while the column of the binding-destination table that is referenced by the external key column is specified in the `referencedColumnName` element of `@JoinColumn`.

The values that can be specified depend on the specifications of the database column name.

Default value

External key column of `@JoinColumn`

8.12.27 @Lob

(1) Description

This annotation specifies the persistence field or persistence property of the `large` object type supported by the database. This annotation can be used together with `@Basic`.

`@Lob` contains the binary type (`Blob`) and character type (`Clob`). The type of `@Lob` is determined based on the type of the persistence field or persistence property. For a character string and character type, the type of `@Lob` is `Clob`, and in other cases, the type is `Blob`.

The applicable targets are method and field.

(2) Element

`@Lob` does not have attributes.

8.12.28 @ManyToMany

(1) Description

This annotation specifies the multiple relationships from an owner entity class having a `ManyToMany` relationship to a non-owner entity class.

The `ManyToMany` relationship includes the owner and non-owner, irrespective of bi-direction or single direction. If the relationship is bi-directional, the binding table can be specified in any direction.

If the `Collection` element type is specified using `Generics`, the non-owner entity class is not required to be specified. In other cases, make sure to specify it.

Furthermore, when you specify `@ManyToMany`, note the settings of the following annotations:

- The elements of the same annotations for `@OneToMany` are same as that of `@ManyToMany`.
- If the properties or fields in which `@ManyToMany` is defined have the same name in the owner and non-owner classes, do not use the default settings (when the `joinColumns` element and `inverseJoinColumns` element is not specified) of `@JoinTable`.
- For the bi-directional relationship, the value of the binding table is updated based on the information of the owner. Even if the mapping information is changed in the entity class in which the `mappedBy` element is specified, the information will not be applied in the binding table.

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@ManyToMany`:

Element name	Optional/Required	Element description
`targetEntity`	Optional	This element specifies the non-owner entity class.
`cascade`	Optional	This element specifies the operations to be cascaded.

Element name	Optional/Required	Element description
`fetch`	Optional	This element specifies the specification value of the fetch strategy.
`mappedBy`	Optional	This element specifies the name of the field or property that maintains a relationship in the owner entity class, when added in the elements of the non-owner entity class.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) targetEntity element

Type

`Class`

Description

This element specifies the non-owner entity class.

The specification of this element is optional when the collection property is defined using `Generics`. In other cases, you must always specify this element.

Default value

The type in which the collection contains parameters

Note: Set up only when the collection property is defined using `Generics`.

(b) cascade element

Type

`CascadeType[]`

Description

This element specifies the operations to be cascaded.

The following table describes the specifiable values:

- `CascadeType.ALL`: The persist, remove, merge, and refresh operations of the owner entity class are cascaded to the related destination.
- `CascadeType.MERGE`: The merge operation of the owner entity class is cascaded to the related destination.
- `CascadeType.PERSIST`: The persist operation of the owner entity class is cascaded to the related destination.
- `CascadeType.REFRESH`: The refresh operation of the owner entity class is cascaded to the related destination.
- `CascadeType.REMOVE`: The remove operation of the owner entity class is cascaded to the related destination.

Default value

Not to be cascaded

(c) fetch element

Type

`FetchType`

Description

This attribute defines the fetch strategy of data from the database. For details on the fetch strategy, see *9.4.5 Synchronization with the database*.

The following two types of values can be specified:

- `EAGER` strategy**:** Requests in which the data must be fetched eagerly
- `LAZY` strategy**:** Requests in which data is fetched lazily when accessed for the first time

Default value
: `FetchType.LAZY`

(d) mappedBy element

Type
: `String`

Description
: This element specifies the name of the field or property that maintains a relationship in the owner entity class, when added in the elements of the non-owner entity class.
: When this element is specified, the relationship becomes bi-directional. For a bi-directional relationship, the value of the binding table is updated based on the information of the owner. Even when the mapping information is changed in the non-owner entity class (the entity class in which the `mappedBy` element is specified), the information will not be applied in the binding table.

Default value
: None

8.12.29 @ManyToOne

(1) Description

This annotation indicates that the class in which `@ManyToOne` is specified has a `ManyToOne` relationship, and also specifies the relationship from the owner entity class to the non-owner entity class.

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@ManyToOne`:

Element name	Optional/Required	Element description
`targetEntity`	Optional	This element specifies the non-owner entity class.
`cascade`	Optional	This element specifies the operations to be cascaded.
`fetch`	Optional	This element specifies the specification value of the fetch strategy.
`optional`	Optional	This element specifies whether or not a null value can be set up for all non-primitive type fields and properties.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) targetEntity element

Type
: `Class`

Description

This element specifies the non-owner entity class.

Default value

Type of the field and property in which the annotation is added

(b) cascade element

Type

`CascadeType[]`

Description

This element specifies the operations to be cascaded.

The following table describes the specifiable values:

- `CascadeType.ALL`: The persist, remove, merge, and refresh operations of the owner entity class are cascaded to the related destination.
- `CascadeType.MERGE`: The merge operation of the owner entity class is cascaded to the related destination.
- `CascadeType.PERSIST`: The persist operation of the owner entity class is cascaded to the related destination.
- `CascadeType.REFRESH`: The refresh operation of the owner entity class is cascaded to the related destination.
- `CascadeType.REMOVE`: The remove operation of the owner entity class is cascaded to the related destination.

Default value

Not to be cascaded

(c) fetch element

Type

`FetchType`

Description

This attribute defines the fetch strategy of data from the database. For details on the fetch strategy, see *9.4.5 Synchronization with the database*.

The following two types of values can be specified:

- `EAGER` strategy: Requests in which the data must be fetched eagerly
- `LAZY` strategy: Requests in which data is fetched lazily when accessed for the first time

Default value

`FetchType.EAGER`

(d) optional element

Type

`boolean`

Description

This element specifies whether or not a null value can be set up for all non-primitive type fields and properties. The following values can be specified:

- `true`: A null value can be set up for all non-primitive type fields and properties.

- `false`: A null value cannot be specified for all non-primitive type fields and properties.

Default value
  `true`

8.12.30 @MapKey

(1) Description

This annotation specifies the map key used for object identification within the map when the non-owner entity class is indicated by the `java.util.Map` type, in the `OneToMany` relationship or `ManyToMany` relationship.

When the `name` element is not specified, the primary key of the correlated entity is used as the map key.

If mapping is done as `@IdClass` when the primary key is a compound primary key, the compound primary key is used as the map key.

If a persistence field or persistence property other than the primary key is used as the map key, the unique key constraints related to the map key can be included.

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@MapKey`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies the name of the persistence field or persistence property of the non-owner entity class that is used as the map key.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type
  `String`

Description
  This element specifies the name of the persistence field or persistence property of the non-owner entity class that is used as the map key.

Default value
  Name of the primary key field or property of the non-owner entity class

8.12.31 @MappedSuperclass

(1) Description

This annotation specifies a mapped superclass.

A mapped superclass is used for inheritance. Hence, there are no tables corresponding to this class. Except for mapping to a subclass, and the inheritance of the related mapping information, the mapped superclass is mapped to the table in the same way as an entity.

The mapping information can be overridden in the subclass using `@AttributeOverride`.

The applicable target is class.

(2) Element

`@MappedSuperclass` does not have attributes.

8.12.32 @NamedNativeQueries

(1) Description

This annotation is specified when multiple `@NamedNativeQuery` are coded concurrently.

The applicable target is class.

(2) Element

The following table lists the elements of `@NamedNativeQueries`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@NamedNativeQuery`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

`NamedNativeQuery[]`

Description

This element specifies an array of `@NamedNativeQuery`.

The values can be specified within the specifiable range of the arrays of `@NamedNativeQuery`. For details, see *8.12.33 @NamedNativeQuery*.

Default value

None

8.12.33 @NamedNativeQuery

(1) Description

This annotation specifies a named query in the SQL. This annotation can be applied to an entity class and mapped superclass.

The applicable target is class.

(2) Element

The following table lists the elements of `@NamedNativeQuery`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name of the named query.
`query`	Required	This element specifies the SQL string.
`hints`	Optional	This element specifies an array of `@QueryHint`.
`resultClass`	Optional	This element specifies the class in which the SQL results are applied.
`resultSetMapping`	Optional	This element specifies the name indicated in the `name` element of `@SqlResultSetMapping`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

`String`

Description

This element specifies the name of the named query.

The value that can be specified is a character string.

Default value

None

(b) query element

Type

`String`

Description

This element specifies the SQL string.

The SQL that can be specified depends on the specifications of the database used.

Default value

None

(c) hints element

Type

`QueryHint[]`

Description

This element specifies an array of `@QueryHint`.

You can specify the value within the specifiable range of the arrays of `@QueryHint`. For details, see *8.12.53 @QueryHint*.

Default value

Blank array

(d) resultClass element

Type

`Class`

Description

This element specifies the class in which the SQL results are applied.

The `resultClass` element is specified when the class, in which you want to map the execution results of the query, exists. Do not specify the `resultClass` element and `resultSetMapping` element concurrently.

The value that can be specified is the class name.

Default value

`void.class`

(e) resultSetMapping element

Type

`String`

Description

This element specifies the name indicated in the `name` element of `@SqlResultSetMapping` in which the result set is defined.

This element is specified when the SQL results are to be mapped to any result set.

Do not specify the `resultClass` element and `resultSetMapping` element concurrently.

You can specify the value within the specifiable range of the `name` element of `@SqlResultSetMapping`. For details, see *8.12.57(2)(a) name element*.

Default value

Null character string

8.12.34 @NamedQueries

(1) Description

This annotation is specified when multiple `@NamedQuery` are coded concurrently.

The applicable target is class.

(2) Element

The following table lists the elements of `@NamedQueries`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@NamedQuery`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type
    `NamedQuery[]`

Description
    This element specifies an array of `@NamedQuery`.
    You can specify the value within the specifiable range of the arrays of `@NamedQuery`. For details, see *8.12.35 @NamedQuery*.

Default value
    None

8.12.35 @NamedQuery

(1) Description

This annotation specifies a named query of JPQL. This annotation can be applied to an entity class and mapped superclass.

The applicable target is class.

(2) Element

The following table lists the elements of `@NamedQuery`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name of the named query.
`query`	Required	This element specifies the query string of JPQL.
`hints`	Optional	This element specifies an array of `@QueryHint`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type
    `String`

Description
    This element specifies the name of the named query.
    The value that can be specified is a character string.

Default value
    None

(b) query element

Type
    `String`

Description

This element specifies the query string of JPQL.

The value that can be specified depends on the specifications of JPQL.

Default value

None

(c) hints element

Type

`QueryHint[]`

Description

This element specifies an array of `@QueryHint`.

You can specify the value within the specifiable range of the arrays of `@QueryHint`. For details, see *8.12.53 @QueryHint*.

Default value

Blank array

8.12.36 @OneToMany

(1) Description

This annotation specifies the multiple relationships from an owner entity class having the `OneToMany` relationship to a non-owner entity class.

The elements of the same annotations for `@OneToMany` are same as that of `@ManyToMany`.

If the `Collection` element type is specified using `Generics`, the non-owner entity class is not required to be specified. In other cases, make sure to specify it.

Furthermore, to achieve a bi-directional relationship, always specify the `mappedBy` element at the non-owner side.

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@OneToMany`:

Element name	Optional/Required	Element description
`targetEntity`	Optional	This element specifies the non-owner entity class.
`cascade`	Optional	This element specifies the operations to be cascaded.
`fetch`	Optional	This element specifies the specification value of the fetch strategy.
`mappedBy`	Optional	This element specifies the name of the field or property that maintains a relationship in the owner entity class, when added in the elements of the non-owner entity class.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) targetEntity element

Type

`Class`

Description

This element specifies the non-owner entity class.

The specification of this element is optional when a collection property is defined using `Generics`. In other cases, you must always specify this element.

Default value

The type in which the collection contains parameters

Note: Set up only when the collection property is defined using `Generics`.

(b) cascade element

Type

`CascadeType[]`

Description

This element specifies the operations to be cascaded.

The following table describes the specifiable values:

- `CascadeType.ALL`: The persist, remove, merge, and refresh operations of the owner entity class are cascaded to the related destination.
- `CascadeType.MERGE`: The merge operation of the owner entity class is cascaded to the related destination.
- `CascadeType.PERSIST`: The persist operation of the owner entity class is cascaded to the related destination.
- `CascadeType.REFRESH`: The refresh operation of the owner entity class is cascaded to the related destination.
- `CascadeType.REMOVE`: The remove operation of the owner entity class is cascaded to the related destination.

Default value

Not to be cascaded

(c) fetch element

Type

`FetchType`

Description

This attribute defines the fetch strategy of data from database. For details on the fetch strategy, see *9.4.5 Synchronization with the database*.

The following two types of values can be specified:

- `EAGER` strategy: Requests in which the data must be fetched eagerly
- `LAZY` strategy: Requests in which data is fetched lazily when accessed for the first time

Default value

`FetchType.LAZY`

(d) mappedBy element

Type

  `String`

Description

  This element specifies the name of the field or property that maintains a relationship in the owner entity class, when added in the elements of the non-owner entity class.

  When this element is specified, the relationship becomes bi-directional.

Default value

  None

8.12.37 @OneToOne

(1) Description

This annotation indicates that the specified class has `OneToOne` relationship, and also specifies the single relationship between entity classes.

Furthermore, to achieve a bi-directional relationship, always specify the `mappedBy` element at the non-owner side.

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@OneToOne`:

Element name	Optional/Required	Element description
`targetEntity`	Optional	This element specifies the non-owner entity class.
`cascade`	Optional	This element specifies the operations to be cascaded.
`fetch`	Optional	This element specifies the specification value of the fetch strategy.
`optional`	Optional	This element specifies whether or not a null value can be set up for all non-primitive type fields and properties.
`mappedBy`	Optional	This element specifies the name of the field that maintains a relationship in the owner entity class, when added in the elements of the non-owner entity class.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) targetEntity element

Type

  `Class`

Description

  This element specifies the non-owner entity class.

Default value

  Type of the field and property in which the annotation is added

(b) cascade element

Type

`CascadeType[]`

Description

This element specifies the operations to be cascaded.

The following table describes the specifiable values:

- `CascadeType.ALL`: The persist, remove, merge, and refresh operations of the owner entity class are cascaded to the related destination.
- `CascadeType.MERGE`: The merge operation of the owner entity class is cascaded to the related destination.
- `CascadeType.PERSIST`: The persist operation of the owner entity class is cascaded to the related destination.
- `CascadeType.REFRESH`: The refresh operation of the owner entity class is cascaded to the related destination.
- `CascadeType.REMOVE`: The remove operation of the owner entity class is cascaded to the related destination.

Default value

Not to be cascaded

(c) fetch element

Type

`FetchType`

Description

This attribute defines the fetch strategy of data from the database. For details on the fetch strategy, see *9.4.5 Synchronization with the database*.

The following two types of values can be specified:

- `EAGER` strategy: Requests in which the data must be fetched eagerly
- `LAZY` strategy: Requests in which data is fetched lazily when accessed for the first time

Default value

`FetchType.EAGER`

(d) optional element

Type

`boolean`

Description

This element specifies whether or not a null value can be set up for all non-primitive type fields and properties. The following values can be specified:

- `true`: A null value can be set up for all non-primitive type fields and properties.
- `false`: A null value cannot be specified for all non-primitive type fields and properties.

Default value

`true`

(e) mappedBy element

Type

`String`

Description

This element specifies the name of the field that maintains a relationship in the owner entity class, when added in the elements of the non-owner entity class. When this element is specified, the relationship becomes bi-directional.

Default value

None

8.12.38 @OrderBy

(1) Description

This annotation specifies the order in which the information is maintained in the collection, when the entity information is acquired.

The applicable targets are method and field.

(2) Element

The following table lists the elements `@OrderBy`:

Element name	Optional/Required	Element description
`value`	Optional	This element is specified when the entities are to be acquired in an order based on the fields or properties other than the primary key.

The details of attribute for mapping with Cosminexus JPA provider are as follows:

(a) value element

Type

`String`

Description

This element is specified when the entities are to be acquired in an order based on the fields or properties other than the primary key. The fields or properties for which the order is to be specified are demarcated by comma (`,`).

The order of collection is specified after the fields or properties. The following values can be specified. If the order is not specified, the ascending order is assumed.

- `ASC`: Ascending order
- `DESC`: Descending order

In the fields or properties specified in the `value` element, specify the column that stores the values for which you can perform the comparative calculation.

Default value

Ascending order based on the primary key of the entity class

8.12.39 @PersistenceContext

(1) Description

This annotation defines the reference of a container-managed EntityManager. This annotation is added to the class to be looked up.

The applicable targets are class, method, and field.

(2) Element

The following table lists the elements of `@PersistenceContext`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies the lookup name of the EntityManager.
`unitName`	Optional	This element specifies the name of the persistence unit defined in the `persistence.xml` file.
`type`	Optional	This element specifies the type of lifecycle of the persistence context.
`properties`	Optional	This element specifies the vendor-dependent properties specified in `@PersistenceProperty`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

`String`

Description

This element specifies the lookup name of the EntityManager.

You are not required to specify this element when using a DI.

Default value

Null character string

(b) unitName element

Type

`String`

Description

This element specifies the name of the persistence unit defined in the `persistence.xml` file.

When the `unitName` element is specified, set the same name for the persistence unit used by EntityManagerFactory that can be accessed by the JNDI name space.

Default value

Null character string

(c) type element

Type

    `PersistenceContextType`

Description

    This element specifies the type of lifecycle of the persistence context.

    The following two types of values can be specified:

    - `TRANSACTION`: Persistence context of the transaction scope
    - `EXTENDED`: Extended persistence context

Default value

    `TRANSACTION`

(d) properties element

Type

    `PersistenceProperty[]`

Description

    This element specifies the vendor-dependent properties of the JPA Provider specified in `@PersistenceProperty`.

    You can specify the value within the specifiable range of the arrays of `@PersistenceProperty`. For details, see *8.12.41 @PersistenceProperty*.

    When the `properties` element is specified, the properties that cannot be recognized are ignored.

Default value

    Blank array

8.12.40 @PersistenceContexts

(1) Description

This annotation is specified when multiple `@PersistenceContext` are coded concurrently.

The applicable target is class.

(2) Element

The following table lists the elements of `@PersistenceContexts`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@PersistenceContext`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

`PersistenceContext[]`

Description

This element specifies an array of `@PersistenceContext`.

You can specify the value within the specifiable range of the arrays of `@PersistenceContext`. For details, see *8.12.39 @PersistenceContext*.

Default value

None

8.12.41 @PersistenceProperty

(1) Description

This annotation sets up properties in the container-managed EntityManager.

Currently, no properties can be used.

The applicable target is the `properties` element of `@PersistenceContext`.

(2) Element

The following table lists the elements of `@PersistenceProperty`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name of the property.
`value`	Required	This element specifies the value of the property.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

`String`

Description

This element specifies the name of the property.

Default value

None

(b) value element

Type

`String`

Description

This element specifies the value of the property.

You can specify the value depends on the specifications of the properties specified in the `name` element.

Default value

None

8.12.42 @PersistenceUnit

(1) Description

This annotation defines the reference of the EntityManagerFactory. This annotation is added to the class to be looked up.

The applicable targets are class, method, and field.

(2) Element

The following table lists the elements of `@PersistenceUnit`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies the lookup name of the EntityManagerFactory.
`unitName`	Optional	This element specifies the name of the persistence unit defined in the `persistence.xml` file.

The details of attributes supported with Cosminexus JPA provider are as follows:

(a) name element

Type

`String`

Description

This element specifies the lookup name of the EntityManagerFactory. This element specifies the name of the EntityManagerFactory to be registered in the JNDI name space.

The value that can be specified is a character string.

You are not required to specify this element when using a DI.

Default value

Null character string

(b) unitName element

Type

`String`

Description

This element specifies the name of the persistence unit defined in the `persistence.xml` file.

When the `unitName` element is specified, set the same name for the persistence unit used by EntityManagerFactory that can be accessed by the JNDI name space.

Default value

Null character string

8.12.43 @PersistenceUnits

(1) Description

This annotation is specified when multiple `@PersistenceUnit` are coded concurrently.

The applicable target is class.

(2) Element

The following table lists the elements of `@PersistenceUnits`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies the arrays of `@PersistenceUnit`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type
`PersistenceUnit[]`

Description
This element specifies the arrays of `@PersistenceUnit`.
You can specify the value within the specifiable range of the arrays of `@PersistenceUnit`. For details, see *8.12.42 @PersistenceUnit*.

Default value
None

8.12.44 @PostLoad

(1) Description

This annotation indicates the callback method invoked after an entity is read from the cache or after the `SELECT` statement is issued in the database. This annotation is applicable in the methods of the entity class, mapped superclass, or entity listener class.

The applicable target is method.

(2) Element

`@PostLoad` does not have attributes.

8.12.45 @PostPersist

(1) Description

This annotation indicates the callback method invoked after the `INSERT` statement is issued in the database. This annotation is applicable in the methods of the entity class, mapped superclass, or entity listener class.

The applicable target is method.

(2) Element

`@PostPersist` does not have attributes.

8.12.46 @PostRemove

(1) Description

This annotation indicates the callback method invoked after the `DELETE` statement is issued in the database. This annotation is applicable in the methods of the entity class, mapped superclass, or entity listener class.

The applicable target is method.

(2) Element

`@PostRemove` does not have attributes.

8.12.47 @PostUpdate

(1) Description

This annotation indicates the callback method invoked after the `UPDATE` statement is issued in the database. This annotation is applicable in the methods of the entity class, mapped superclass, or entity listener class.

The applicable target is method.

(2) Element

`@PostUpdate` does not have attributes.

8.12.48 @PrePersist

(1) Description

This annotation indicates the callback method invoked before the `INSERT` statement is issued in the database. This annotation is applicable in the methods of the entity class, mapped superclass, or entity listener class.

The applicable target is method.

(2) Element

`@PrePersist` does not have attributes.

8.12.49 @PreRemove

(1) Description

This annotation indicates the callback method invoked before the `DELETE` statement is issued in the database. This annotation is applicable in the methods of the entity class, mapped superclass, or entity listener class.

The applicable target is method.

(2) Element

`@PreRemove` does not have attributes.

8.12.50 @PreUpdate

(1) Description

This annotation indicates the callback method invoked before the `UPDATE` statement is issued in the database. This annotation is applicable in the methods of the entity class, mapped superclass, or entity listener class.

The applicable target is method.

(2) Element

`@PreUpdate` does not have attributes.

8.12.51 @PrimaryKeyJoinColumn

(1) Description

This annotation specifies the column used as the external key when binding with another table. This annotation is used in the following cases:

- When the names of the primary key of the superclass and the primary key of the subclass of an entity are different in the JOINED strategy of the inheritance mapping strategy
- When the primary table and secondary table are to be bound in `@SecondaryTable`#
- When the primary key of the non-owner entity class is used as an external key in the `OneToOne` relationship

#

Here, this annotation is used within `@SecondaryTable`.

The applicable targets are class, method, and field.

(2) Element

The following table lists the elements of `@PrimaryKeyJoinColumn`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies the column name for binding the target tables.
`referencedColumnName`	Optional	This element specifies the column name of the primary key of binding-destination table that is referenced by the column specified in the `name` element.
`columnDefinition`	Optional	This element is used to describe the constraints added to the column in the DDL, when the `CREATE` statement is output. Note that Cosminexus JPA provider does not support this attribute.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

`String`

Description

This element specifies the column name for binding the target tables.

The column name that can be specified depends on the database specifications.

Default value

- **When the JOINED strategy is used**

  Column name of the primary key of primary table of the superclass
- **When @SecondaryTable is used**

  Column name of the primary key of primary table
- **When the OneToOne relationship is used**

  Column name of the primary key of target entity table

(b) referencedColumnName element

Type

`String`

Description

This element specifies the column name of the primary key of binding-destination table that is referenced by the column specified in the `name` element. Specify the same value as the character string of the name element of `@Column`. Arrange the upper case and lower case characters in the character string to be specified.

The column name that can be specified depends on the database specifications.

Even when the unique key constraints are used instead of specifying the primary key in the column in the `OneToOne` relationship, the operation will continue, but will not be guaranteed.

Default value

- **When the JOINED strategy is used**

  Column name of the primary key of primary table of the superclass

- **When @SecondaryTable is used**
  Column name of the primary key of primary table
- **When the OneToOne relationship is used**
  Column name of the primary key of target entity table

8.12.52 @PrimaryKeyJoinColumns

(1) Description

This annotation is specified when multiple `@PrimaryKeyJoinColumn` are coded concurrently.

The applicable targets are class, method, and field.

(2) Element

The following table lists the elements of `@PrimaryKeyJoinColumns`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@PrimaryKeyJoinColumn`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type
   `PrimaryKeyJoinColumn[]`

Description
   This element specifies an array of `@PrimaryKeyJoinColumn`.
   You can specify the value within the specifiable range of `@PrimaryKeyJoinColumn`. For details, see *8.12.51 @PrimaryKeyJoinColumn*.

Default value
   None

8.12.53 @QueryHint

(1) Description

This annotation specifies a database-specific query hint.

You can set up a pessimistic lock and the cache functionality of the entity.

The applicable target is the `hints` element of `@NamedQuery` or `@NamedNativeQuery`.

(2) Element

The following table lists the elements of `@QueryHint`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name of the hint.
`value`	Required	This element specifies the value of the hint.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

`String`

Description

This element specifies the name of the hint to be used. The following value can be specified:

`cosminexus.jpa.pessimistic-lock`

This is the name of the hint that specifies whether or not to use a pessimistic lock.

Default value

None

(b) value element

Type

`String`

Description

This element specifies the value of the hint. The following values are specified based on the name of the hint specified in the `name` element:

Specification value when `cosminexus.jpa.pessimistic-lock` is specified in the `name` element

- `NoLock`: Specified when the pessimistic lock is not used.
- `Lock`: Specified when the pessimistic lock is used. If the target table is already locked, unlocking is awaited. The SQLs issued at this point are specified as follows, for each used database:
  In Oracle: `SELECT.... FOR UPDATE`
  In HiRDB: `SELECT....WITH EXCLUSIVE LOCK`
- `LockNoWait`: Specified when the pessimistic lock is used. If the target table is already locked, an exception occurs. The SQLs issued at this point are specified as follows, for each used database:
  In Oracle: `SELECT.... FOR UPDATE NO WAIT`
  In HiRDB: `SELECT....WITH EXCLUSIVE LOCK NO WAIT`

Default value

When `cosminexus.jpa.pessimistic-lock` is specified in the `name` element

`NoLock`

8.12.54 @SecondaryTable

(1) Description

This annotation specifies the secondary table in the entity class.

This annotation is specified when the entity class is mapped in multiple tables of the database.

When `@SecondaryTable` is not specified within the entity class, all persistence properties or persistence fields of the entity class will be mapped to the tables specified in the primary table.

The applicable target is class.

(2) Element

The following table lists the elements of `@SecondaryTable`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the secondary table name.
`catalog`	Optional	This element specifies the catalog name of the secondary table. Note that Cosminexus JPA provider does not support this attribute.
`schema`	Optional	This element specifies the schema name of the secondary table.
`pkJoinColumns`	Optional	This element specifies the external key column used to bind the secondary table to the primary table.
`uniqueConstraints`	Optional	This element specifies the unique key constraints in the table. Note that Cosminexus JPA provider does not support this attribute.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

`String`

Description

This element specifies the secondary table name.

The table name that can be specified depends on the database specifications.

Default value

None

(b) schema element

Type

`String`

Description

This element specifies the schema name of the secondary table.

The schema name that can be specified depends on the database specifications.

Default value

Default schema name of the database used

(c) pkJoinColumns element

Type

`PrimaryKeyJoinColumn[]`

Description

This element specifies the external key column of the secondary table. This annotation is specified in the arrays of `@PrimaryKeyJoinColumn`.

When this element is not specified, the external key column of the secondary table has the same name and type as the primary key column of the primary table. Therefore, the secondary table references the primary key column of the primary table.

Default value

You can specify the value within the specifiable range of the arrays of `@PrimaryKeyJoinColumn`. For details, see *8.12.51 @PrimaryKeyJoinColumn*.

8.12.55 @SecondaryTables

(1) Description

This annotation is specified when multiple `@SecondaryTable` are coded concurrently.

The applicable target is class.

(2) Element

The following table lists the elements of `@SecondaryTables`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@SecondaryTable`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

`SecondaryTable[]`

Description

This element specifies an array of `@SecondaryTable`.

You can specify the value within the specifiable range of the arrays of `@SecondaryTable`. For details, see *8.12.54 @SecondaryTable*.

Default value

None

8.12.56 @SequenceGenerator

(1) Description

This annotation specifies the settings of the sequence generator that creates the primary key. The following settings are required when using `@SequenceGenerator`:

- Specify `GenerationType.SEQUENCE` in the `strategy` element of `@GeneratedValue`.
- Set up the name specified in the `generator` element of `@GeneratedValue` to the `name` element of `@SequenceGenerator`.

The sequence generator is specified in the fields or properties of the entity class or primary key. The scope of the sequence generator name is enabled in the persistence unit.

When creating a sequence object, specify a positive integer in the increment interval (`INCREMENT BY`) between sequential numbers, and in the initial value (`START WITH`) of the generated sequential number. When `1` is specified in the initial value (`START WITH`), the primary key is generated from `1`. The operation will not be guaranteed if a negative value is specified.

The applicable targets are class, method, and field.

(2) Element

The following table lists the elements of `@SequenceGenerator`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name specified in the `generator` element of the `@GeneratedValue` annotation.
`sequenceName`	Optional	This element specifies the name of the database sequence object for acquiring an existing primary key value, or an already defined primary key value.
`initialValue`	Optional	This element specifies the initial value when the generation of the primary key value by the sequence object is started. Note that Cosminexus JPA provider does not support this attribute. Ignore when the value is specified.
`allocationSize`	Optional	This element specifies the size of allocating the primary key value from the sequence.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

`String`

Description

This element specifies the name specified in the `generator` element of the `@GeneratedValue` annotation.

The value that can be specified is a character string.

Default value

None

(b) sequenceName element

Type

`String`

Description

This element specifies the name of the database sequence object for acquiring an existing primary key value, or an already defined primary key value.

The sequence object name that can be specified depends on the database specifications.

Default value

Specified value of the `generator` element of `@GeneratedValue`

(c) allocationSize element

Type

`int`

Description

This element specifies the allocation size of the primary key value from the sequence. The sequence object name that can be specified depends on the database specifications.

The size that can be specified is a numeric value that is at least one more than the `int` type. Specify a value same as the increment interval of the sequence object. The operation will not be guaranteed if you specify a different value.

Note that in this element, you can specify the maximum value used during execution. If you specify a large value for acquiring the management area of the sequence number, the `java.lang.OutOfMemoryError` exception will occur during the execution.

Default value

`50`

8.12.57 @SqlResultSetMapping

(1) Description

This annotation specifies the result set mapping of an SQL query.

The applicable target is class.

(2) Element

The following table lists the elements of `@SqlResultSetMapping`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the name of the result set mapping.
`entities`	Optional	This element specifies an array of `@EntityResult`.
`columns`	Optional	This element specifies an array of `@ColumnResult`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

`String`

Description

This element specifies the name of the result set mapping.

The value that can be specified is a character string.

Default value

None

(b) entities element

Type

`EntityResult[]`

Description

This element specifies an array of `@EntityResult`.

You can specify the value within the specifiable range of the arrays of `@EntityResult`. For details, see *8.12.15 @EntityResult*.

Default value

Blank array

(c) columns element

Type

`ColumnResult[]`

Description

This element specifies an array of `@ColumnResult`.

You can specify the value within the specifiable range of the arrays of `@ColumnResult`. For details, see *8.12.7 @ColumnResult*.

Default value

Blank array

8.12.58 @SqlResultSetMappings

(1) Description

This annotation is specified when multiple `@SqlResultSetMapping` are coded concurrently.

The applicable target is class.

(2) Element

The following table lists the elements of `@SqlResultSetMappings`:

Element name	Optional/Required	Element description
`value`	Required	This element specifies an array of `@SqlResultSetMapping`.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

`SqlResultSetMapping[]`

Description

This element specifies an array of `@SqlResultSetMapping`.

You can specify the value within the specifiable range of the arrays of `@SqlResultSetMapping`. For details, see *8.12.57 @SqlResultSetMapping*.

Default value

None

8.12.59 @Table

(1) Description

This annotation specifies the primary table in the entity class.

Even when `@Table` is not specified explicitly in the entity class, the entity class is handled as if `@Table` were specified. In such a case, the default value will be applied in each element of `@Table`.

If more than one table is specified for mapping the entities, use either `@SecondaryTable` or `@SecondaryTables`.

The applicable target is class.

(2) Element

The following table lists the elements of `@Table`:

Element name	Optional/Required	Element description
`name`	Optional	This element specifies the table name.
`catalog`	Optional	This element specifies the catalog name of the table. Note that Cosminexus JPA provider does not support this attribute.
`schema`	Optional	This element specifies the schema name of the table.
`uniqueConstraints`	Optional	This element specifies the unique key constraints in the table. Note that Cosminexus JPA provider does not support this attribute.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

`String`

Description

This element specifies the table name.

The table name that can be specified depends on the database specifications.

Default value

Entity name

(b) schema element

Type

`String`

Description

This element specifies the schema name of the table.

The schema name that can be specified depends on the database specifications.

Default value

Default schema name of the database used

8.12.60 @TableGenerator

(1) Description

This annotation specifies the settings of the generator that creates the primary key.

The following settings are required when using `@TableGenerator`:

- Specify `GenerationType.TABLE` in the `strategy` element of `@GeneratedValue`.
- Set up the name specified in the `generator` element of `@GeneratedValue` to the `name` element of `@TableGenerator`.

The table generator is specified in the fields or properties of the entity class or primary key. The scope of the generator name is enabled in the persistence unit.

Use the rows of the generator table when generating the primary key value in an entity.

When creating a table for managing the sequence, specify a positive integer in the initial value. If `0` is specified in the initial value, the primary key will be generated from `1`.

The applicable targets are class, method, and field.

(2) Element

The following table lists the elements of `@TableGenerator`:

Element name	Optional/Required	Element description
`name`	Required	This element specifies the generator name for the primary key value.
`table`	Optional	This element specifies the name of the table that maintains the generated primary key values.
`catalog`	Optional	This element specifies the catalog name of the table that maintains the generated primary key values. Note that Cosminexus JPA provider does not support this attribute.
`schema`	Optional	This element specifies the schema name of the table that maintains the generated primary key values.
`pkColumnName`	Optional	This element specifies the primary key column name of the table that maintains the generated primary key values.

Element name	Optional/Required	Element description
valueColumnName	Optional	This element specifies the column name that maintains the final generated value.
pkColumnValue	Optional	This element specifies the primary key value of the table that maintains the generated primary key values.
initialValue	Optional	This element specifies the value used for initializing the column that maintains the recent generated values. Note that Cosminexus JPA provider does not support this attribute. Ignore when the value is specified.
allocationSize	Optional	This element specifies the size of allocating the primary key value from the generator.
uniqueConstraints	Optional	This element specifies the unique key constraints in the table that maintains the generated primary key values. Note that Cosminexus JPA provider does not support this attribute. Ignore when the value is specified.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) name element

Type

`String`

Description

This element specifies the generator name for the primary key value.

The value that can be specified is a character string.

Default value

None

(b) table element

Type

`String`

Description

This element specifies the name of the table that maintains the generated primary key values.

The table name that can be specified depends on the database specifications.

Default value

`"SEQUENCE"`

(c) schema element

Type

`String`

Description

This element specifies the schema name of the table that maintains the generated primary key values.

The schema name that can be specified depends on the database specifications.

Default value

Default schema name of the database used

(d) pkColumnName element

Type

`String`

Description

This element specifies the primary key column name of the table that maintains the generated primary key values.

The column name that can be specified depends on the database specifications.

Default value

`"SEQ_NAME"`

(e) valueColumnName element

Type

`String`

Description

This element specifies the column name that maintains the final generated value.

The column name that can be specified depends on the database specifications.

Default value

`"SEQ_COUNT"`

(f) pkColumnValue element

Type

`String`

Description

This element specifies the primary key value of the table that maintains the generated primary key values.

The value that can be specified depends on the type of column of the generated primary key.

Default value

Character string specified in the `name` element

(g) allocationSize element

Type

`int`

Description

This element specifies the allocation size of the primary key value from the generator.

The value that can be specified is a numeric value that is at least one more than the `int` type.

Note that you can specify the maximum value used during the execution in this element. If you specify a large value for acquiring the management area of the sequence number, the `java.lang.OutOfMemoryError` exception will occur during the execution.

Default value

`50`

8.12.61 @Temporal

(1) Description

This annotation is specified in the persistence property or persistence field having the type that expresses the time (`java.util.Date` and `java.util.Calendar`). This annotation can be used along with `@Basic`.

However, `@Version` and `@Temporal` cannot be specified concurrently. Specify either of these annotations.

The applicable targets are method and field.

(2) Element

The following table lists the elements of `@Temporal`:

Element name	Optional/Required	Element description
`value`	Required	This element is specified in the `TemporalType` enumeration type corresponding to the database type.

The details of attributes that are supported with Cosminexus JPA provider are as follows:

(a) value element

Type

`TemporalType`

Description

This element is specified in the `TemporalType` enumeration type corresponding to the database type.

The following three types of values can be specified:

- `TemporalType.DATE`: Same as `java.sql.Data`.
- `TemporalType.TIME`: Same as `java.sql.Time`.
- `TemporalType.TIMESTAMP`: Same as `java.sql.Timestamp`.

Default value

None

8.12.62 @Transient

(1) Description

This annotation specifies the fields or properties of the following non-persisting classes:

- Entity class
- Mapped superclass
- Embedded class

The applicable targets are method and field.

The value of a field, in which `@Transient` is defined, is not persisted. However, since this value is stored in the persistence context, you can acquire the setup value. Howeachou cannot acquire the value from another EntityManager.

(2) Element

`@Transient` does not have attributes.

8.12.63 @Version

(1) Description

This annotation specifies the `version` field or `version` property used to make use of the optimistic lock functionality.

The following types are supported by the `version` field or `version` property:

- `int`
- `java.lang.Integer`
- `short`
- `java.lang Short`
- `long`
- `java.lang Long`
- `java.sql.Timestamp`

Make note of the following points when using this annotation:

- You cannot specify `@Version` and `@Temporal` concurrently. Specify either of these annotations.
- Do not specify the `version` property in a table other than the primary table.
- In some applications, the field or property specified in `@Version` must not be updated.
- For bulk update, when multiple records are updated at once using SQL, the `version` field or `version` property is not updated automatically. Therefore, when you use the optimistic lock for performing bulk update, you must reference and update manually.
- You can set up only a single `version` field or `version` property for an entity class. If you set up multiple `version` fields or `version` properties, only a single will be enabled. The sequence for enabling the settings is not fixed.

The applicable targets are method and field.

(2) Element

`@Version` does not have attributes.

8.12.64 Correspondence between the annotations and O/R mapping

The following table describes the correspondence between the annotations and O/R mapping files:

Table 8–12: Correspondence between the annotations and O/R mapping files

Annotation	O/R mapping elements
@AssociationOverride	<association-override>
@AssociationOverrides	--
@AttributeOverride	<element-override>
@AttributeOverrides	--
@Basic	<basic>
@Column	<column>
@ColumnResult	<column-result>
@DiscriminatorColumn	<discriminator-column>
@DiscriminatorValue	<discriminator-value>
@Embeddable	<embeddable>
@Embedded	<embedded>
@EmbeddedId	<embedded-id>
@Entity	<entity>
@EntityListeners	<entity-listeners>
@EntityResult	<entity-result>
@Enumerated	<enumerated>
@ExcludeDefaultListeners	<exclude-default-listeners>
@ExcludeSuperclassListeners	<exclude-superclass-listeners>
@FieldResult	<field-result>
@GeneratedValue	<generated-value>
@Id	<id>
@IdClass	<id-class>
@Inheritance	<inheritance>
@JoinColumn	<join-column>
@JoinColumns	--
@JoinTable	<join-table>
@Lob	<lob>
@ManyToMany	<many-to-many>
@ManyToOne	<many-to-one>
@MapKey	<map-key>
@MappedSuperclass	<mapped-superclass>
@NamedNativeQueries	--
@NamedNativeQuery	<named-native-query>
@NamedQueries	--

Annotation	O/R mapping elements
@NamedQuery	<named-query>
@OneToMany	<one-to-many>
@OneToOne	<one-to-one>
@OrderBy	<order-by>
@PersistenceContext	--#
@PersistenceContexts	--#
@PersistenceProperty	--#
@PostLoad	<post-load>
@PostPersist	<post-persist>
@PostRemove	<post-remove>
@PostUpdate	<post-update>
@PrePersist	<pre-persist>
@PreRemove	<pre-remove>
@PreUpdate	<pre-update>
@PrimaryKeyJoinColumn	<primary-key-join-column>
@PrimaryKeyJoinColumns	--
@QueryHint	<hint>
@SecondaryTable	<secondary-table>
@SecondaryTables	--
@SequenceGenerator	<sequence-generator>
@SqlResultSetMapping	<sql-result-set-mapping>
@SqlResultSetMappings	--
@Table	<table>
@TableGenerator	<table-generator>
@Temporal	<temporal>
@Transient	<transient>
@UniqueConstraint	<unique-constraint>
@Version	<version>

Legend:

--: Not applicable.

#

Not an annotation for O/R mapping.

9 Cosminexus JPA Provider

This chapter gives an overview of the Cosminexus JPA Provider functionality, describes the application implementation methods, and the execution environment settings.

9.1 Organization of this chapter

Cosminexus JPA Provider is a JPA provider that implements API functions and the internal processing determined in the JPA specifications and is provided by Application Server provided to the user in the library format.

This chapter describes the Cosminexus JPA Provider functionality. The following table describes the organization of this chapter.

Table 9–1: Organization of this chapter (Cosminexus JPA Provider functionality)

Category	Title	Reference location
Description	Cosminexus JPA Provider	*9.2*
	Updating a database using entities	*9.3*
	Entity operations by `EntityManager`	*9.4*
	Defining the mapping information between the database and Java objects	*9.5*
	Entity relationships	9.6
	Cache functionality of the entity objects	*9.7*
	Auto-numbering of the primary key values	9.8
	Database operations based on the query language	*9.9*
	Optimistic lock	*9.10*
	Pessimistic lock in JPQL	*9.11*
Implementation	Creating an entity class	*9.12*
	Procedure for inheriting an entity class	*9.13*
	Procedure for using `EntityManager` and `EntityManagerFactory`	*9.14*
	Procedure for specifying the callback method	*9.15*
	Procedure for referencing and updating the database with the query language	*9.16*
	JPQL coding method	9.17
	Defining `persistence.xml`	*9.18*
Settings	Settings in the execution environment	*9.19*

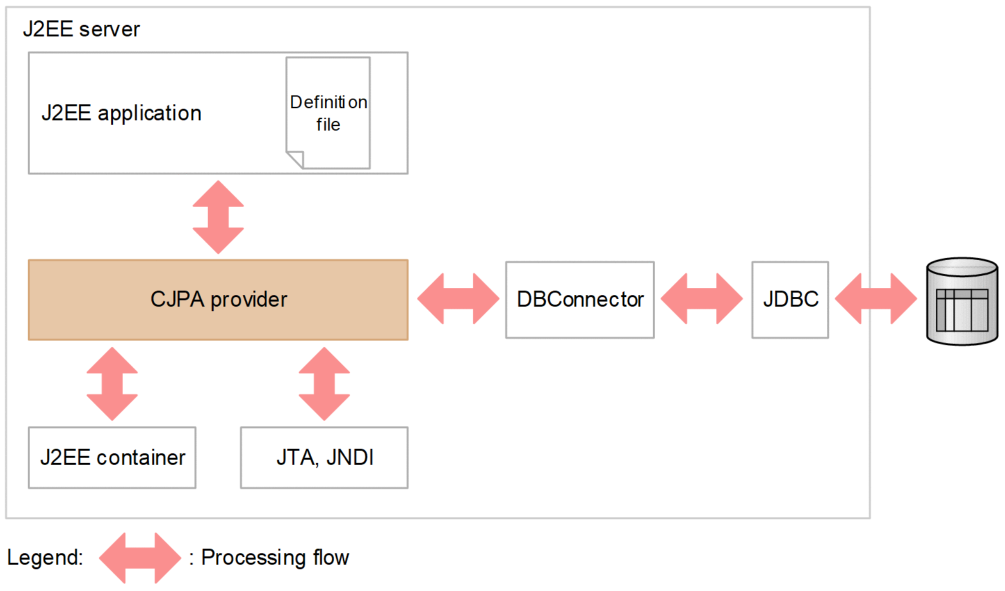
Note:

There is no specific description of *Operations* for this functionality.

9.2 Cosminexus JPA Provider

Cosminexus JPA Provider is the JPA provider provided by Application Server. The following figure shows the positioning of Cosminexus JPA Provider.

Figure 9–1: Positioning of Cosminexus JPA Provider



Cosminexus JPA Provider is the J2EE server functionality and uses JTA and JNDI to operate the database. The following sections give an overview of the Cosminexus JPA Provider functionality.

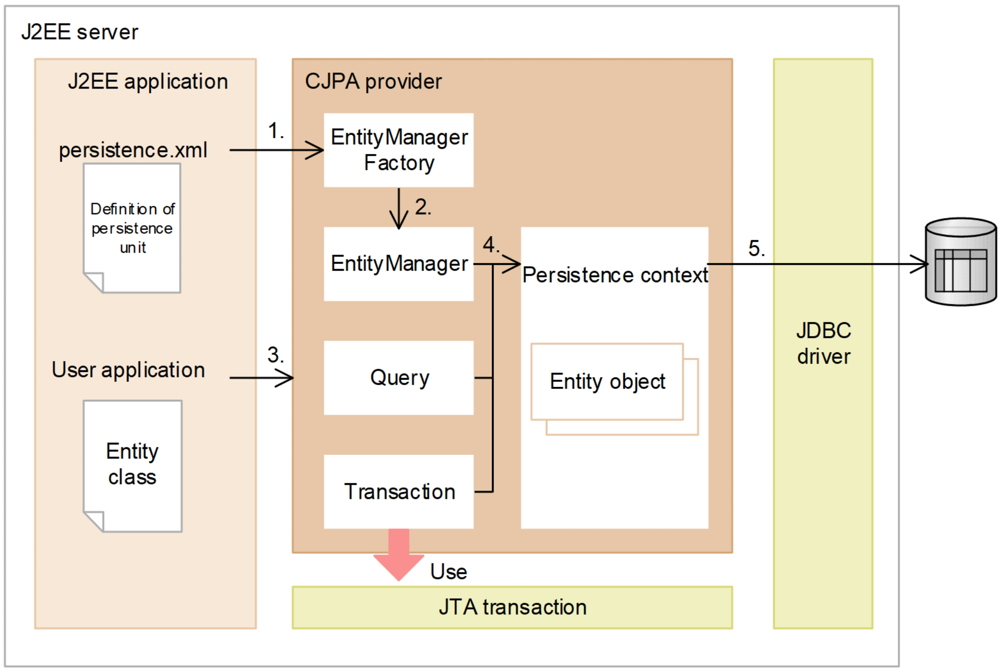
9.2.1 Processing in Cosminexus JPA Provider

Cosminexus JPA Provider provides the following functionality:

- Mapping of the entity objects and database
- Managing the entity objects
- Data operations using JPQL
- Accessing the database through JDBC

To use JPA, you operate the interfaces provided by Cosminexus JPA Provider by using the user-created applications. The following figure shows the processing executed in Cosminexus JPA Provider.

Figure 9–2: Processing executed in Cosminexus JPA Provider



A description of the figure is as follows:

1. Cosminexus JPA Provider generates `EntityManagerFactory` from the persistence unit information defined in `persistence.xml`.
   A persistence unit is a logical group that collects information such as the classes to be mapped, O/R mapping information, and the data sources. For details on a persistence unit, see *8.4.4 Persistence unit*.
2. `EntityManager` is generated from the generated `EntityManagerFactory`.
3. `EntityManager`, query, or transaction is operated from the user application.
4. The entity object is operated through the persistence context from `EntityManager`, query, or transaction.
5. The changes in the entity object are applied to the database through the JDBC driver.

Note that there are two types of `EntityManager`:

- Container-managed `EntityManager`
  `EntityManager` managed by the J2EE container
- Application-managed `EntityManager`
  `EntityManager` managed by an application

For details on the types of `EntityManager`, see *8.5.1 EntityManager and persistence context*.

Furthermore, a transaction manages operations such as the transaction commit and rollback. With Cosminexus JPA Provider, you can use the resource local transaction as the transaction. To integrate with JTA transactions, use the functionality provided by the J2EE container.

9.2.2 Functionality provided by Cosminexus JPA Provider

The following functionality is provided by Cosminexus JPA Provider.

Table 9–2: Functionality provided by Cosminexus JPA Provider

Functionality	Overview of functionality	Standard/ Extended #	Reference location
Updating a database using entities	Updates the data in the database using the entities generated from the user applications.	Standard	*9.3*, *9.4*
Defining the mapping information between the database and Java objects	The mapping information between the database and Java objects can be defined with annotations or the O/R mapping file.	Standard	*9.5*
Entity relationship settings	Sets up the relationship between the database tables by using entities.	Standard	*9.6*
Entity object cache	This functionality stores the content of the entity object of `EntityManager` in the memory. When an entity object with the same content is invoked, the entity object in the memory is used.	Extended	*9.7*
Auto-numbering of primary key	Cosminexus JPA Provider automatically stores the primary key when the entity object is used to insert the data in the database. Even if the user does not specify the primary key, a unique value is stored.	Standard	*9.8*
Database operations based on the query language	The database can be operated using the query language. With Cosminexus JPA Provider, you can use JPQL or SQL as the query language.	Standard	*9.9*
Optimistic lock	This is one of the database locking methods. The data is updated after confirming that the data in the database is not being updated by another transaction. An optimistic lock does not lock the data, so a deadlock does not occur.	Standard	*9.10*
Pessimistic lock in JPQL	This is one of the database locking methods. A dedicated lock is set for the record so that the record is not updated by another transaction. A pessimistic lock is executed only when JPQL is used.	Extended	*9.11*

#

Indicates whether the functionality is based on the JPA 1.0 specifications.

Standard: Indicates that the functionality is based on the JPA 1.0 specifications.

Extended: Indicates that the functionality is unique to Cosminexus JPA Provider.

For details on the functionality, see the sections mentioned in the *Reference location* column in the above table.

Also, you can collect PRF traces by using the CJPA provider. For details on key points of collecting PRF traces, see *15. Performance Analysis Trace*.

9.2.3 Preconditions for using Cosminexus JPA Provider

This section describes the preconditions for using Cosminexus JPA Provider.

(1) Available components

The components that use Cosminexus JPA Provider are EJB 3.0 and later in the case of EJBs and Servlet 2.5 and later in the case of Web components. For details on the available components, see *8.3.2 Available components*.

(2) Connectable databases

You can connect to the following databases when you use Cosminexus JPA Provider:

- Oracle
- HiRDB

(3) Available DB Connectors

Cosminexus JPA Provider uses DB Connector to update the database. You can use DB Connectors listed in the following table with Cosminexus JPA Provider.

Table 9–3: DB Connectors available with Cosminexus JPA Provider

Database used	Transaction type	Available DB Connector
Oracle	`LocalTransaction` `NoTransaction`	`DBConnector_Oracle_CP.rar`
	`XATransaction`	`DBConnector_Oracle_XA.rar`

Oracle RAC	`LocalTransaction`
	`NoTransaction`
		`DBConnector_Oracle_CP.rar`
		`DBConnector_CP_ClusterPool_Root.rar`
		`DBConnector_Oracle_CP_ClusterPool_Member.rar`

HiRDB	`LocalTransaction` `NoTransaction`	`DBConnector_HiRDB_Type4_CP.rar`
	`XATransaction`	`DBConnector_HiRDB_Type4_XA.rar`

(4) Environment that cannot be used

You cannot use Cosminexus JPA Provider in the following environment:

- Execution environment for the batch applications
- Execution environment for the EJB client applications

(5) Using the method cancellation functionality

With Cosminexus JPA Provider, a unique binary code is embedded in the accessor method for OneToOne and ManyToOne LAZY fetch. As a result, the accessor method is subject to method cancellation. Therefore, when LAZY fetch is specified for the OneToOne relationship or ManyToOne relationship, the entity class, embeddable class, and mapped super-class must be registered in the protected area.

Note that if the classes are not registered in the protected area, method cancellation might occur on the embedded binary code. The operations when the classes are not registered in the protected area might not function properly.

Whether the classes are registered or not registered in the protected area, the following stack trace is output due to method timeout. However, if the classes are registered in the protected area, method cancellation does not occur.

In the following example, the embedded `EntityClass1._toplink_getmappingClass2` is invoked by extending the `EntityClass1.getMappingClass2` method invoked by the user. Consequently, a method timeout occurs, but method cancellation does not occur.

```
message-id        message(LANG=ja)
KDJE52703-W       A timeout occurred while the user program was executing. (t
hreadID = 23794987, rootAPInfo = 10.209.11.124/5964/0x4828eb62000128e0, appl
ication = JPA_JavaEE_TP, bean = TestBean, method = doTest)
    jpa.test.annotation.onetoone.entity.EntityClass1._toplink_getmappingClas
s2(EntityClass1.java)
                       locals:
                         (jpa.test.annotation.onetoone.entity.EntityClass1)
this = <0x11e35878> (jpa.test.annotation.onetoone.entity.EntityClass1)
                        at jpa.test.annotation.onetoone.entity.EntityClass1.
getMappingClass2(EntityClass1.java:34)
                       locals:
                         (jpa.test.annotation.onetoone.entity.EntityClass1)
this = <0x11e35878> (jpa.test.annotation.onetoone.entity.EntityClass1)
```

For details on registering the classes in the protected area, see *2.2.5 criticalList.cfg (Protected areas list file)* in the *uCosminexus Application Server Definition Reference Guide*.

(6) Using the annotation reference control functionality

When you use Cosminexus JPA Provider, you cannot use the annotation reference control functionality. If the annotation reference control functionality is enabled, you cannot define the persistence context and persistence unit references.

9.2.4 Estimating the number of DB Connector connections

This section describes the estimation of the DB Connector resources required for using Cosminexus JPA Provider.

With Cosminexus JPA Provider, you obtain the DB Connector connections. The formula for estimating the number of connections used by Cosminexus JPA Provider is as follows:

Formula for estimating the number of connections

Number of connections used by the Cosminexus JPA Provider applications

= Number of concurrent executions of applications using the JPA functionality #

# In the applications using the JPA functionality, when `EntityManager` of multiple persistence units is used or when the transactions used by `EntityManager` for invoking the business methods are different, a connection is required for each `EntityManager`.

Important note

When the user acquires a connection separately from the JPA functionality, the number of connections can be reduced by using the connection sharing functionality. For details on connection sharing, see the manual *uCosminexus Application Server Common Container Functionality Guide*.

9.3 Updating a database using entities

With Cosminexus JPA Provider, you can update a database using entities.

To use the JPA, the user must create an entity class for handling the database tables as Java objects. If you use the entity class, the data in the database is operated through Cosminexus JPA Provider. At this time, you operate the database through the `EntityManager` interface provided by Cosminexus JPA Provider.

The database operations using the entity class are as follows:

1. Generating the entity class instance (entity object).
2. Passing the entity object to the argument of the `EntityManager` interface to operate the database.
   To perform these operations, an entity class must be created.

`EntityManager` is an object used for operating the entities and controlling the states. The entity operations include the `persist` operation, `remove` operation, `merge` operation, `flush` operation, and `refresh` operation. For details on these operations, see *9.4.1 Transition of entity states*.

9.4 Entity operations by EntityManager

The state of the entity is controlled by `EntityManager` provided with Cosminexus JPA Provider. This section describes the entity operations in `EntityManager`.

9.4.1 Transition of entity states

An entity has a state. When you perform operations for an entity, the state of the entity changes. This section describes the types of entity states, the operations executed for the entity, and the entity operations and state transition.

(1) Types of entity states

An entity has four types of states, namely `new`, `managed`, `detached`, and `removed`. The entity state is changed by using the `EntityManager` method to operate an entity.

The following table lists and describes each entity state.

Table 9–4: Entity state

State of entity instances	Explanation
`new`	A state in which the entity is newly generated. A newly generated entity instance does not have a persistence identity, and is, therefore, not associated with the persistence context.
`managed`	A state in which the entity has a persistence identity associated with the persistence context and is managed with the persistence context.
`detached`	A state in which the entity has a persistence identity that is not associated with the persistence context.
`removed`	A state in which the entity has a persistence identity and is associated with the persistence context. Also, includes the state in which the entity instance is scheduled for deletion from the database.

(2) Operations for the entities

If the user searches the database records, Cosminexus JPA Provider stores the obtained data in the entity fields. Also, when the user updates the database contents, the entity state is applied to the database by changing the state of the entity registered in the persistence context and by committing the transaction. Thus, by executing operations for the entity, the database information is updated.

The following table lists and describes the types of operations for the entities.

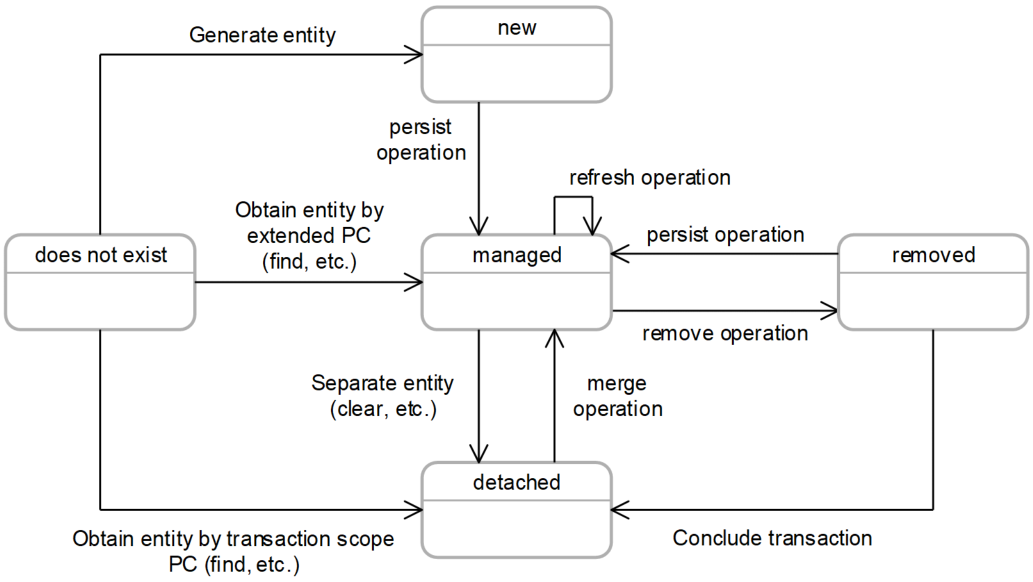
Table 9–5: Types of operations for the entities

Operations	Explanation
`flush`	This operation applies the content of the entity object to the database.
`merge`	In this operation, the entity object that was not managed by `EntityManager` is managed by `EntityManager`.
`persist`	This operation manages and perpetuates the entity object.
`refresh`	This operation applies the database content to the entity object.
`remove`	This operation sets the entity object to the state of scheduled deletion.

(3) Operations and state transition for the entities

The following figure shows the operations and the state transition for the entity instances.

Figure 9–3: Operations and state transition for the entity instances



The following table describes the operations and state transition for the entities.

Table 9–6: Transition of entity states

Operations	State			
	new	managed	detached	removed
`persist`	`managed`[#1]	`managed`	`managed`[#1]	`managed`
`remove`	`new`	`removed`	Exception[#2, #3]	`removed`
`merge`	• Copy source `new` • Copy destination `managed`	• Copy source `managed` • Copy destination `managed`	• Copy source `detached` • Copy destination `managed`	Exception[#3, #4]
`refresh`	Exception[#2, #4]	`managed`[#5]	Exception[#2, #4]	Exception[#2, #4]
`commit`	--	#6	--	`detached`
`rollback`	--	`detached`	--	`detached`
`flush`	--	`managed`	--	`detached`
`clear`	--	`detached`	--	`detached`

Legend:

--: Not applicable

#1

If an exception occurs in the `persist` operation, the state enters the original state without being changed.

#2

The exception that occurs is `IllegalArgumentException`.

#3

If the corresponding line does not exist in the database, the operation is ignored.

#4

If an exception occurs, the state does not change and remains as the original.

#5

If the corresponding line does not exist in the database, `EntityNotFoundException` occurs.

#6

For a persistence context in the transaction scope, the state is `detached`. For an extended persistence context, the state is `managed`.

Furthermore, the operations when `persist`, `remove`, `merge`, and `refresh` are executed outside the transaction vary according to the type of the persistence context.

- For a persistence context in the transaction scope
  `TransactionRequiredException` occurs.
- For an extended persistence context
  The state changes and the states are applied to the database when the next transaction is concluded.

(4) Propagation of operations to the entities

When the entities have a relationship and if you specify the `cascade` attribute of the annotation indicating the relationship, the operations for the entity are propagated to the related entities. You can specify the values listed in the following table in the `cascade` attribute.

Table 9–7: Types of cascade attributes

Types of cascade attributes	Explanation
`CascadeType.ALL`	The `persist`, `remove`, `merge`, and `refresh` operations are propagated to the relation destination.
`CascadeType.PERSIST`	The `persist` operation is propagated to the relation destination.
`CascadeType.REMOVE`	The `remove` operation is propagated to the relation destination.
`CascadeType.MERGE`	The `merge` operation is propagated to the relation destination.
`CascadeType.REFRESH`	The `refresh` operation is propagated to the relation destination.

9.4.2 persist operation for the entities

To execute the `persist` operation for the entities, invoke the `persist` method of `EntityManager`. If the `persist` method of `EntityManager` is invoked and the `persist` processing is cascaded, the entity is managed in the persistence and persistence context of the database.

The following table describes the state of the entity after the `persist` operation, for each entity state.

Table 9–8: State of the entities in the persist operation

State of the entity	State of the entity after the persist operation
`new`#	The state changes to `managed`. The entity changed to `managed` is added into the database during or before transaction commit or is added into the database as a result of the execution of the `flush` operation.
`managed`	The `persist` operation is ignored. However, if `PERSIST` or `ALL` is specified in the `cascade` attribute for the relationship from one entity to another entity, the `persist` operation is propagated to the entities referenced by this entity.

State of the entity	State of the entity after the persist operation
detached#	If the line corresponding to the entity does not exist in the database, the state changes to managed. If the corresponding line exists, EntityExistsException occurs.
removed	The state changes to managed.

#: For Cosminexus JPA Provider, when the state of an entity is new or detached, the results of the transition of entity states differ according to whether the data corresponding to the entity exists on the database. Note the following for Cosminexus JPA Provider:

- If an entity that is not perpetuated in the database is specified in the argument, the state changes to managed.
- If a line duplicating with the entity passed in an argument exists on the database and if that entity is managed with the persistence context, EntityExistsException occurs during the persist processing.
- If a line duplicating with the entity passed in an argument exists on the database, but if the entity is not managed with the persistence context, EntityExistsException or another PersistenceException occurs during flush and commit.

Important note

The entity is registered in the persistence context when the entity is perpetuated in the database and when the entity information is read from the database. Note that after the entity state is changed to managed, if the persistence processing is rolled back, the entity is not managed with the persistence context.

9.4.3 remove operation for the entities

To execute the remove operation for the entities, invoke the remove method of EntityManager. If the remove method of EntityManager is invoked and the remove processing is cascaded, the state of the entity becomes removed. A removed entity is deleted from the database by the transaction commit processing.

The following table describes the state of the entity after the remove operation, for each entity state.

Table 9–9: State of entity in the remove operation

State of the entity	Result of state transition
new	The remove operation is ignored. However, if REMOVE or ALL is specified in the cascade attribute for the relationship from one entity to another entity, the remove operation is propagated to the entities referenced by this entity.
managed	The state changes to removed. If REMOVE or ALL is specified in the cascade attribute for the relationship from one entity to another entity, the remove operation is propagated to the entities referenced by this entity.
detached	The state changes as follows: • If the line corresponding to the detached entity passed in the argument exists in the database, IllegalArgumentException is thrown during the remove operation. • For Cosminexus JPA Provider, if the corresponding line does not exist in the database, the remove operation is ignored. However, if REMOVE or ALL is specified in the cascade attribute for the relationship with other entities, the remove operation is propagated to the entities referenced by this entity.
removed	The remove operation is ignored. The operation is also not propagated to other entities.

Note 1: A removed entity is deleted from the database as a result of the execution of operations during transaction commit, before transaction commit, or during flush operations.

Note 2: After the entity is deleted, the entity contents change to the contents immediately after the remove processing was invoked, however, excluding the content immediately after the entity was generated.

9.4.4 Obtaining the entities from the database

By invoking the `find` method or `getReference` method of `EntityManager`, you can obtain the entities corresponding to the primary key specified in the argument, from the database.

The entities returned by the `find` method or `getReference` method of `EntityManager` use the persistence context of the transaction scope. Therefore, when the method is invoked within the transaction, the state changes to `managed`. Also, if the `find` method or `getReference` method is invoked outside the transaction, the state changes to `detached`.

The following table describes the state of entities obtained with the `find` operation or `getReference` operation.

Table 9–10: State of entities obtained with the find operation or getReference operation

Persistence context	State of the obtained entity
Persistence context of the transaction scope (outside the transaction)	`detached`
Persistence context of the transaction scope (within the transaction)	`managed`
Extended persistence context (outside the transaction)	`managed`
Extended persistence context (within the transaction)	`managed`

Note that for Cosminexus JPA Provider, the following points differ from the JPA specifications in the `find` operation or `getReference` operation. Note that apart from these differences, there are no other functionality differences with the JPA specifications.

- With the JPA specifications, in the `getReference` method of `EntityManager`, you can return `Proxy` to the entity corresponding to the primary key given in the argument and not in the actual instance. However, with Cosminexus JPA Provider, Lazy loading is not supported for `@Basic`. Therefore, with Cosminexus JPA Provider, entity instance is returned as the return value of `getReference` instead of `Proxy`.

  For details on the supported range of Lazy loading for a relationship, see *9.4.5(2) Reading the entity information from the database*.
- With Cosminexus JPA Provider, if an entity that does not exist between the `find` method and `getReference` method is specified in the argument, a null value is returned with the `find` method. Also, `EntityNotFoundException` is thrown with the `getReference` method.

9.4.5 Synchronization with the database

When the transaction commit or `flush` method is executed, the state of the entity is written to the database. On the other hand, unless `refresh` is invoked explicitly, the state of the entity loaded in the memory is not refreshed.

This section describes writing of the entity information to the database and reading of the entity information from the database.

(1) Writing the entity information to the database

This section describes the transition of entity states in the `flush` operation or transaction commit. The following table describes the state transition results for each entity A state.

Table 9–11: State transition of entity instances in the flush operation or transaction commit

State of entity A	Result of state transition
`new`	The `flush` operation is ignored.
`managed`	Entity A is synchronized with the database.
`removed`	Entity A is deleted from the database.
`detached`	The `flush` operation is ignored.

Also, if entity A with the `managed` state has a relationship to entity B, the `persist` operation is cascaded according to the conditions described in the following table by extending the `flush` processing.

Table 9–12: Cascading of the flush processing and persist operation in commit for the related entity B

Specification of the cascade attribute of the relationship to entity B	State of entity B	Results
`PERSIST` or `ALL` is specified	--	The `persist` operation is cascaded to entity B.
`PERSIST` or `ALL` is not specified	`new`	• In the `flush` operation `IllegalStateException` occurs and the transaction is marked for rollback. • In the commit processing `IllegalStateException` occurs and the transaction commit operation fails.
	`managed`	Entity B is synchronized with the database.
	`removed`	• In the flush operation `IllegalStateException` occurs and the transaction is marked for rollback. • In the commit processing `IllegalStateException` occurs and the transaction commit operation fails.
	`detached`	• When entity A is the owner of the relationship The change in the relationship is synchronized with the database. • When entity B is the owner of the relationship An exception occurs. With Cosminexus JPA Provider, the operations in this case are not supported.

Legend:

--: Not applicable

Note that if the `flush` method is invoked outside the transaction, `TransactionRequiredException` occurs.

Tip

Persistence relation for relationship and database

- A `managed` entity having a bi-directional relationship is perpetuated on the basis of the reference stored in the owner side of the relationship. The application developer must create an application so that when changes occur, the entity is stored in the owner side and the non-owner side respectively without conflicting with the state of the entity on the memory.

- For unidirectional OneToOne and OneToMany, it is the developer's responsibility to guarantee that the relation of the relationships defined in the entity and the relation between the database tables is matching.

(2) Reading the entity information from the database

If the `refresh` method of `EntityManager` is invoked, the changes performed in the entity until then are destroyed and the state of the entity is overwritten by the database contents. At this time, if the corresponding line does not exist in the database, `EntityNotFoundException` occurs.

If you invoke the `refresh` method of `EntityManager` when the state of entity is not `managed`, `IllegalArgumentException` occurs.

(a) Timing when the entity information is read from the database

The entity is read from the database when the `refresh` method or `find` method is executed or when a query is issued. The related entities can also be read at this time. This is called the *fetch strategy*. Specify the fetch strategy in the `fetch` attribute of each relationship. Specify one of the following types in the `fetch` attribute:

- **FetchType.EAGER**
  When the entity information is read from the database, the related field and entity information is read.
- **FetchType.LAZY**
  When the field or relation destination is accessed for the first time, the entity is read from the database. This is called *Lazy loading*.

If you specify `FetchType.EAGER`, the related field and entity information is read every time the entity information is read from the database. Therefore, specify `FetchType.LAZY` to prevent the obtaining of unnecessary relation destination entities.

The following table describes the supported range of the `fetch` attribute in Cosminexus JPA Provider.

Table 9–13: Supported range of the fetch attribute for each relationship

Relationship annotation	Supported range
`@ManyToMany`	The default value is `FetchType.LAZY`.
`@OneToMany`	The default value is `FetchType.LAZY`.
`@OneToOne`	The default value is `FetchType.EAGER`. Note that if LAZY is specified, see *(b)*.
`@ManyToOne`	The default value is `FetchType.EAGER`. Note that if LAZY is specified, see *(b)*.
`@Basic`	The `fetch` attribute is ignored. The default `FetchType.EAGER` is always applied.

(b) LAZY fetch in @OneToOne and @ManyToOne

If you specify `LAZY` in the fetch strategy for a `@OneToOne` and `@ManyToOne` relationship, when the entity class is loaded, the binary code is embedded in the `getter` method for the field specifying `LAZY`.

If you invoke the `getter` method, the relation destination entity is obtained from the database through the processing of the embedded binary code. Because the binary code is embedded in the `getter` method, the `getter` method used for accessing the field that forms the target of relation must be set up. Furthermore, `@OneToOne` or `@ManyToOne` must be specified in that `getter` method.

Important note

The `getter` method determines the type of the relation destination entity from the `targetEntity` type of the relationship. The type of class specified in `targetEntity` must be castable in the field or property type.

9.4.6 Separate and merge operations of an entity from the persistence context

An entity separated from the persistence context is called a *detached entity*. The entity state becomes `detached` at the following times:

- When a transaction in the persistence context of the transaction scope is committed
- When a transaction is rolled back
- When the persistence context is cleared (when `EntityManager.clear()` is invoked)
- When `EntityManager` is closed
- When the entity is serialized and the entity value is passed

The instance of the separated entity continues to exist outside the persistence context that is perpetuated and obtained. The states of the entity and the database are not synchronized.

(1) Accessing a detached entity

An application can access the `detached` entity even after the persistence context ends. In this case, the entity fields and relation destinations must already be fetched. The fields and the relation destination entities that are not fetched by the `detached` entity cannot be accessed. Note that `FetchType.EAGER` is always applied to `@Basic` specified in the field, so the relation destination entity and field information are already obtained.

To access a relationship from the `detached` entity, one of the following conditions must be satisfied:

- The entity instance is obtained using `find()`
- The entity is obtained by using a query or the entity is explicitly requested using the `FETCH JOIN` clause
- The persistence instance that is not a primary key is already accessed with the application
- The entity is already obtained from another valid entity by tracing the relation specified as `fetch=EAGER`

If unavailable instances are accessed and if available instances are accessed in a disabled state, an exception occurs. However, when `FetchType.LAZY` is specified with Cosminexus JPA Provider, if you access un-fetched fields and relation destination entities even if the entity is detached after `EntityManager` terminates, sometimes the value is obtained from the database and the contents can be referenced.

(2) merge processing of an entity

By invoking the `merge` method of `EntityManager` or by cascading the `merge` processing, you can merge the `detached` entity with the persistence context managed by `EntityManager`.

The following table describes the transition of entity states in the `merge` processing for each state of entity A.

Table 9–14: Results of transition of entity states in the merge processing

State of entity A	Results of state transition
new	A new entity A' is created with managed state and the state of entity A is copied to entity A'. Note that the entity of the merge argument remains new.
managed	The merge operation is ignored. However, if MERGE or ALL is specified in the cascade attribute for the relationship from entity A to other entities, the merge operation is propagated to the entities referenced by entity A.
detached	The state of entity A is copied to entity A' that has the same ID and already exists and is being managed, or a new managed entity is created that is a copy of entity A. Note that the entity of the merge argument remains detached.
removed	IllegalArgumentException is thrown by the merge operation, or transaction commit fails. Note that the state of entity A remains removed.

Note 1: If the relation from entity A to entity B is specified with cascade=MERGE or cascade=ALL, all the entity B are recursively merged as entity B'. Entity B' is set in entity A'. Note that if entity A is in the managed state, entity A and entity A' are the same.

Note 2: If entity B is referenced when cascade=MERGE or cascade=ALL is not specified in the relation of entity A, when entity A is merged with entity A' and if you trace the relation from entity A', you will finally reach the reference of the managed entity B' with the same persistence identity as entity B.

With the JPA specifications, the fields marked as LAZY to imply that the field is not fetched, are ignored during merge. However, with Cosminexus JPA Provider, @Basic operates as EAGER, therefore, all the fields that are not relationships are subject to the merge processing.

Also, if the Version string is used in the entity, the version of the entity is checked during the merge operation and during the flush and commit processing invoked after the merge operation. If the Version string does not exist, the version of the entity is not checked with the merge operation. For details, see *9.10.1 Optimistic lock processing*.

Note that Cosminexus JPA Provider does not support the processing to return to the persistence context using the entity merge processing between vendors.

(3) Notes

- In Cosminexus JPA Provider, the managed entity becomes a detached entity due to transaction commit with the persistence context of the transaction scope. On the other hand, with the extended persistence context, the managed entity remains managed.
- If the transaction is rolled back in the transaction scope and in the extended persistence context, all the existing managed instances and removed instances become detached. The state of the instance is the state existing when the transaction is rolled back.
- If the transaction is rolled back, the state of the persistence context becomes the state existing at rollback, so the state conflicts with the database state. Note that in Cosminexus JPA Provider, the version attribute state and the generated state form conflicting states, therefore, if the merge operation is performed, an exception might occur.

9.4.7 managed entity

You can use the contains() method of EntityManager to obtain information about whether the entity instance is being managed with the current persistence context.

This section describes the conditions for the return value of the contains() method.

- **Conditions when the contains() method returns true**

    - When the entity is acquired from the database and is not deleted from `EntityManager`, or is not separated
    - When the entity instance is generated and the `persist` method is executed for that entity or the `persist` operation is propagated to that entity
- **Conditions when the contains() method returns false**
    - When the entity instance is separated
    - When the `remove` method is executed for the entity or the `remove` operation is propagated to the entity
    - When the entity instance is generated and the `persist` method is not executed for that entity or the `persist` operation is not propagated to that entity

The actual `insert` and `delete` processing in the database is delayed until the conclusion of the transaction. At the same time, note that the propagation of `persist` and `remove` is applied immediately with the `contains` method.

Make sure that the entity instance is only managed using a single persistence context in the application. With Cosminexus JPA Provider, the operations do not function properly when the same Java instance is managed with multiple persistence contexts.

9.5 Defining the mapping information between the database and Java objects

With Cosminexus JPA Provider, you can define the information for mapping the database and the Java objects. You define the mapping information with an annotation or the O/R mapping file.

- **Definition using an annotation**
  You define the mapping information directly in the entity class of the application.
- **Definition using an O/R mapping file**
  An O/R mapping file is an XML file used for describing the mapping information. You use tags to define the mapping information.

If the mapping information is defined in both annotations and O/R mapping file, the definition in the O/R mapping file is given priority. Therefore, if you use the O/R mapping file to change the mapping information defined by using annotations, you can change the mapping information without changing the application.

Determine whether you want to use an annotation or an O/R mapping file or you want to use the annotation and O/R mapping file together, in accordance with the application creation policy.

9.6 Entity relationships

With an entity, the relation between the database tables can be expressed using a relationship. With Cosminexus JPA Provider, you can set an entity relationship.

9.6.1 Relationship types

A relationship consists of a relation and a direction. The possible relations are OneToOne, ManyToOne, OneToMany, and ManyToMany. Furthermore, the possible directions of each relation are unidirectional and bi-directional. The relations and directions are combined to form the following seven types of relationship:

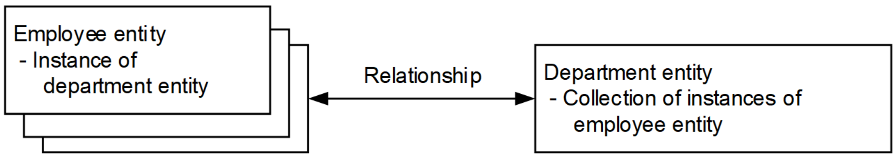
- Unidirectional OneToOne relationship
- Unidirectional ManyToOne relationship
- Unidirectional OneToMany relationship
- Unidirectional ManyToMany relationship
- Bi-directional OneToOne relationship
- Bi-directional ManyToOne/OneToMany relationship
- Bi-directional ManyToMany relationship

This section describes relationships using the employees and departments of a company as an example. The relation between the employees and departments is as follows:

- The entity expresses the employees and departments respectively.
- The employees are assumed to belong to some department.
- A department maintains multiple employees.
- The departments are referenced from employees and the employees are referenced from departments.

This example expresses the bi-directional ManyToOne/ OneToMany relationship of entities. In this case, Many stands for the employees and One stands for the department. The following figure shows the relation between the employee entity and the department entity.

Figure 9–4: Relation between the employee entity and the department entity


Employee entity
- Instance of department entity
Relationship
Department entity
- Collection of instances of employee entity

Note that in an entity, you can specify settings to propagate the operations to the relation destination entity. These settings are called *cascade*. When you perform an operation for an entity and if cascade is specified, similar operations are automatically executed even in the entities that have a relationship with the operated entity. If cascade is used, the user can save the effort of performing operations for the relation destination entity.

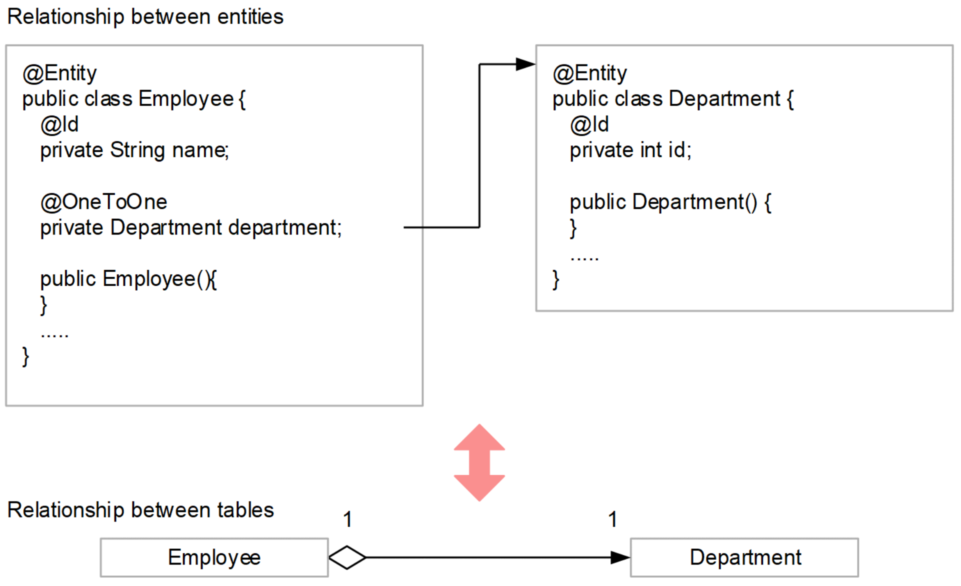
9.6.2 Annotations for relationships

The entities handle the relation between tables as a relationship. When you handle a relationship with an entity, the reference to another entity is stored as a field in the entity. In addition, set the following relationship annotations in the persistence property or instance variable that references an entity:

- OneToOne (One-to-one)
- OneToMany (One-to-many)
- ManyToOne (Many-To-One)
- ManyToMany (Many-To-Many)

The following figure shows the relationship between entities and the relation between tables.

Figure 9–5: Relationship between entities and relation between tables



```
@Entity
public class Employee {
   @Id
   private String name;

   @OneToOne
   private Department department;

   public Employee(){
   }
   .....
}
```

```
@Entity
public class Department {
   @Id
   private int id;

   public Department() {
   }
   .....
}
```

If the relationship annotation is not specified for referencing the entity, the operations will not function properly. Note that if the `generic` type is not used in the reference where collection is used, the target entity must be specified as the relationship target.

Instead of using an annotation, you can also define a relationship using the O/R mapping file. However, note that when a relationship is defined in the O/R mapping file and if the same definition is specified in the annotation, the definition in the annotation is overwritten.

Default mapping in the relationship annotations

When the annotations for the OneToOne, OneToMany, ManyToOne, and ManyToMany relationships are applied, the default mapping rules are applied. Similarly when a relationship is specified in the O/R mapping file, the default mapping is applied.

When you use the default mapping of the relationship annotations, if the database tables and relations differ from the default values specified in *9.6.4 Default mapping (bi-directional relationship)* or *9.6.5 Default mapping (unidirectional relationship)*, the SQL statement cannot be created correctly in the database query during execution. An exception occurs.

9.6.3 Direction of relationships

A relationship includes a bi-directional relationship and a unidirectional relationship. When a bi-directional relationship is handled, a relationship has an owner and a non-owner. A unidirectional relationship only has an owner. The owner of a relationship can take decisions on updating a database relationship.

The following rules are applied to a bi-directional relationship:

- The non-owner of a bi-directional relationship specifies the owner based on the `mappedBy` element of `@OneToOne`, `@OneToMany`, and `@ManyToMany`. With the `mappedBy` element, you specify the properties or field names that reference the non-owner side using the owner entity.
- With the ManyToOne/OneToMany bi-directional relationship, set `many` as the owner. Therefore, the `mappedBy` element cannot be specified in `@ManyToOne`.
- With the OneToOne bi-directional relationship, the entity on the side containing the external key becomes the owner.
- In the ManyToMany bi-directional relationship, any entity might be set as the owner.

A relationship annotation has a `cascade` attribute. If you specify the `cascade` attribute, you can propagate the operations for the entities even for the reference destination entities. However, you can specify `REMOVE` in the `cascade` attribute of the relationship annotation only for OneToOne or OneToMany. If `cascade=REMOVE` is applied for other relations, the operations might not function properly. For details on the `cascade` attribute when the entities have relationships, see *9.4.1(4) Propagation of operations to the entities*.

Important note

Cosminexus JPA Provider does not implement the check for maintaining the consistency of relationships during execution. Therefore, when you update the relationships during the execution of an application, even if the update causes an inconsistency in the relationships, no warning or exception occurs.

Note that when a value is fetched from the database with collection relationships such as OneToMany or ManyToMany, an empty collection is returned as the relationship value if the related entity does not exist.

9.6.4 Default mapping (bi-directional relationship)

This section describes the default mapping of a bi-directional relationship.

(1) Bi-directional OneToOne relationship

This section describes the default mapping of a bi-directional OneToOne relationship applied in the following conditions:

Conditions

- Entity A references the stand-alone instance of entity B and sets `@OneToOne`.
- Entity B references the stand-alone instance of entity A and sets `@OneToOne`. The persistence property (or field) name that references entity B using entity A is specified in the `mappedBy` attribute of `@OneToOne`.
- Entity A is the relationship owner.

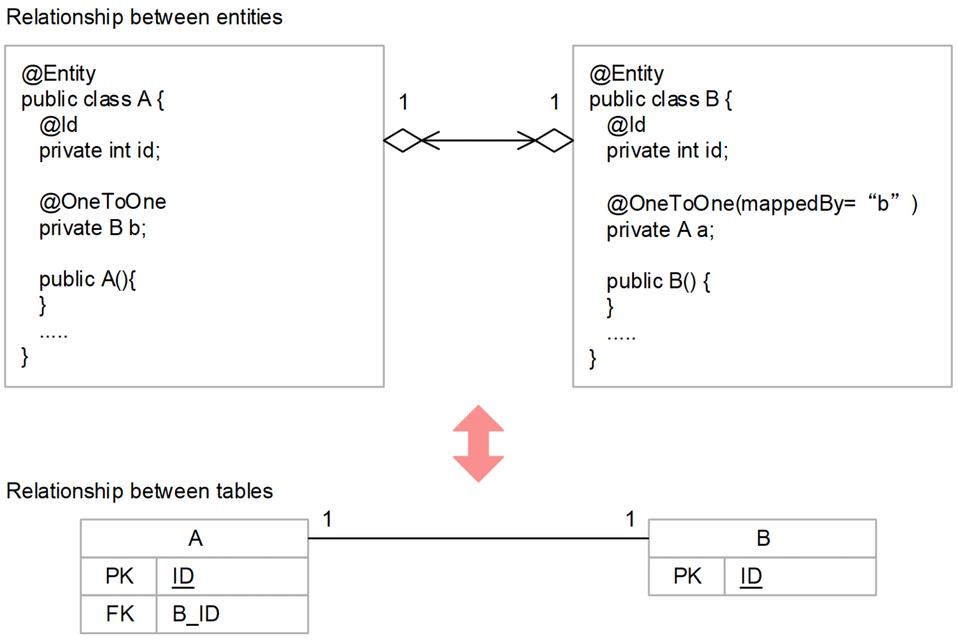
Applied default mapping

- Entity A is mapped to table A.
- Entity B is mapped to table B.

- Table A must have the external key for table B. The name of the external key string is as follows:
  Name of the external key string
  *Name of the relationship property (or field) of entity A*_ *Name of the primary key string of table B*
  Note: Italics indicate a variable value.

Also, the external key string has the same type as the primary key of table B and is a unique key constraint.

Figure 9–6: Default mapping in a bi-directional OneToOne relationship



(2) Bi-directional ManyToOne/ OneToMany relationship

This section describes the default mapping of a bi-directional ManyToOne/OneToMany relationship applied in the following conditions:

Conditions

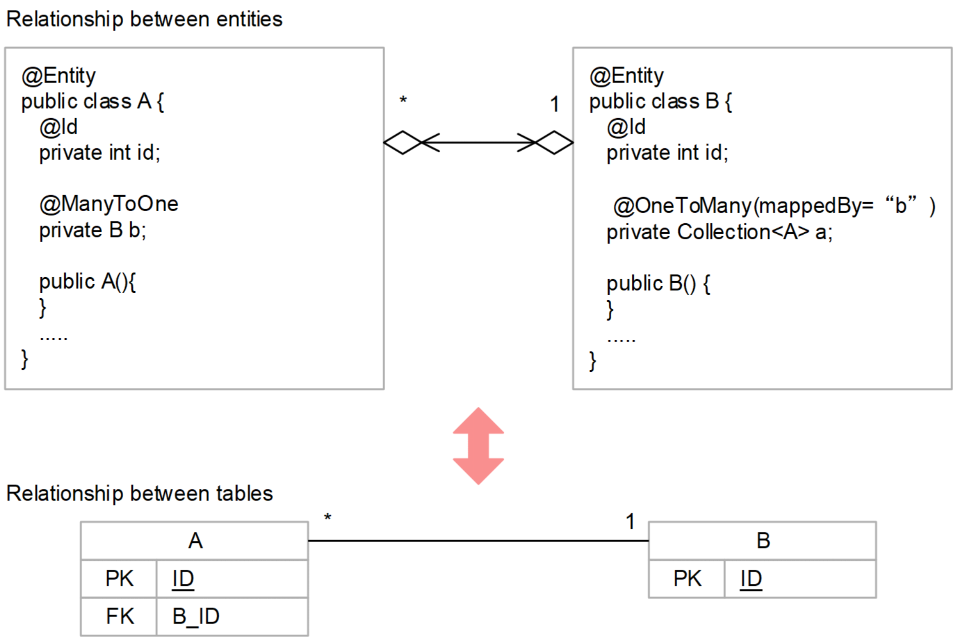
- Entity A references the stand-alone instance of entity B and sets `@ManyToOne` (or the corresponding XML tags in the O/R mapping file).
- Entity B references the entity A collection and sets `@OneToMany` (or the corresponding XML tags in the O/R mapping file). The `mappedBy` attribute is specified in `@OneToMany`. The `mappedBy` attribute specifies the persistent property (or field) name set for referencing entity B with entity A.
- Entity A is the relationship owner.

Applied default mapping

- Entity A is mapped to table A.
- Entity B is mapped to table B.
- Table A must have the external key for table B. The name of the external key string is as follows:
  Name of the external key string
  *Name of the relationship property (or field) of entity A*_ *Name of the primary key string of table B*
  Note: Italics indicate a variable value.

The external key string has the same type as the primary key of table B.

Figure 9–7: Default mapping in a bi-directional ManyToOne/ OneToMany relationship



(3) Bi-directional ManyToMany relationship

This section describes the default mapping of a bi-directional ManyToMany relationship applied in the following conditions:

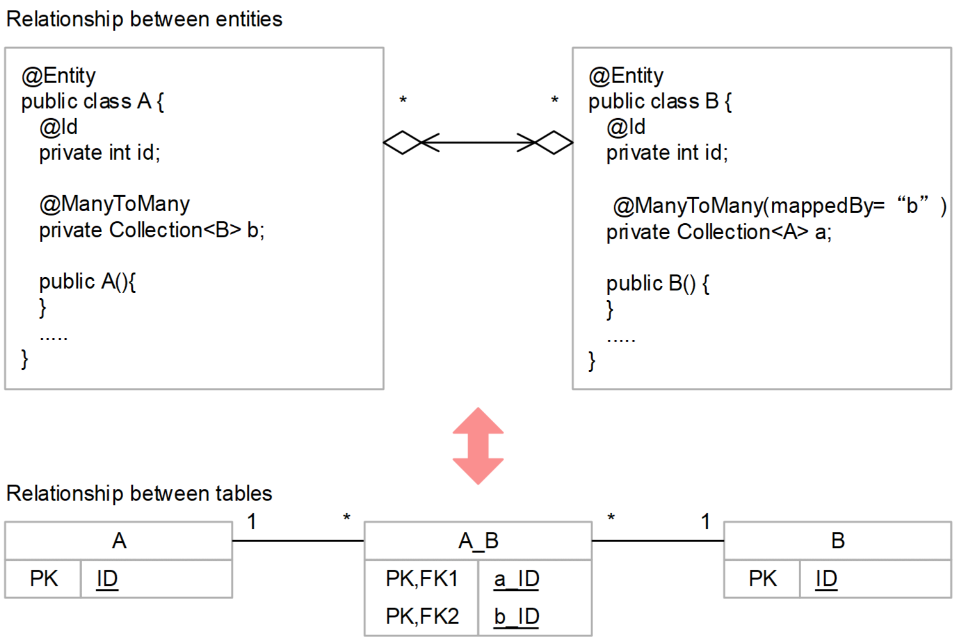
Conditions

- Entity A references the entity B collection. `@ManyToMany` (or the corresponding XML element in the O/R mapping file) is set in the collection.
- Entity B references the entity A collection. `@ManyToMany` (or the corresponding XML tags in the O/R mapping file) is set in the collection and the `mappedBy` attribute is specified. The `mappedBy` attribute specifies the persistent property (or field) name set for referencing entity B with entity A.
- Entity A is the relationship owner.

Applied default mapping

- Entity A is mapped to table A.
- Entity B is mapped to table B.
- Apart from table A and B, a junction table named A_B where the name of the owner table appears at the beginning, is necessary. This junction table has two external key strings.
  The first external key string references table A and has the same type as the primary key of table A. The name of this external key string is as follows:
  Name of the external key string
  *Name of the relationship property (or field) of entity B*_*Name of the primary key of table A*
  The other external key string references table B and has the same type as the primary key of table B. The name of this external key string is as follows:
  Name of the external key string
  *Name of the relationship property (or field) of entity A*_*Name of the primary key of table B*
  Note: Italics indicate a variable value.

Figure 9–8: Default mapping in a bi-directional ManyToMany relationship



```
@Entity
public class A {
  @Id
  private int id;

  @ManyToMany
  private Collection<B> b;

  public A(){
  }
  .....
}
```

```
@Entity
public class B {
  @Id
  private int id;

   @ManyToMany(mappedBy= “b” )
  private Collection<A> a;

  public B() {
  }
  .....
}
```

9.6.5 Default mapping (unidirectional relationship)

A unidirectional relationship includes a Single-Valued relationship and a Multi-Valued relationship. The following points describe each relationship:

- **Unidirectional Single-Valued relationship**

  A unidirectional Single-Valued relationship is a relationship that references the stand-alone instance and where only the owner exists.

  Possible unidirectional Single-Valued relationships are unidirectional OneToOne relationships and unidirectional ManyToOne relationships.

- **Unidirectional Multi-Valued relationship**

  A unidirectional Multi-Valued relationship is a relationship that references the entity in the collection format and where only the owner exists.

  Possible unidirectional Multi-Valued relationships are unidirectional OneToMany relationships and unidirectional ManyToMany relationships.

This section describes the default mapping of a unidirectional relationship.

(1) Unidirectional Single-Valued relationship

This section describes the default mapping of a unidirectional OneToOne relationship and a unidirectional ManyToOne relationship.

(a) Unidirectional OneToOne relationship

This section describes the default mapping of a unidirectional OneToOne relationship applied in the following conditions:

Conditions

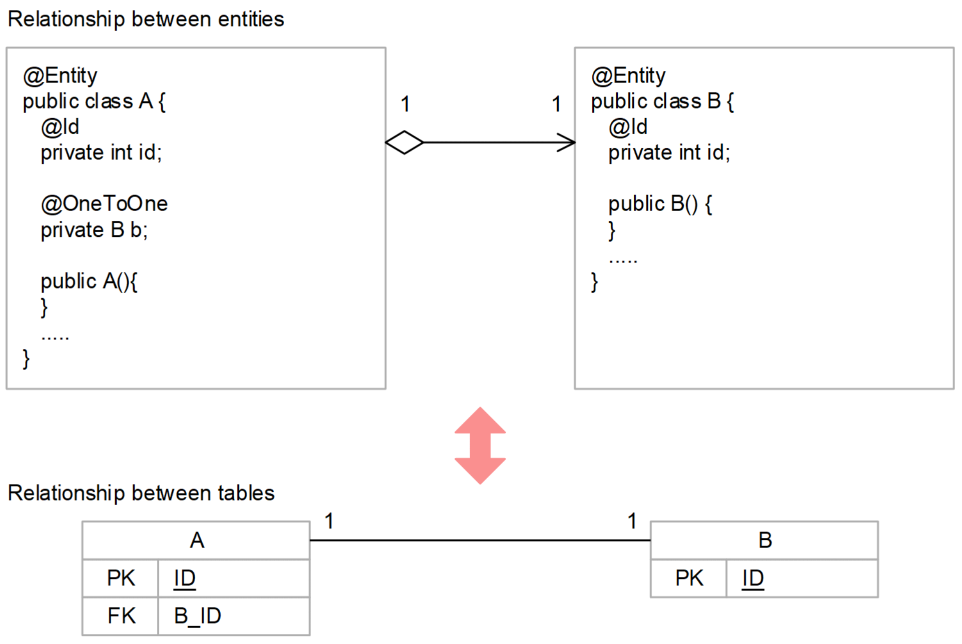
- Entity A references the stand-alone instance of entity B and sets `@OneToOne` (or the corresponding XML tags in the O/R mapping file).
- Entity A is not referenced from entity B.
- Entity A is the relationship owner.

Applied default mapping

- Entity A is mapped to table A.
- Entity B is mapped to table B.
- Table A must have the external key for table B. The name of the external key string is as follows:
  Name of the external key string
  *Name of the relationship property (or field) of entity A*_ *Name of the primary key string of table B*
  Note: Italics indicate a variable value.

The external key string has the same type as the primary key of table B and is a unique key constraint.

Figure 9–9: Default mapping in a unidirectional OneToOne relationship



(b) Unidirectional ManyToOne relationship

This section describes the default mapping of a unidirectional ManyToOne relationship applied in the following conditions:

Conditions

- Entity A references the stand-alone instance of entity B and sets `@ManyToOne` (or the corresponding XML tags in the O/R mapping file).
- Entity A is not referenced from entity B.

Applied default mapping

- Entity A is mapped to table A.
- Entity B is mapped to table B.
- Table A must have the external key for table B. The name of the external key string is as follows:

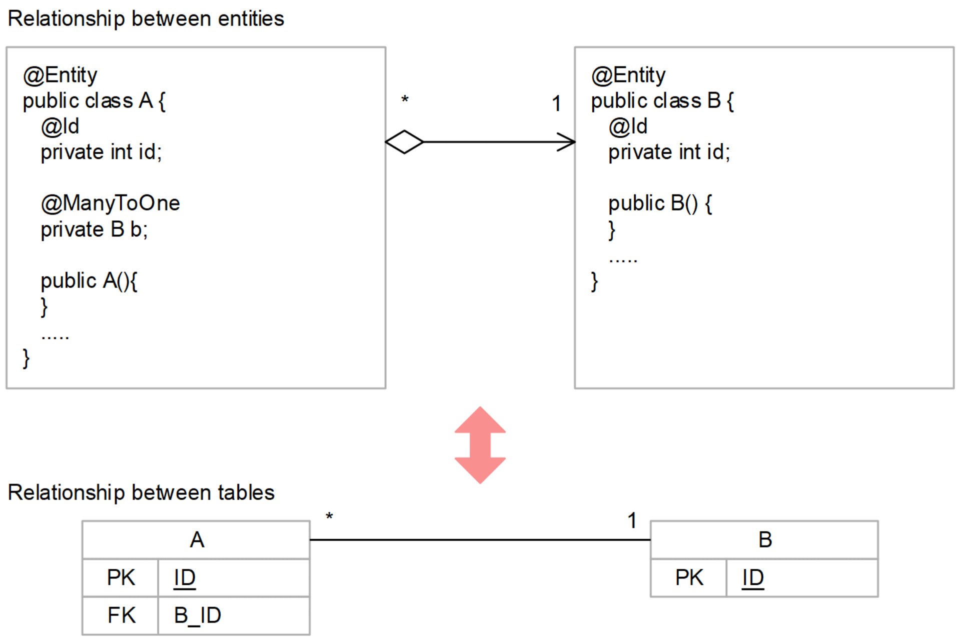
Name of the external key string

Name of the relationship property (or field) of entity A_ Name of the primary key string of table B

Note: Italics indicate a variable value.

The external key string must have the same type as the primary key of table B.

Figure 9–10: Default mapping in a unidirectional ManyToOne relationship



(2) Unidirectional Multi-Valued relationship

This section describes the default mapping of a unidirectional OneToMany relationship and a unidirectional ManyToMany relationship.

(a) Unidirectional OneToMany relationship

This section describes the default mapping of a unidirectional OneToMany relationship applied in the following conditions:

Conditions

- Entity A references the entity B collection. `@OneToMany` (or the corresponding XML tags in the O/R mapping file) is set in the collection.
- Entity A is not referenced from entity B.
- Entity A is the relationship owner.

Applied default mapping

- Entity A is mapped to table A.
- Entity B is mapped to table B.
- Apart from table A and B, a junction table named A_B where the name of the owner table appears at the beginning, is necessary. This junction table has two external key strings.

  The first external key string references table A and has the same type as the primary key of table A. The name of this external key string is as follows:

  Name of the external key string

  Name of entity A_ Name of the primary key string of table A

The other external key string references table B, has the same type as the external key of table B, and is a unique key constraint. The name of this external key string is as follows:

Name of the external key string

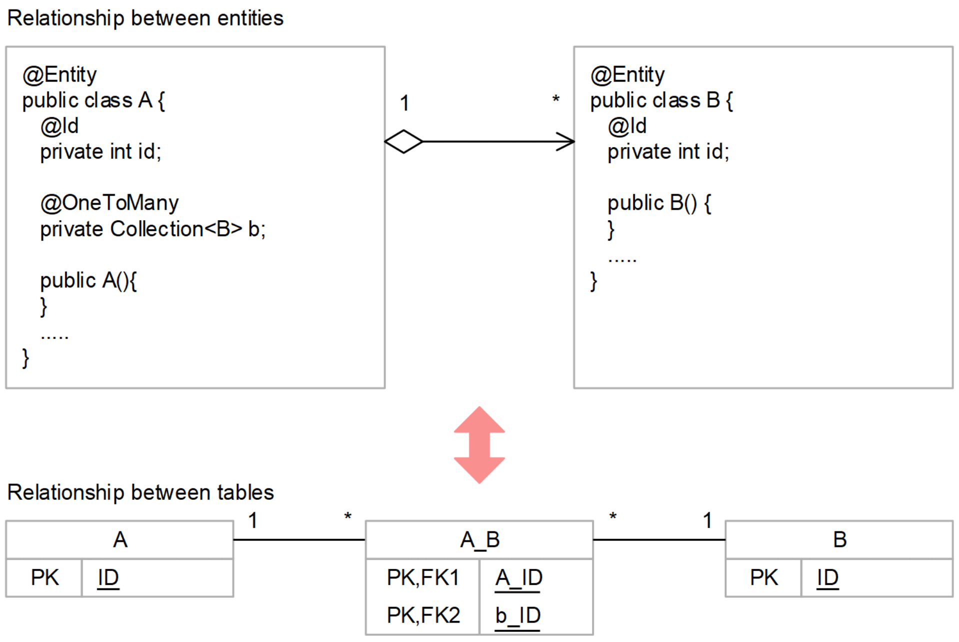
*Name of the relationship property (or field) of entity A*_ *Name of the primary key string of table B*

Note: Italics indicate a variable value.

Important note

A unidirectional OneToMany relationship that uses a junction table as described in this example might not be supported with JPA Providers other than Cosminexus JPA Provider. Note this when you operate applications created with a unidirectional OneToMany relationship using JPA Providers other than Cosminexus JPA Provider.

Figure 9–11: Default mapping in a unidirectional OneToMany relationship



(b) Unidirectional ManyToMany relationship

This section describes the default mapping of a unidirectional ManyToMany relationship applied in the following conditions:

Conditions

- Entity A references the entity B collection. `@ManyToMany` (or the corresponding XML tags in the O/R mapping file) is set in the collection.
- Entity B does not reference entity A.
- The owner is entity A.

Applied default mapping

- Entity A is mapped to table A.
- Entity B is mapped to table B.
- Apart from table A and B, a junction table named A_B where the name of the owner table appears at the beginning, is necessary. This junction table has two external key strings.

One of the external key strings references table A and has the same type as the external key of table A. The name of this external key string is as follows:

Name of the external key string

*Name of entity A*_*Name of the primary key of table A*

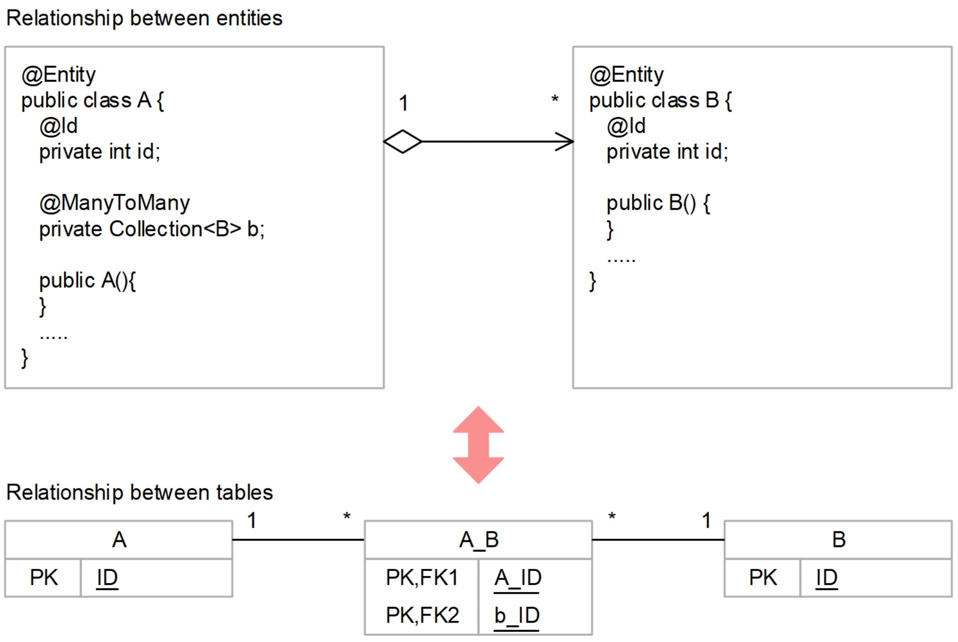
The other external key string references table B and has the same type as the primary key of table B. The name of this external key string is as follows:

Name of the external key string

*Name of the relationship property (or field) of entity A*_ *Name of the primary key of table B*

Note: Italics indicate a variable value.

Figure 9–12: Default mapping in a unidirectional ManyToMany relationship



```
@Entity
public class A {
   @Id
   private int id;

   @ManyToMany
   private Collection<B> b;

   public A(){
   }
   .....
}
```

```
@Entity
public class B {
   @Id
   private int id;

   public B() {
   }
   .....
}
```

9.7 Cache functionality of the entity objects

The cache functionality of the entity objects is a functionality that stores the entity objects used by an application in the memory. When you use the cache functionality of the entity objects, the entity objects cached in Cosminexus JPA Provider are used if the same entity objects are operated. The data is not read from the database again, so the access to the database is minimized and the load on the processing performance can be reduced. Note that this functionality is unique to Cosminexus JPA Provider.

This section describes the cache functionality of the entity objects.

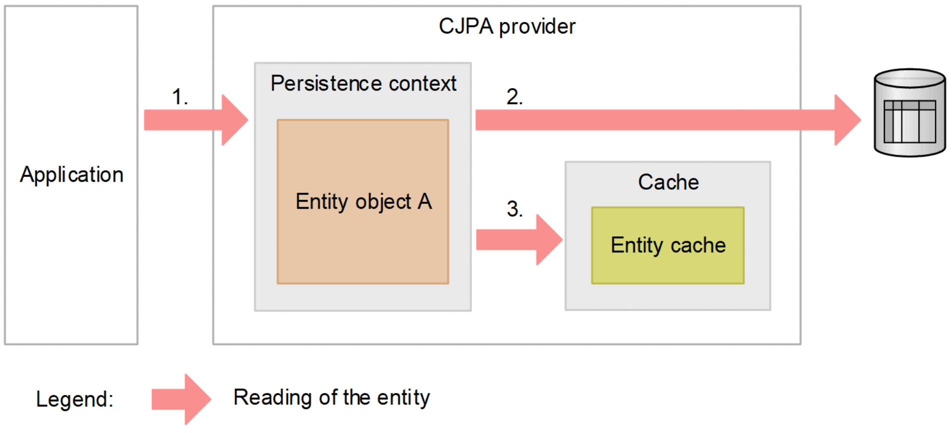
9.7.1 Processing of the cache functionality

If you use the cache functionality, when the same entity is read, the data is obtained from the cache instead of the database. This section describes the procedure for processing the cache functionality, the cache registration and update timing, and the procedure for updating the cache.

(1) Procedure for processing the cache functionality

The following figure shows the flow of processing of the cache functionality.

Figure 9–13: Flow of processing of the cache functionality



A description of the above figure is as follows:

- **Reading of the entity**
  When an entity is first read with methods such as `find`, the flow of processing is as follows:
  1. The entity is read.
  2. The data is obtained from the database.
  3. The entity object of the obtained data is registered in the cache.

When the entity has a relationship and when you obtain the relationship destination entity, if the target entity exists in the cache, the entity in the cache is referenced without accessing the database.

A cache exists for each persistence context.

(2) Cache registration and update timing

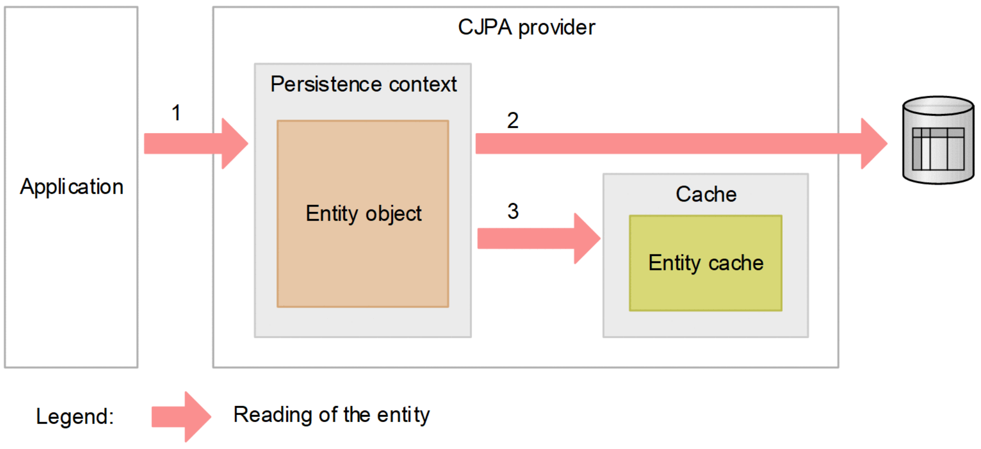
Registration and update in a cache is implemented at the following times:

- Cache registration timing
  During the reading of an entity object (`find` operation) when the target cache does not exist
- Cache update timing
  - When the entity is refreshed (`refresh` operation)
  - When a transaction is committed
  - When an exception occurs in the optimistic lock processing
    Note that when the `OptimisticLockException` exception occurs in the optimistic lock processing, the entity objects registered in the cache are deleted.

(3) Procedure for updating the cache

The following figure shows the flow of updating the cache.

Figure 9–14: Updating the cache



A description of the figure is as follows:

1. The following operations are implemented for the entity object:
   - `refresh` operation
   - Transaction commit
2. During the `refresh` operation, the database is accessed.
3. The data existing in the persistence context is registered in the cache and the cache data is updated.

Note that the cache is registered even when JPQL is executed. The cache is registered if JPQL is executed when the target cache does not exist. The cache data is used in JPQL as well, but the database is accessed irrespective of the presence or absence of the cache data. Therefore, the cache-based processing performance cannot be expected to improve. For details, see point *(4)*.

The cache stores information for the entity objects; therefore, if the object returned during the execution of the query is the entity itself, the cache is updated. The cache is not updated when other fields are specified. The JPQL examples when the cache update is valid and when the update is invalid are as follows:

- JPQL example when the cache is updated

```
SELECT emp FROM Employee AS emp
```

- JPQL example when the cache is not updated

```
SELECT emp.id, emp.name, emp.address FROM Employee AS emp
```

(4) Relation between JPQL and cache

When all the entity objects are obtained with JPQL, if there is information registered in the cache, the cache information is obtained. The cache functionality of Cosminexus JPA Provider requires a primary key that is the ID for identifying the target entity. To obtain the primary key, the JPQL result must be obtained and at this time, the database is accessed. The primary key is extracted from the database access results and the entity object is obtained from the cache. Also, if the target data does not exist in the cache, the entity is created from the database information.

With JPQL, the database is accessed regardless of the presence or absence of the cache data. Therefore, for JPQL, cache-based performance improvement cannot be expected.

9.7.2 Cache reference forms and cache types

There are three types of cache reference forms:

- **Hard reference**
  This reference is not collected by GC.
- **Weak reference (java.lang.ref.WeakReference)**
  If the reference is weakly reachable, the reference is collected by GC. Note that whether the cache reference is weakly reachable depends on the specifications in `java.lang.ref.WeakReference`.
  The examples of weak reference cache that are not collected with Cosminexus JPA Provider are as follows:
  - Cache of an entity object registered in the persistence context
  - Cache that references a non-weakly reachable cache using a relationship
  - Cache wherein the cache of another entity object with an inheritance relationship is not weakly reachable, when the entity inheritance strategy is used
- **Soft reference (java.lang.ref.SoftReference)**
  This is a reference form that caches out when the remaining amount of memory decreases.
  A soft reference is collected by GC depending on the consumption rate and survival time of a resource. The specifications such as the collection timing and the selection of objects for collection depend on JavaVM.

The cache types differ depending on which form is used to reference the cache. The following table describes the mapping of the cache reference forms and cache types.

Table 9–15: Mapping of the cache reference forms and cache types

Cache reference forms	Cache types
Hard reference	`Full`
Hard reference + Weak reference#1	`HardWeak`
Weak reference + Soft reference#1	`SoftWeak`
Weak reference	`Weak`
None#2	`None`

#1
The reference forms are combined.

#2

The entity object is not cached.

With Cosminexus JPA Provider, you can choose the cache type. Choose the type based on the application design and environment. Specify the cache type in `persistence.xml`. For details on `persistence.xml`, see *13.2 persistence.xml*.

The following points describe the cache types.

(1) Full

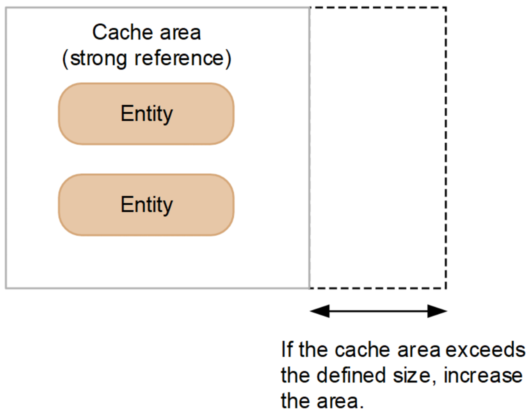
All the entities are cached with a hard reference.

If you specify `Full` in the cache type, the access to the database decreases, so the processing load decreases. However, the memory continues to be occupied, therefore, the load on the memory increases.

Specify `Full` when the duration of the entity object is long and when the reference is created for a few entity objects that require frequent access. Furthermore, when several entity objects are read, the memory load increases, so we do not recommend using `Full` to update multiple records in a batch.

When `Full` is specified, the hard reference area is allocated with the specified cache size. If the hard reference area exceeds the defined size, the area is increased based on the Hashtable specifications. The following figure shows the image in the cache when `Full` is specified.

Figure 9–15: Cache image for Full


Cache area
(strong reference)
Entity
Entity
If the cache area exceeds the defined size, increase the area.

(2) HardWeak

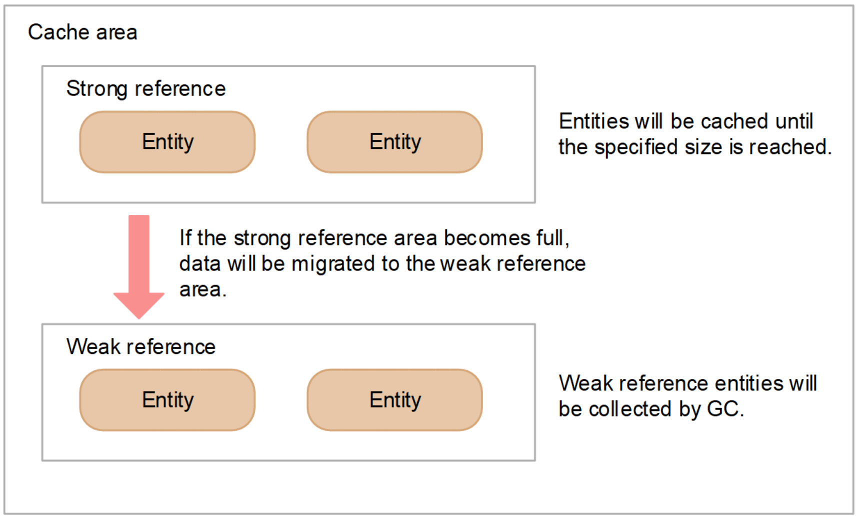
The entities are cached with a hard reference and weak reference.

When you want to store the entity object in a list, use a hard reference. Create the hard reference area with a fixed length only up to the value specified in the cache size. If the cache size reaches the specified value, the old entity objects are moved to the weak reference. At this time, the entity objects for which cache registration has not been used for the longest time are moved sequentially to the weak reference. If the entity objects moved to the weak reference are used, the entity objects are once again stored in the hard reference area.

If you specify `HardWeak`, you can use the entity objects with a long duration to efficiently control the memory used in the cache.

If the state of insufficient memory occurs frequently in a system where `SoftWeak` is used, you cannot take advantage of the soft reference; therefore, use `HardWeak`. The following figure shows the image in the cache for `HardWeak`.

Figure 9–16: Cache image for HardWeak


Cache area
Strong reference
Entity
Entity
Entities will be cached until the specified size is reached.
If the strong reference area becomes full, data will be migrated to the weak reference area.
Weak reference
Entity
Entity
Weak reference entities will be collected by GC.

In the case of `HardWeak`, the cache is stored with a hard reference in the hard reference cache area. Also, the cache is stored with a weak reference in the weak reference cache area.

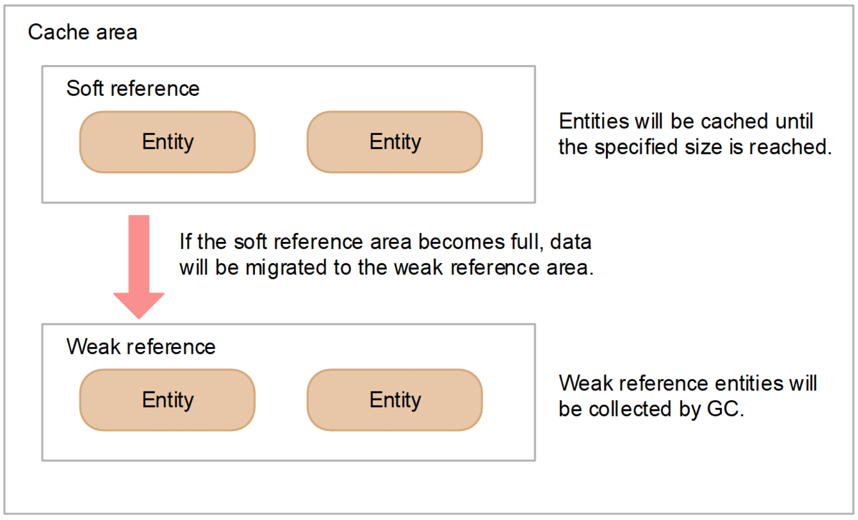
(3) SoftWeak

The entities are cached with a soft reference and weak reference.

You use the soft reference to store the entity objects in a list and create a soft reference area of a fixed length only up to the value specified in the cache size. If the cache size reaches the specified value, the old entity objects are moved to the weak reference area. At this time, the entity objects for which cache registration has not been used for the longest time are moved sequentially to the weak reference. If the entity objects moved to the weak reference are used, the entity objects are once again stored in the hard reference area.

If you specify `SoftWeak`, you can use the entity objects with a long duration to efficiently control the memory used in the cache. Therefore, when you use the cache functionality, we recommend that you specify `SoftWeak`. The following figure shows the image in the cache for `SoftWeak`.

Figure 9–17: Cache image for SoftWeak


Cache area
Soft reference
Entity
Entity
Entities will be cached until the specified size is reached.
If the soft reference area becomes full, data will be migrated to the weak reference area.
Weak reference
Entity
Entity
Weak reference entities will be collected by GC.

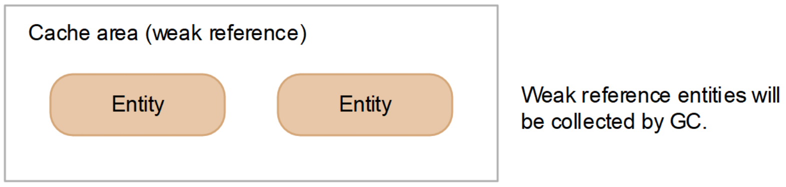
In the case of `SoftWeak`, the cache is stored with a soft reference in the soft reference cache area. Also, the cache is stored with a weak reference in the weak reference cache area.

(4) Weak

All the entities are cached with a weak reference.

Therefore, all the entity objects are subject to GC. If you specify weak, the memory load decreases, but the cache might be deleted due to GC. Use the `Weak` cache type in systems that do not place significance on the cache functionality of the entity objects. The following figure shows the image in the cache for `weak`.

Figure 9–18: Cache image for Weak


Cache area (weak reference)
Entity
Entity
Weak reference entities will be collected by GC.

In the `weak` cache type, the entities are cached with a weak reference.

(5) NONE

No entity objects are cached. Use the `None` cache type if you want to destroy an entity object immediately after the entity object is read from the database.

9.7.3 Scope of the cache functionality

The cache data is stored for each persistence context. Therefore, the data cannot be obtained from the cache even if the entity object is invoked in another persistence context.

Note that the cache data is stored for the duration from the generation of the persistence context until the persistence context is destroyed.

9.7.4 Notes on using the cache functionality

This section describes the notes on using the cache functionality of the entity objects.

(1) Notes on updating or deleting the data in a query

When JPQL and native query are used in the application to update and delete the data, the cache contents are not updated. Execute operations such as the `refresh` operation to obtain the database contents again.

The cache data is read in the following operations; therefore, the data is not updated in the database:

1. The data is read into the entity object.
   The entity data is registered in the cache as well.
2. The delete query containing the data read in 1 is executed.
   The data is deleted by the execution of the delete query, but the cache is not deleted.
3. The same data as in 1 is read into the entity object.
   Because the data is the same as 1, the data existing in the cache is read.

4. The data in 3 is flushed.
   The applicable line does not exist in the database, so the add or update processing cannot be performed.

(2) Notes on multiple persistence contexts using a cache

By using the cache, you can reduce the database access frequency. However, on the other hand, a time lag occurs in the data due to the cache and the frequency of optimistic lock exception might increase.

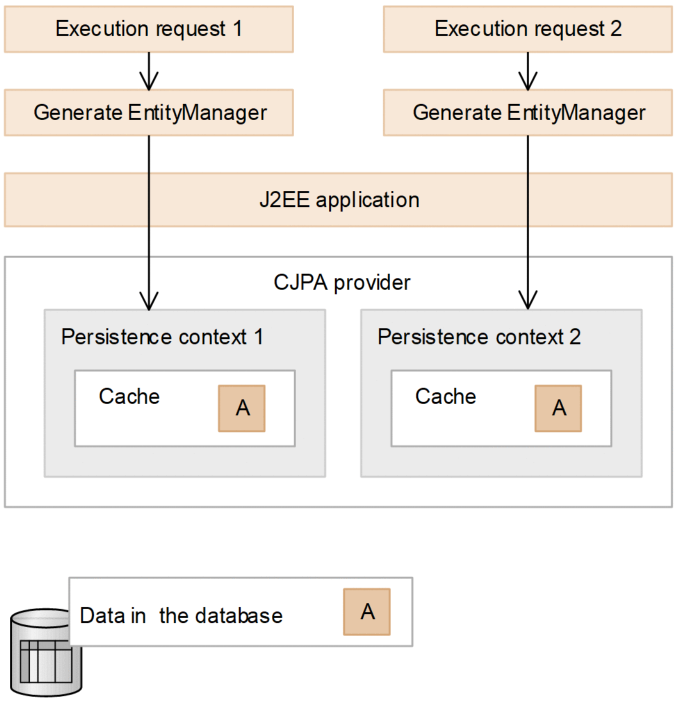
A cache exists for each persistence context. Therefore, multiple `EntityManagers` existing in pairs with the persistence contexts are generated simultaneously, and if the entities with the same primary keys are operated at the same time, even if the data is updated, the user might not be able to reference the updated data in a timely manner. Due to this, the optimistic lock exception occurs easily.

The following points describe the mechanism and action for the occurrence of the optimistic lock exception.

(a) Example of optimistic lock exception

This section describes an example of cache for each persistence context in the environment shown in the following figure.

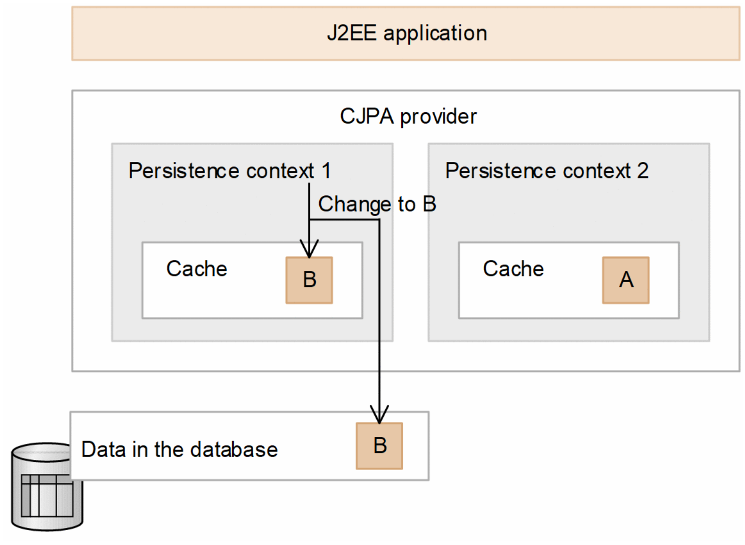
Figure 9–19: Environment described in this example



In the figure, the cache and the database are consistent and data A is already stored in the cache.

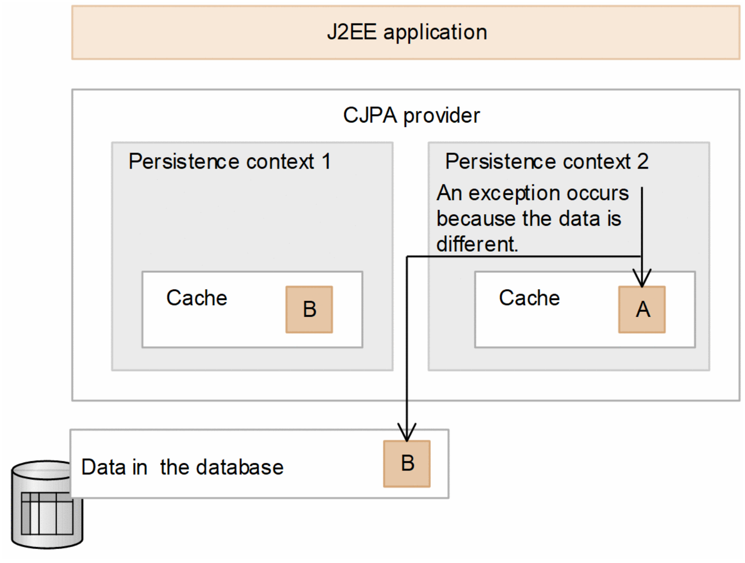
1. In persistence context 1, the data is changed from A to B.
   At this time, the contents of the cache and the database are identical, so an exception does not occur.

Figure 9–20: Changes in data in persistence context 1



2. After the processing of 1 ends, A data is changed in persistence context 2.
   The cache data is not changed; so the database data and the cache are inconsistent. Therefore, an exception is thrown due to the optimistic lock.

Figure 9–21: Changes in data in persistence context 2



In such an environment, if you use the cache, the non-updated cache is left behind and an exception might occur due to the optimistic lock.

(b) Action

When an optimistic lock exception occurs, the relevant objects in the cache are deleted. Therefore, execute the `find` method or the `refresh` method to obtain all the related data from the database once again. With this, you can synchronize the cache data and the database.

(3) Notes on the cache registration and update timing

- When you use JPQL to obtain the pessimistic lock, the entity object returned when the query is executed is not registered in the cache.

- When a native query is used to execute the query that sets the entity as the return value in `@SqlResultSetMapping`, the entity object of the return value is stored in the cache.
- During the `refresh` operation of an entity, if `OptimisticLockException` occurs in another thread after the entity object is read and the cache is deleted, the entity object is not registered in the cache even if the refresh processing is executed.

9.8 Auto-numbering of the primary key values

The primary key numbering functionality automatically generates the primary key value when the entity object is used to insert a record. Due to this functionality, even if the user does not specify the primary key value, a unique value is stored. Cosminexus JPA Provider provides the primary key numbering functionality.

How to generate the primary key value

There are four methods for generating the primary key values:

- `TABLE`

  In this method, you generate the primary key values by using a table to store the primary key values.

- `SEQUENCE`

  In this method, you use the database sequence objects to generate the primary key values. However, if HiRDB is used as the database, implement the same processing as that of `TABLE` with Cosminexus JPA Provider.

- `IDENTITY`

  In this method, you use the `identity` column of the database to generate the primary key values. However, with Cosminexus JPA Provider, the operations differ depending on the database type used.

  In HiRDB, you implement the same processing as that of `TABLE`.

  In Oracle, you implement the same processing as that of `SEQUENCE`.

- `AUTO`

  You choose the generation method suitable for the database used. With Cosminexus JPA Provider, choose `TABLE` for both HiRDB and Oracle.

Timing when the primary key values are numbered

In Cosminexus JPA Provider, the primary key values are numbered during the `flush` operation or when a transaction is committed.

Example of SEQUENCE as the primary key value generation method

The following is an example of using `SEQUENCE` as the generation method of the primary key values. In this example, a sequence object named `EMP_SEQ` is assumed to have been created beforehand.

```
@Entity
public class Employee {
...
    @SequenceGenerator(
        name="EMPLOYEE_GENERATOR",
        sequenceName="EMP_SEQ"

    )
    @Id
    @GeneratedValue(strategy=GenerationType.SEQUENCE,
 generator="EMPLOYEE_GENERATOR")
    @Column(name="EMPLOYEE_ID")
    public Integer getId() {
        return id;
    }
...
}
```

9.9 Database operations based on the query language

To use the JPA for operating the database data, implement the operation through the `javax.persistence.Query` interface. Using the `javax.persistence.Query` interface enables you to execute operations, such as search, update, and delete, for multiple records at the same time. To use the `javax.persistence.Query` interface, the user uses the query language to operate the database.

With Cosminexus JPA Provider, you can use JPQL and SQL as the query language. A description of JPQL and SQL is as follows:

- **JPQL**
  JPQL is a query language defined in the JPA specifications. This language does not depend on the database and operates for the entity class.
- **SQL**
  SQL is a database-dependent query language. SQL is also called a *native query*. This language operates for the database data.

For details on the database operations based on JPQL and SQL, see *9.16 Procedure for referencing and updating the database with the query language*. Furthermore, for details on the JPQL syntax, see *9.17 JPQL coding method*.

9.10 Optimistic lock

With Cosminexus JPA Provider, you use `EntityManager` and persistence context to manage the entities to be persisted. The changed entity information is applied to the database when the `flush` method or transaction commit processing is executed. When the same line of the database might be updated at the same time in multiple transactions, the data integrity must be ensured. Cosminexus JPA Provider provides the optimistic lock to ensure data integrity.

The optimistic lock is a lock functionality for making sure that the data is not updated by other applications from the beginning of the data update processing until the update processing is complete. The optimistic lock does not lock the database, and therefore, has the advantage that deadlocks do not occur.

This section describes the optimistic lock.

9.10.1 Optimistic lock processing

If you use the optimistic lock, Cosminexus JPA Provider checks whether the database data is being updated by other applications, instead of the user. If the database data is being updated by other applications, Cosminexus JPA Provider throws an exception and notifies the user that the data is being updated. Furthermore, Cosminexus JPA Provider marks the transaction for rollback.

(1) Procedure for checking whether the data is updated

Whether the data is updated is checked by the presence or absence of update of the version column prepared in the database table. If the data in the database is updated, the version number of the version column is updated. As a result, the user understands that the database was updated by another application. The following table describes the states and operations of the version column when the database is updated.

Table 9–16: States and operations of the version column when the database is updated

State of the version column in the table	Operation
When the value of the version column is not updated	Cosminexus JPA Provider applies the entity information to the database. At this time, the value of the version column in the database is updated.
When the value of the version column is updated	Indicates that the database data is being updated by another application. Therefore, Cosminexus JPA Provider throws `OptimisticLockException` and marks the transaction for rollback.

In this way, using the state of the version column, you can ensure that another transaction does not update the data during the period from the reading of the entity until the database is updated.

(2) Checking the versions of the persistence fields and relationships

To use the optimistic lock, you check the versions of both persistence fields and relationships of the entities. To check the versions, set the `Version` field (property) corresponding to the version column for the entity. Set the `Version` field by using `@Version` or the `<version>` tag of the O/R mapping file.

The version of the entity is checked at one of the following timings:

- When the entity state is changed and that change is written to the database
- When the entity state is changed to `managed` by the `merge` processing#

#

The version is not only checked during the execution of `merge`, but also during `flush` or when the transaction is committed.

When the version of the entity is determined to be old by the version check, `OptimisticLockException` occurs. The transaction is also marked for rollback.

(3) Checking the versions during the flush operation or transaction conclusion

You can check the version of the entity during the `flush` operation or transaction conclusion. To check the version, specify the entity in the `lock` method of `EntityManager`. By using the `lock` method of `EntityManager`, you can add the entity as a target for version check in the transaction and change the update policy of the `version` column.

Cosminexus JPA Provider supports `LockModeType.READ` and `LockModeType.WRITE` as the timing for updating the version column (`LockModeType` of the `lock` method). Regardless of the content specified in the update timing for the version column, Cosminexus JPA Provider ensures that the following two events do not occur during transaction conclusion:

- Dirty Read

  Transaction T1 changes a line. Then, before T1 executes commit and rollback, another transaction T2 reads the same line and obtains the changed value. Finally, T2 succeeds in commit.

  Whether T1 executes commit or rollback is no longer important, but whether T1 performs commit or rollback before or after commit by T2 becomes important.
- Un-Repeatable Read

  Transaction T1 reads a line. Then, before T1 executes commit, another transaction T2 changes or deletes that line. Finally, both the transactions succeed in commit.

If you specify `LockModeType.WRITE` as the `LockModeType`, the `version` column is forcefully updated even when there is no change in the entity state. The version column is updated when `flush` or transaction commit is invoked. Note that if the entity is deleted before the version column is updated, the update of the version column might be omitted.

9.10.2 Exception processing when optimistic lock fails

If you want to check the execution of an optimistic lock explicitly, invoke the `flush()` method. If an optimistic lock exception occurs during the invocation of the `flush()` method, you can catch the exception processing and perform the recovery processing. If the optimistic lock has no problems, the entity version is checked and the `Version` column is updated by the `flush()` method.

A coding example of the exception processing is as follows:

```
try{
  em.flush();
} catch (OptimisticLockException e){
  //  Exception processing
}
```

As shown in this example, by wrapping `OptimisticLockException` during the exception processing, you can show the optimistic lock exception as an application exception. However, note that the transaction in this case is marked for rollback and cannot be committed.

Note that with Cosminexus JPA Provider, the entity that causes the exception is not stored in `OptimisticLockException`. The `getEntity()` method of `OptimisticLockException` always returns a null value.

9.10.3 Notes on using the optimistic lock

This section describes the notes on using the optimistic lock.

(1) Notes on the Version field settings

The notes on the `Version` field settings are as follows:

- If the `Version` field is not set, the version of that entity is not checked. In this case, the user shall create an application that maintains the consistency between the entity and the database.
- If the transaction contains entities for which the `Version` field is set and entities for which the `Version` field is not set, the version is only checked for the entities where the `Version` field is set. Note that the non-inclusion of a version in the entity does not affect the transaction conclusion processing.
- The user can reference the `Version` field value, but must not update the value. However, the user can update the `Version` field value during bulk update processing.

(2) Notes on using the lock method

The notes on using the `lock` method are as follows:

- The `lock` method of `EntityManager` is not supported for entities that do not have the `Version` field (property). If an entity that does not have a `Version` field (property) is specified and the `lock(entity, LockModeType.READ)` or `lock(entity, LockModeType.WRITE)` is invoked, `PersistenceException` occurs.
- When the state of an entity that has a `Version` field (property) is updated, regardless of whether the `lock` method is invoked, the dirty read and un-repeatable read events do not occur.
- When you specify `LockModeType.READ` with Cosminexus JPA Provider, the `UPDATE` statement is issued to check whether the database value corresponding to the entity is changed. Therefore, the `UPDATE` statement sets a lock for the database. The `UPDATE` statement is issued during the execution of `flush` processing and when the transaction is committed. The `UPDATE` statement is also issued when there is no change in the state of the entity object. However, the statement is not issued if the entity is deleted.

(3) Exclusive control of clients in HiRDB

The optimistic lock of Cosminexus JPA Provider is a locking method that assumes that the database Isolation level is accessed with `Read Committed`. If the database is HiRDB, the Isolation level is `Repeatable Read` by default; therefore, you must change the level to `Read Committed`.

Set the Isolation level for each client in the `PDISLLVL` parameter of the data guarantee level of the client environment variable. The default value is `Repeatable Read (2)`. Therefore, change the value to `Read Committed (1).` An example of a change in setting is as follows:

Example of change: `PDISLLVL=1`

Specify the client environment variable in the value of the `environmentVariables` property with the `<config-property>` tag of the HITACHI Connector Property file or add the client environment variable in the configuration file for the client environment variable group of HiRDB.

If the data guarantee level of the client environment variable is operated with the default `Repeatable Read`, a lock is set in the shared mode. Therefore, note that if you combine the issue of reference series SQL such as the `find` method and the issue of update series SQL such as the `flush` method, a deadlock occurs easily.

9.11 Pessimistic lock in JPQL

The pessimistic lock is the method of exclusively locking the target records when multiple transactions update the same record on the database. If a transaction sets a pessimistic lock for a particular record, another transaction cannot reference or update that record (however, in Oracle, the record can be referenced). A pessimistic lock is only available when JPQL is used.

If you use a pessimistic lock, until the transaction that obtained the lock terminates, the lock awaits release. Therefore, though the concurrent executions are not more than the optimistic lock, you can prevent the transaction commit errors that occur in the optimistic lock.

How to specify a pessimistic lock

Implement the pessimistic lock by using the query hint supported by Cosminexus JPA Provider. Execute the pessimistic lock by specifying the `setHint()` method of the `Query` method or by specifying `@QueryHint` in the `@NamedQuery` attribute.

Example of implementing the pessimistic lock functionality

An example of implementing the pessimistic lock functionality is as follows:

Example of implementation 1

An example of specifying the pessimistic lock in the `setHint()` method of the `Query` method is as follows:

```
Query query = manager.createQuery("SELECT emp FROM  Employee  AS emp");
query.setHint("cosminexus.jpa.pessimistic-lock","Lock");
```

Example of implementation 2

An example of specifying the pessimistic lock in `@QueryHint` of the `@NamedQuery` attribute is as follows:

```
@NamedQuery(
name="employee_list",
query="SELECT emp FROM  Employee AS emp",
hints={ @QueryHint(name="cosminexus.jpa.pessimistic-lock", value="Lock"
) }
)
@Entity
public class Employee{
...
}
```

Important note

The pessimistic lock is only available for JPQL. The pessimistic lock is not enabled even by specifying the `createNativeQuery` method or by specifying `@QueryHint` in the `hints` attribute of `@NamedNativeQuery`. The pessimistic lock is also not enabled by specifying the `hint` attribute of the `<named-native-query>` tag in the O/R mapping file. Note that the locking specifications for the pessimistic lock in Cosminexus JPA Provider conform to the specifications of the database used.

9.12 Creating an entity class

To create an application that uses the JPA, you define the entity class in the application. An entity class is used for handling the database table records as Java objects. When a user performs the `new` operation in the application, an entity class instance is generated.

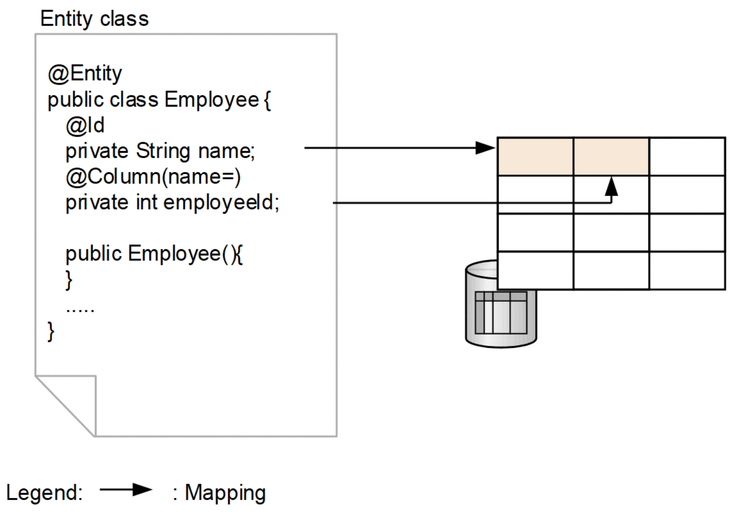
This section describes the creation of a JPA application.

9.12.1 Defining the mapping between an entity class and database

An entity class is mapped to a line in the database table. The fields stored in the entity class are mapped to the values in the table columns. If the user updates a field value for an entity class instance, Cosminexus JPA Provider also updates the corresponding column of the database table. Therefore, the user can change the database state without issuing an SQL statement for the database.

The following figure shows the mapping between an entity class and a database table.

Figure 9–22: Mapping between the entity class and database table



```
@Entity
public class Employee{
  @Id
  private String name;
  @Column(name=)
  private int employeeId;

  public Employee(){
  }
  .....
}
```

You define the correspondence relationship between the entity fields and database columns in an annotation or the O/R mapping file. You can define the correspondence in either an annotation or the O/R mapping file. However, if the definition is specified in both, the settings in the O/R mapping file are given priority over the annotation. If different values are specified for the same settings in the annotation and the O/R mapping file, the value in the annotation is overwritten with the value in the O/R mapping file.

9.12.2 Requirements for creating entity classes

When you create an application that uses the JPA, you must follow the requirements for creating the entity classes and the requirements for database mapping determined in the JPA specifications. The requirements for creating entity classes are as follows:

- The entity class must be specified in `@Entity` or in the `<entity>` tag of the O/R mapping file.
- The entity class has a constructor without an argument.
- Do not set `enum` and interfaces as entity classes.

- The entity or the mapped super-class that form the root of the class hierarchy of the entity class must have a primary key. Make sure that you define one primary key in the entity hierarchy.
- When the entity class instance is passed as a method argument with pass by value, you must implement the `Serializable` interface.
- The state of the database column is expressed by the instance variable of the entity and the instance variable corresponds to the JavaBean property. Do not change the value of the instance variable by direct access from the client. Change the value through the accessor method (`getter` / `setter` method) or the business method.
- Set the persistent instance variable of the entity to an access level that can be referenced from `private`, `protected`, or `package`.
- Declare the constructor without an argument as `public` or `protected`.
- Do not set the entity class to `final`. Also, do not set the persistence instance variable of the entity class and all the methods to `final`.

With Cosminexus JPA Provider, if these conditions are not satisfied, an exception might occur. Note that even if an exception does not occur, the operations might not function properly if an entity class that does not satisfy these conditions is created.

Furthermore, for Cosminexus JPA Provider, do not associate one database column with multiple fields in the entity class. If this condition is not satisfied, an exception might occur during the execution of the application. Even if the exception does not occur, the operations might not function properly.

9.12.3 Specifying the access methods for the entity class fields

Cosminexus JPA Provider accesses the entity class fields when the entity state is written to the database and the database state is read as an entity. The access method in this case is called *access type*. An access type includes properties and fields. You specify the access type in an annotation or in the O/R mapping file. The following table describes the access types and the specification methods.

Table 9–17: Access types and specification methods

Access type	Description	Specification method	
		Annotation	O/R mapping file
Property	Method of obtaining the instance variable through the `getter` method.	Specify the annotation in the `getter` method of the field.	Specify `PROPERTY` in the `<access>` tag.
Field	Method of directly referencing the instance variable.	Specify an annotation in the field.	Specify `FIELD` in the `<access>` tag.

If the access type is a property, the property stored by the entity is called the *persistence property*. Also, when the access type is field, the entity field is called the *persistence field*.

Notes on the access types

Remember the following points when you specify the access types:

- If the access type is a field, Cosminexus JPA Provider directly accesses the persistence field. The instance variable for which `@Transient` is not set is subject to persistence.
- If the access type is property, Cosminexus JPA Provider uses the accessor method to obtain the persistence property value. The property for which `@Transient` is not set in the accessor method is subject to persistence.

- If the access type is property, you cannot set the mapping annotation in the `setter` method. In the case of Cosminexus JPA Provider, the mapping annotation set in the `setter` method is ignored.
- You cannot set the mapping annotation in the field and property in which `@Transient` is specified or `<transient>` tag is specified in the O/R mapping file. If the mapping annotation is set, an exception occurs when the application starts.
- Set the accessor method of the property to `public` or `protected`. This is prohibited in the JPA specifications, but is not checked with Cosminexus JPA. Also, even in the case of `private`, an exception does not occur.
- If the mapping annotation is applied to the accessor methods of both, the persistence field and the persistence property, all the annotations set in the accessor method of the persistence property are ignored.

9.12.4 Creating the accessor method

This section describes the signature rules of the accessor method and the addition of the business logic to the accessor method.

(1) Method signature rules of the accessor method

When Cosminexus JPA Provider accesses a persistence property, the accessor method of the property must follow the same method signature rules as the following JavaBeans:

- `T getProperty()`
- `void setProperty(T t)`

For a property that returns the return value `boolean`, you can also change the name of the `getter` method to `isProperty`. Note that when you use Cosminexus JPA Provider, an exception occurs when the application starts if only the `getter` method or the `setter` method exists.

Also, when you handle collection values in the persistence fields and persistence properties, define the following collection values in the interface:

- `java.util.Collection`
- `java.util.Set`
- `java.util.List`
- `java.util.Map`

If the persistence property becomes a collection value, set the accessor method signature to one of these interfaces or you can also use the generic type of these collections (example: `Set<T>`).

(2) Adding the business logic to the accessor method

In addition to the `setter`/`getter` processing of the property, the accessor method can also include the business logic, such as verifying the values. If the access type is property, the business logic operates when Cosminexus JPA Provider invokes the accessor method.

In this case, however, note the following points:

- The order in which the accessor methods that load and store the persistent states are invoked, is not defined when Cosminexus JPA Provider is executed. Therefore, the execution order of the logic included in `getter` is not yet decided.
- For portability when the access type is property and lazy fetch is specified in the persistence property, we recommend that you do not access the entity contents until the entity contents are fetched by Cosminexus JPA Provider.
- When the access type is property and when a logic that changes the value is added as the business logic, Cosminexus JPA Provider does not guarantee data consistency.

If the `Runtime` exception occurs in the property accessor method, the current transaction is marked for rollback. If Cosminexus JPA Provider reads the persistent contents of the entity and throws an exception in the accessor method used for storing the contents, the transaction is rolled back. Also, the `PersistenceException` exception that wraps the application exception occurs.

9.12.5 Types of persistence fields and persistence properties of the entities

Set the persistence fields/ persistence properties of the entities to the following types:

- Java primitive type
- `java.lang.String`
- Other serialize types
- Primitive type wrapper class
- `java.math.BigInteger`
- `java.math.BigDecimal`
- `java.util.Date`
- `java.util.Calendar`
- `java.sql.Date`
- `java.sql.Time`
- `java.sql.Timestamp`
- User-defined serialize type
- `byte[]`
- `Byte[]`
- `char[]`
- `Character[]`
- `enum`
- Entity type and the collection duplicable in the entity type
- Embeddable class

When you use Cosminexus JPA Provider, the operations might not function properly if you specify types other than those mentioned above. An exception might also occur when the application is executed.

However, when you use Cosminexus JPA Provider, you must associate the instance variable type of the entity with the database column type. The JDBC driver associates the Java types and the database types. With Cosminexus JPA Provider, to connect to Oracle, you use Oracle JDBC Thin Driver provided by Oracle as the JDBC driver. To connect to HiRDB, you use HiRDB Type4 JDBC Driver as the JDBC driver. The user must convert the types supported by the JDBC driver as well as determine the instance variable type of the entity.

9.12.6 Specifying the primary key in the entities

With an entity, make sure that you specify the primary key in the class hierarchy. When you specify the primary key, follow the below rules:

- For a simple (not complex type) primary key, specify `@Id` in the persistence field or persistence property or specify the key in the O/R mapping file. Accordingly, associate the primary key with the entity fields.
- For a complex type primary key, specify the primary key class in a single field as `@EmbeddedId` or specify the complex primary key as a field set using `@IdClass` and `@Id`.
- For a complex type primary key, create a class that contains the primary key, called the *primary key class*.

If these conditions are not satisfied, an exception occurs when you start the application.

Also, do not change the primary key value of the entity in the application. If the primary key value is changed in the application, an exception occurs during execution.

(1) Primary key type

Set one of the following types for the simple or complex type primary keys:

- Java primitive type
- Type that wraps primitive
- `java.lang.String`
- `java.util.Date`
- `java.sql.Date`

Note that if the approximation type (for example, floating-point number type) is specified as the primary key, rounding off errors and problems such as lack of reliability in the results of the `equals` method occur in Cosminexus JPA Provider. Therefore, the operations when the approximation type is used in the primary key might not function properly with Cosminexus JPA Provider. If `java.util.Date` is used in a field/ property as the primary key, the `temporal type` attribute must be specified as the `DATE` type.

(2) Complex type primary key

The complex type primary key can be handled with an entity. The methods of specifying a complex type primary key in the entity include the method of using the embedded class and the method of using `@IdClass`. The following points describe each method:

(a) Method of using the embedded class

To use an embedded class, create a class that assigns `@Embeddable` and define the complex type primary key as a field in that class. With the entity class, define the type field of the class that assigns `@Embeddable` and annotate `@EmbeddedId`. The examples of entity class and embedded class are as follows:

- Example of an entity class

```
@Entity
public class Employee {
    private EmployeePK employeePK;

    public Employee(){
    }

    @EmbeddedId
    public EmployeePK getEmployeePK(){
      retrun this.employeePK;
    }

    public void setEmployeePK(EmployeePK employeePK){
      this.employeePK = employeePK;
    }
    ...
}
```

- Example of an embedded class

```
@Embeddable
public class EmployeePK {
    private String name;
    private int employeeId;

    public EmployeePK(){
    }

    public boolean equals(Object obj){
      ...
    }

    public int hashCode(){
      ...
    }
    ...
}
```

For details on the embedded class, see *(3) Embedded class*. Note that you can also use the O/R mapping file instead of the annotation.

(b) Method of using @IdClass

To use `@IdClass`, define multiple instance variables corresponding to the primary key in the entity class and assign `@Id`. Also, use `@IdClass` to specify the primary key class. With the primary key class, define a field or property with the same name and type as the primary key defined in the entity. The examples of entity class and primary key class are as follows:

- Example of an entity class

```
@Entity
@IdClass(EmployeePK.class)
public class Employee {
    private String name;
    private int employeeId;

    public Employee(){
    }

    @Id
    public String getName(){
      retrun this.name;
    }

    public void setName(String name){
      this.name = name;
    }

    @Id
    public int getEmployeeId(){
      retrun this.employeeId;
    }

    public void setName(int employeeId){
      this.employeeId = employeeId;
    }
    ...
}
```

- Example of a primary key class

```
public class EmployeePK implements Serializable {
    private String name;
    private int employeeId;

    public EmployeePK(){
    }

    public boolean equals(Object obj){
      ...
    }

    public int hashCode(){
      ...
    }
    ...
}
```

As in the case of the embedded class, you can use the O/R mapping file instead of the annotation. Note that the access type of the primary key class is determined by the access type of the entity class corresponding to the primary key.

To handle a complex type primary key, use either an embedded class or `@IdClass`. However, follow the below rules:

- The primary key class must be `public` and must have a constructor without argument.
- When you use a persistence property, set the property of the primary key class to `public` or `protected`.
- The primary key class must be serializable.
- Define the `equals` and `hashCode` method with the primary key class. When the primary keys on the mapped database are equal, `true` must be returned for `equals` and the `hashCode` value must be equal.
- The complex primary key must be mapped as an embedded class or must map multiple fields/properties of the entity class.

- When a primary key class is mapped to the complex fields/properties of the entity class, match the field/property name of the primary key of the primary key class with the name of the entity class. Also, unify the types.

With Cosminexus JPA Provider, the operations might not function properly if these conditions are not satisfied. If the conditions are not satisfied, an exception might also occur when you start the application.

(3) Embedded class

If you prepare a class that brings together some fields for persistence, you can store the fields as entity fields. Such a class is called an *embedded class*.

The embedded class is embedded in an entity and is mapped to the same database table as the entity. Therefore, unlike the entity, the embedded class does not have a primary key.

When using the embedded class, the user specifies `@Embeddable` in the embedded class. Also, specify `@Embedded` in the embedding destination field or property in the entity class that is embedded. Note that you can also define the embedded class similarly in the O/R mapping file instead of the annotation.

You can also use the embedded class for defining the complex type primary key. In this case, specify `@EmbeddedId` instead of `@Embedded` in the embedded entity class.

Make sure that you conform to the following creation requirements for the embedded class:

1. Make sure that the embedded class is defined in `@Embeddable` or in the `<embeddable>` tag of the O/R mapping file.
2. Do not set `enum` and the interfaces as an embedded class.
3. When the entity class containing the embedded class is passed by value as a `detached` object, implement the `Serializable` interface.
4. Do not set the embedded class, the persistence instance variables of the embedded class, and all the methods to `final`.
5. The embedded class must have a constructor without an argument.
6. Declare the constructor without an argument as `public` or `protected`.
7. The instance variable of the embedded class must be referable from `private`, `protected`, or `package`.
8. Specify settings so that the persistence instance variable of the embedded class is not accessed directly from the client. Do not access the persistence instance variable of the embedded class with the `accessor` method (`getter/setter` method) of the entity and the other business methods.

With Cosminexus JPA Provider, if the conditions 1 and 2 are not satisfied, an exception occurs and the application fails to start. Also, in the case of conditions 3 to 8, an exception might occur, but even if an exception does not occur, the operations might not function properly.

You determine the access type of the embedded class using the access type of the entity class on the embedded side.

When using the embedded class, set one embedded hierarchy. Also, the embedded class object cannot be shared by multiple entities. If these conditions are not satisfied, the operations might not function properly with Cosminexus JPA Provider.

9.12.7 Default mapping rules for the persistence fields and persistence properties

If the O/R mapping information is not specified for the persistence fields or persistence properties other than relationship, the following default mapping rules are applied:

- For a class in which `@Embeddable` is annotated, the field/ property is mapped to the database according to the specification in the entity in `@Embedded`.
- If the persistence field/ persistence property type is one of the following, the mapping method is the same as when `@Basic` is defined:
  - Java primitive type
  - Primitive type wrapper
  - `java.lang.String`
  - `java.math.BigInteger`
  - `java.math.BigDecimal`
  - `java.util.Date`
  - `java.util.Calendar`
  - `java.sql.Date`
  - `java.sql.Time`
  - `java.sql.Timestamp`
  - `byte[]`
  - `Byte[]`
  - `char[]`
  - `Character[]`
  - `enum`
  - Any type that implements `Serializable`

Note that the operations might not function properly if types other than those mentioned above are specified.

9.13 Procedure for inheriting an entity class

The inheritance of an entity class has the following features:

- An entity class can use either an abstract class or a concrete class. Apart from the abstract class and concrete class, you can also define `@Entity`. Furthermore, both abstract class and concrete class can be mapped as entities and queries can be issued for both the classes.
- An entity class can be inherited from another entity class.
- An entity class can inherit a non-entity class. A non-entity class can also inherit an entity class.

This section describes the inheritance class types and the inheritance mapping strategy of the entity classes.

9.13.1 Inheritance class types

The inheritance class types of an entity class include the abstract entity class, mapped super-class, and non-entity class. The following is a description of each type:

(1) Abstract entity class

You can define an abstract class as an entity class. An abstract entity class differs from the concrete entity class in that an abstract entity class cannot create an instance directly. An abstract entity class can be mapped as an entity and can also be specified as a query target for operating/obtaining the subclass entity.

With a subclass, the accessor method of the property can be overridden. However, if the O/R mapping information on the persistence fields and persistence properties is overridden in a subclass, the operations might not function properly. To override the O/R mapping information in a subclass, use `@AssociationOverride` or `@AttributeOverride`.

You specify the abstract entity class in `@Entity` or the O/R mapping file.

(2) Mapped superclass

A mapped superclass is a class that can become the superclass of an entity class. You can define the persistence fields and mapping information and also set a configuration to inherit the persistence fields and mapping information.

A mapped superclass cannot specify a specific table, so `@Table` cannot be specified. Therefore, the mapped superclass cannot be set as an entity. Unlike the entity class, a mapped superclass cannot issue a query, so the mapped superclass cannot be passed to the arguments of `EntityManger` and query operations. Also, a mapped superclass cannot be set as a relationship target.

A mapped superclass can be defined as an abstract class as well as a concrete class. To define a class as a mapped superclass, define `@MappedSupperclass` or `<mapped-superclass>` tag in the O/R mapping file. The mapping information is provided for the inherited entity class.

A class designed as a mapped superclass can be mapped using the same method as the entity class, except when the mapping can only be provided for the subclasses.

If an entity is applied as a subclass, the information specified in the mapped superclass is inherited in the subclass. By using `@AssociationOverride` or the corresponding XML element in the O/R mapping file, the mapping information can be overridden in the subclass.

(3) Non-entity class of the entity inheritance hierarchy

An entity class can have a superclass of a non-entity class. A superclass can be defined as a concrete class and abstract class.

A superclass of a non-entity class has the following features:

- Only the behavior is inherited.
- The state is not persistent.
- All the inherited states are not persistent even in the inherited entity classes.
- A non-persistent state is not controlled by `EntityManager`.
- The annotation of a superclass is ignored.

Do not set a non-entity class in the method arguments of `EntityManager` and the `Query` interface and in the mapping information. In Cosminexus JPA Provider, an exception occurs during the execution of the application.

9.13.2 Inheritance mapping strategy

When an entity is inherited, you can specify the method of mapping the class hierarchy to a table as the inheritance mapping strategy. You specify the inheritance mapping strategy by using `@Inheritance` or the `<inheritance>` tag of the `<entity>` tag in the O/R mapping file.

There are three types of inheritance mapping strategies:

- **SINGLE TABLE strategy**
  The SINGLE TABLE strategy is a strategy method of mapping all the classes existing in the inheritance hierarchy of the entity class to one table.
- **JOINED strategy**
  The JOINED strategy is a strategy method of mapping the top-level of the class hierarchy to a single table.
- **TABLE PER CLASS strategy**
  The TABLE PER CLASS strategy is a strategy method of mapping each class existing in the class hierarchy of the entity class to separate tables.

However, with Cosminexus JPA Provider, the TABLE PER CLASS strategy is not available. If the TABLE PER CLASS strategy is used when Cosminexus JPA Provider is used, an exception occurs when the application starts.

Important note

When you use Cosminexus JPA Provider, note the following points:

- You cannot combine and specify multiple inheritance strategies in a class hierarchy. Also, no check is performed to verify whether multiple inheritance strategies are combined. If multiple inheritance strategies are specified, the operations might not function properly.
- If the column specified in `@DiscriminatorColumn` is defined in an entity field, the value set in the entity field is not applied to the database even if the `persist` operation and then commit is executed. The value specified in `@DiscriminatorValue` or the default value is applied. Also, note that after commit, the value set in the field before commit is stored as is.

These strategies are specified in the value of the enumeration type `javax.persistence.DiscriminatorType`. The following points describe each strategy:

(1) SINGLE TABLE strategy

The SINGLE TABLE strategy is a strategy method of mapping all the classes present in the inheritance hierarchy of the entity class to one table. Therefore, the table must have an identification column as the column for identifying the class.

Specify the identification column in `@DiscriminatorColumn` or in the O/R mapping file. The default identification column name is `DTYPE`. If an identification column does not exist in the database, an exception occurs during the execution of the application.

If you want to specify the value stored in the identification column, use `@DiscriminatorValue` or the `<discriminator-value>` tag beneath the `<entity>` tag of the O/R mapping file.

Important note

When you use the SINGLE TABLE strategy, the user must be able to specify a null value in the table column corresponding to the subclass field.

(2) JOINED strategy

The JOINED strategy is a strategy method of mapping the top-level of the class hierarchy to a single table. Each subclass is indicated by the subclass-specific fields that are not inherited from the superclass and by different tables with the primary key string that functions as the external key of the primary key for the superclass table.

In the case of the JOINED strategy, like in the case of the SINGLE TABLE strategy, the table to which the superclass is mapped must have an identification column.

Important note

When using the JOINED strategy, binding must be executed multiple times during the generation of the subclass instances. Therefore, if the hierarchy structure becomes deep, the performance might deteriorate. Also, `JOIN` is necessary for issuing queries across the class hierarchy.

9.14 Procedure for using EntityManager and EntityManagerFactory

This section describes the procedure for using `EntityManager` and `EntityManagerFactory` that are used from the application.

9.14.1 Entity lifecycle management with EntityManager

`EntityManager` is an object with an interface for executing the following operations for the database:

- Registering and deleting entities.
- Searching entities using the primary keys.
- Issuing a query across the entities.

If you register an entity for `EntityManager`, the entity state is perpetuated in the database at an appropriate time such as when the transaction is committed.

Also, `EntityManager` has a relation with the persistence context that expresses the entity set. When you register the entity in `EntityManager`, the entity belongs to a specific persistence context. Also, `EntityManager` manages the entity lifecycle.

The entity set managed by `EntityManager` is defined with a unit called the *persistence unit*. You define the persistence unit in the application configuration file `persistence.xml`.

The notes on the persistence context and persistence unit are as follows:

- The entity must be unique in the persistence context. Therefore, set one entity expressing the same line of the database in the same persistence context. Note that if the persistence contexts are different, you can have multiple entities expressing the same line in the database. For the locking method in the database in this case, see *9.10 Optimistic lock* or *9.11 Pessimistic lock in JPQL*.
- Each persistence unit is mapped to a single database. For details on the definition, see *8.8 Definitions in persistence.xml*.

9.14.2 How to set up EntityManager and EntityManagerFactory

You set up the `EntityManager` and `EntityManagerFactory` you want to use in an application in an annotation or the DD.

- **EntityManagerFactory settings**
  - In an annotation: Specify the settings in `@PersistenceUnit`.
  - In the DD: Specify the settings in the `<persistence-unit-ref>` tag.
- **EntityManager settings**
  - In an annotation: Specify the settings in `@PersistenceContext`.
  - In the DD: Specify the settings in the `<persistence-context-ref>` tag.

For details on annotations, see *8.12 javax.persistence package*. For details on the tags to be set for DD, see the *uCosminexus Application Server Definition Reference Guide*.

9.14.3 Notes on the API functions of EntityManager

This section describes the notes on the API functions provided by `EntityManager`. For details on the EntityManager APIs, see *8.12 javax.persistence package*.

- When `EntityManager` of transaction scope persistence context is used, the `persist`, `merge`, `remove`, and `refresh` methods must be executed in the transaction context. If the transaction context does not exist, `javax.persistence.TransactionRequiredException` is thrown.
- The `find` method and `getReference` method do not require execution in the transaction context. Therefore, if `EntityManager` of transaction scope persistence context is used, the resulting entity has a `detached` state. Also, if `EntityManager` of extended persistence context is used, the resulting entity has a `managed` state.
- If the argument of the `createQuery` method is not a valid JPQL string, the `IllegalArgument` exception is sent and the execution of the query fails.
- If the executed native query does not match the specifications for the database to be connected to or if the defined result set is not compatible with the query results, the execution of the query fails and the `PersistenceException` exception is thrown when the query is executed.
- If a runtime exception is sent from a method of the `EntityManager` interface, the current transaction is marked for rollback.
- The `Query` object obtained from `EntityManager` and the `EntityTransaction` object are enabled while `EntityManager` is open.

9.15 Procedure for specifying the callback method

To receive an entity lifecycle, you can specify a method in the lifecycle callback method. The method defined as a callback method is invoked corresponding to the persistence-related lifecycle event.

This section describes the location for specifying the callback method, the implementation methods, and the order of invocation.

9.15.1 Location for specifying the callback method

You can specify the callback method at the following locations:

- In the entity class or mapped superclass
- Entity listener class associated with the entity class or mapped superclass

The entity listener class is a dedicated class for implementing the callback method. If you use the entity listener class, you can separate the parts for implementing the callback method.

Specify the callback method in an annotation or the O/R mapping file. However, the default callback method is specified in the O/R mapping file and cannot be specified in an annotation. Note that the default callback method indicates an entity listener applied to all the entities in a persistence unit.

This section describes how to specify the callback listener.

(1) Specifying the callback method in an annotation

If you use an annotation for specifying the callback method, set the annotations listed and described in the following table in the method. You can invoke the method in compliance with a lifecycle event.

Table 9–18: Specifying the callback method using annotations

Annotation	Executed contents
`@PostLoad`	The callback method is executed after the entity is loaded in the persistence context or after the `refresh` operation is applied. The method is executed after the entity is read from the cache or the `SELECT` statement is issued for the database.
`@PrePersist` `@PreRemove`	The callback method is invoked before `EntityManager` executes the `persist` or `remove` operation of the entity. When the merge operation is applied and a new `managed` instance is created, the `PrePersist` callback method is invoked for the `managed` instance after the entity state is copied. The `PrePersist` or `PreRemove` callback methods are also invoked for all the instances where the operations are cascaded. The `PrePersist` or `PreRemove` method is always invoked synchronously as a part of the `persist`, `merge`, or `remove` operations.
`@PostPersist` `@PostRemove`	The `PostPersist` and `PostRemove` callback methods are invoked after the entity is perpetuated or deleted by the `persist` or `remove` operations. These callback methods are also invoked for all the entities where the operations are cascaded. The `PostPersist` or `PostRemove` methods are respectively invoked after the database `insert` or `delete` operations are performed. These database operations are executed immediately after the `persist`, `merge`, or `remove` operations or after the `flush` method is invoked. However, the database operations may also be executed at the end of a transaction. The generated primary key can be used with the `PostPersist` method.
`@PreUpdate` `@PostUpdate`	The `PreUpdate` and `PostUpdate` callback methods are respectively invoked before and after the database `update` operation of the entity data.

Annotation	Executed contents
	These database operations are executed when the entity state is updated or when the state is flushed in the database. However, the database operations might also be executed at the end of a transaction. When an entity is perpetuated and then updated and when an entity is updated and then deleted in one transaction, the `PreUpdate` and `PostUpdate` callback might not occur.

To use the entity listener class, you must specify the entity listener class by specifying `@EntityListener` for the entity. The following is an example of specifying an entity listener class:

```
@Entity
@EntityListeners(CallbackListener.class)
public class Employee implements Serializable{
...
}
```

(2) Specifying the callback listener in the O/R mapping file

Specify the following settings to use the O/R mapping file for specifying the callback method:

- To specify the entity listener class and the callback method of that class, use the `<entity-listener>` tag of the O/R mapping file. Specify the lifecycle listener method by using the `<pre-persist>` tag, `<post-persist>` tag, `<pre-remove>` tag, `<post-remove>` tag, `<pre-update>` tag, `<post-update>` tag, and `<post-load>` tag beneath the `<entity-listener>` tag.
- When you specify the callback method of the entity listener class, you can specify maximum one method for each callback event by using the tags beneath the `<entity-listener>` tag.
- If you specify the `<entity-listener>` tag of the O/R mapping file for the lower tags of the `<entity-listeners>` tag in the `<persistence-unit-defaults>` tag, you can specify the default callback method.
- If you specify the `<entity-listener>` tag in the lower tags of the `<entity-listeners>` tag existing in the `<entity>` tag or `<mapped-subclass>` tag, the callback listener is specified for the entity or mapped superclass and the subclasses.
- The callback listener is invoked in the order of listeners specified in the `<entity-listeners>` tag. For details on the order for invoking the listeners, see *9.15.3 Order of invoking the callback methods*.

9.15.2 Implementing the callback methods

The user implements the callback method as and when required. The callback method signature differs in the callback method implemented in the entity class and mapped superclass and in the callback method of the entity listener class.

The callback method defined in the entity class and mapped superclass has the following signature:

```
void <METHOD>
```

The callback method defined in the entity listener class has the following signature:

```
void <METHOD>(Object)
```

In the argument `Object`, you specify the entity instance in which the callback method is executed.

(1) Notes on using the callback methods

Note the following regarding the callback methods. If the following conditions are not satisfied, an exception occurs at application startup and the application fails to start:

- A constructor without argument must be specified in `public`.
- The `public`, `private`, `protected`, and package level access is allowed with the callback method. However, `static` and `final` are not available.
- One class cannot have multiple lifecycle callback methods for the same lifecycle event. However, the same method might be used in multiple callback events.

(2) Rules applied to the callback methods

The following rules are applied to the callback methods:

- With the callback method, the issuing of unchecked or runtime exceptions are allowed. The runtime exception thrown by the callback method executed in the transaction rolls back the transaction. If multiple callback methods are specified, after the runtime exception is thrown, the remaining callback methods are not executed.
- With the callback method, you can execute JNDI, JDBC, JMS, and Enterprise Bean.
- Do not perform the following operations with a callback method:
  - Invoking `EntityManager`.
  - Executing a query operation.
  - Accessing other entity instances.
  - Updating a relationship.

  If a callback method is used for such methods, the operations might not function properly.
- If the callback method is invoked in the Java EE environment, the entity callback listener shares the naming context of the components to be invoked. Furthermore, the callback method of the entity is invoked in the transaction and in the security context of the invocation source components used when the callback method is invoked.

9.15.3 Order of invoking the callback methods

If multiple callback methods are defined for an entity, the invocation order follows the below rules:

1. The default listeners are invoked in the order defined in the O/R mapping file.
   Unless `@ExcludeDefaultListeners` or the `<exclude-default-listeners>` tag of the O/R mapping file is explicitly specified, the default listeners are applied to all the entities in the persistence unit.
2. The callback methods are invoked in the order specified in `@EntityListeners`.
   Note that if you use the O/R mapping file, you can execute the following operations:
   - Specifying the order of invoking the callback methods for the entities.
   - Overriding the order specified in the annotations.
3. The callback method specified in the entity (or mapped superclass) is invoked.

(1) Invocation order in the inheritance hierarchy

If the entity listener is defined multiple times in the inheritance hierarchy of the entity class and mapped superclass, the invocation order is as follows:

1. The default callback listener, if present, is invoked first.
2. The callback methods of the entity listener class are invoked sequentially from the listener specified in the superclass. At this time, if items such as `@EntityListener` are specified, that order is followed.
3. After the callback methods of all the entity listeners are invoked, the callback methods defined in the entity (or mapped superclass) are invoked sequentially from the listener specified in the superclass.

If the callback method is overridden in the subclass, the overridden method is not invoked. If the overridden callback method specifies different lifecycle events or if the overridden callback method is not a lifecycle callback method, the overridden method is invoked. Also, the callback method settings of the method are overridden.

(2) Excluding the callback methods

- If you specify `@ExcludeDefaultListeners` or the `<exclude-default-listeners>` tag of the O/R mapping file, the default entity listener is not invoked in the entity class (or mapped superclass) and the subclasses.
- If `@ExcludeSuperclassListeners` or the `<exclude-superclass-listeners>` tag of the O/R mapping file is applied to the entity class and mapped superclass, the listener callback method is not invoked in that class and the subclasses. `@ExcludeSuperclassListeners` or the `<exclude-superclass-listeners>` tag of the O/R mapping file does not exclude the invocation of the default entity listener.
- If you use the O/R mapping file to explicitly specify the default or superclass listener excluded for the entity and mapped superclass, the default or superclass listener is applied to the entity and the subclasses.

9.16 Procedure for referencing and updating the database with the query language

A query expresses a processing request (inquiry) for a database as a string. A query is used to issue commands such as commands to search, update, and delete the data in a database to a system. To execute a query, use the `javax.persistence.Query` interface. The types of query include JPQL and native query. This section describes the procedure for referencing and updating the database with a query.

9.16.1 Procedure for referencing and updating the database with JPQL

JPQL is a query language used for searching and updating the database and for using the database functionality such as the set function. While the SQL is a query language using a table as the target, JPQL is a query language defined in the JPA specifications using an entity class as the target.

You can define a query in an annotation or the O/R mapping file. If an entity is defined in the same persistence unit as the query, you can use the abstract schema type expressing the entity set in the query. Also, if you use the path expression, you can use a query across the relationship defined in the persistence unit. For details on the path expression, see *9.17.4(2) Path expression*.

If you code and execute JPQL in an application, the SQL statements are issued for the database to be connected to in the following order:

1. If JPQL is executed, Cosminexus JPA Provider interprets the JPQL contents.
2. Based on the annotation and O/R mapping file information coded in the target entity class, JPQL is set up in the SQL statements specific to the database product to be connected to and then issued.

This section describes how to use JPQL.

(1) How to obtain the Query object

To use JPQL for obtaining the `Query` object, use the following methods of the `EntityManager` interface provided by Cosminexus JPA Provider. This section describes the methods of the `EntityManager` interface.

(a) Query createQuery(String JPQL statement)

An example of coding `createQuery` is as follows. In the argument, specify the JPQL statement you want to execute.

```
Query q = em.createQuery(
                   "SELECT c  " +
                   "FROM Customer c " +
                   " WHERE c.name LIKE "Smith");
```

(b) Query createNamedQuery(String query name)

A query that can be given a name and defined in advance is called a *named query*. You define a named query by assigning `@NamedQuery` in any entity class. Specify the query name in the `name` attribute of `@NamedQuery` and then specify the JPQL statement in the `query` attribute.

In Cosminexus JPA Provider, you cannot specify multiple named queries with the same name. If multiple named queries with the same name are specified, a warning message KDJE55535-W is output. If such multiple named queries with the same name are specified in Cosminexus JPA Provider, there is no certainty about which query will be operated.

The following is an example of defining `@NamedQuery`. In this example, `@NamedQuery` is used and the query is registered beforehand with the name `findAllCustomersWithName`. By passing the named query name registered in the `createNamedQuery` method of the application, the query registered beforehand is obtained and used.

- Registering the query name in `@NamedQuery`

```
@NamedQuery(
      name="findAllCustomersWithName",
      query="SELECT c FROM Customer c WHERE c.name LIKE :custName"
)
@Entity
public class Customer {
...
}
```

- Example of coding the named query with the `createNamedQuery` method

```
@Stateless
public class MySessionBean {
...
  @PersistenceContext
  public EntityManager em;
  ...
  public void doSomething() {
    ...
    Query q = em.createNamedQuery("findAllCustomersWithName")
                                     .setParameter("custName", "Smith");
  }
}
```

Note that with the same persistence unit, you can also use the named query defined in another entity.

(2) How to specify the parameters

When you generate a query with JPQL, you can use a parameter in the conditional expression coded in the `WHERE` clause and set the value dynamically. Set the parameter value with the `setParameter` method of the `Query` interface. The parameters include location parameters and named parameters. The following is a description of each parameter:

Location parameters

Code a combination of question mark (`?`) and a numeric value in the location where you want to insert the parameter in the `WHERE` clause. Specify the parameter value with the `setParameter` method of the `Query` interface. The format and example of coding the location parameters is as follows:

- Coding format

```
Query setParameter(int location, Object value)
```

- Coding example

```
Query q = em.createQuery(
        "SELECT c FROM Customer c WHERE c.balance < ?1")
.setParameter(1, 20000);
```

Named parameters

Code a combination of `':'` (colon) and any string (however, excluding the characters 0 to 9) in the location where you want to insert the parameter in the `WHERE` clause. Specify the parameter value with the `setParameter` method of the `Query` interface. The format and example of coding the named parameters is as follows:

- Coding format

```
Query setParameter(String parameter-name, Object value)
```

  Do not specify `':'` (colon) at the beginning of the parameter name. Also, the named parameters are case sensitive.

- Coding example

```
Query q = em.createQuery(
        "SELECT c FROM Customer c WHERE c.name LIKE :custName")
        .setParameter("custName", "John");
```

Note the following when you use parameters:

- Do not mix and use the location parameters and named parameters in one query. In Cosminexus JPA Provider, the operations might not function properly if the parameters are mixed.
- In the `setParameter` method, specify as follows when you want to specify the `java.util.Date` type or `java.util.Calendar` type objects as parameter values. Note that the time type of the parameter value must be specified using the enumeration type `TemporalType`. For details, see the *Java documentation*.
  - `Query setParameter(int` *location*`, Date` *date*`, TemporalType` *time-type*`)`
  - `Query setParameter(String` *parameter-name*`, Date` *date*`, TemporalType` *time-type*`)`
  - `Query setParameter(int` *location*`, Calendar` *calendar*`, TemporalType` *time-type*`)`
  - `Query setParameter(String` *parameter-name*`, Calendar` *calendar*`, TemporalType` *time-type*`)`

(3) Obtaining and executing the query results

You use the following methods of the `Query` interface to execute the generated query, to return the query results, and to execute update query. The following points describe each method:

(a) Object getSingleResult()

Use this method to return the query result as a single object.

When you execute this method, the data is searched. As a result of the search, the single hit line is stored in the entity object and returned with the `Object` type. The return value of the `Object` type must be cast in the target entity class.

If multiple lines are hit, the `NonUniqueResultException` exception occurs. If no lines are hit, the `NoResultException` exception occurs.

(b) List getResultList()

Use this method to return the query result as a list.

When you execute this method, the data is searched. As a result of the search, the multiple hit lines are stored in the entity object and returned in a list. Multiple lines are assumed to be returned as execution results; therefore, if no lines are hit, an empty list is returned.

(c) int executeUpdate()

Use this method to execute the update query.

When you execute this method, a query will be executed to simultaneously delete or update multiple lines in a table. The execution result returns the number of lines hit.

9.16.2 Procedure for referencing and updating the database with the native query

With Cosminexus JPA Provider, as a query language other than JPQL, you can directly code a database-specific native query and reference or update a database.

This section explains how to use native queries.

(1) How to obtain the Query object

To use native query for obtaining the `Query` object, you use the following methods of the `EntityManager` interface provided by Cosminexus JPA Provider.

(a) Query createNativeQuery(String SQL statement)

An example of coding `createNativeQuery` is as follows. In the argument, specify the native query you want to execute.

```
Query q = em.createNativeQuery(
                   "SELECT o.id, o.quantity, o.item " +
                   "FROM Order o, Item i " +
                   "WHERE (o.item = i.id) AND (i.name = 'widget')");
```

(b) Query createNativeQuery(String SQL statement, Class result storing class)

An example of coding `createNativeQuery` is as follows. Specify the native query you want to execute in the first argument and the class object for storing the execution result in the second argument.

```
Query q = em.createNativeQuery(
                   "SELECT o.id, o.quantity, o.item " +
                   "FROM Order o, Item i " +
                   "WHERE (o.item = i.id) AND (i.name = 'book')",
                   com.hitachi.Order.class);
```

In this example, if a query is executed, the collection of all the `Order` entities is returned for the item named `book`.

Note that if the query results specified in the `SELECT` clause and the class object specified in the argument are inconsistent, an exception occurs.

(c) Query createNativeQuery(String SQL statement, String result set mapping name)

Specify the native query you want to execute in the first argument and the result set mapping name for storing the execution result in the second argument. Specify the result set mapping with `@SqlResultSetMapping`. For details on result set mapping, see *(2) Result set mapping*.

An example of defining @SqlResultSetMapping and an example of coding createNativeQuery are as follows:

- Example of defining @SqlResultSetMapping

```
@SqlResultSetMapping(name="BookOrderResults",
      entities=@EntityResult(entityClass=com.hitachi.Order.class))
```

- Example of coding createNativeQuery

```
Query q = em.createNativeQuery(
                   "SELECT o.id, o.quantity, o.item " +
                   "FROM Order o, Item i " +
                   "WHERE (o.item = i.id) AND (i.name = 'book')",
                   "BookOrderResults");
```

In this example, if a query is executed, the collection of all the Order entities is returned for the item named book. By using @SqlResultSetMapping, you can obtain the same results as the coding example when @NamedQuery is used in *9.16.1(1) How to obtain the Query object*.

Note that if the query results specified in the SELECT clause and the @SqlResultSetMapping settings specified in the argument are inconsistent, an exception occurs.

(d) Query createNamedQuery(String query name)

Like JPQL, you can use the createNamedQuery method for a native query. In the native query, specify the named native query name in the argument.

You define the named native query by assigning @NamedNativeQuery in any entity class. In the query name argument, use the name specified in the name attribute of @NamedNativeQuery.

In Cosminexus JPA Provider, you cannot specify multiple named queries with the same name. If multiple named queries with the same name are specified, a warning message KDJE55522-W is output. If such multiple named queries with the same name are specified in Cosminexus JPA Provider, there is no certainty about which query will be operated.

The following is an example of using the createNamedQuery method. In this example, @NamedNativeQuery is used and the query is registered beforehand with the name findBookOrder. By passing the named query name registered in the createNamedQuery method of the application, the query registered beforehand is obtained and used.

- Registering the query name in @NamedNativeQuery

```
@NamedNativeQuery( name="findBookOrder",
       query="SELECT o.id, o.quantity, o.item " +"
             "FROM Order o, Item i " +
             "WHERE (o.item = i.id) AND (i.name = 'book')")
)
@Entity
public class Order {
...
}
```

- Example of coding the named native query with the createNamedQuery method

```
@Stateless
public class MySessionBean {
  ...
  @PersistenceContext
  public EntityManager em;
  ...
  public void doSomething() {
    ...
    Query q = em.createNamedQuery("findBookOrder ")
  }
}
```

Note that with the same persistence unit, you can also use the named native query defined in another entity.

(2) Result set mapping

Result set mapping is a functionality used for mapping and receiving the execution results of the native query in any entity class and for receiving the execution results of the native query using the scalar value.

With result set mapping, the mapping information of the column values obtained as the execution results of the native query is assigned to any entity class by specifying `@SqlResultSetMapping`.

(a) Coding format of @SqlResultSetMapping

The coding format of `@SqlResultSetMapping` is as follows:

```
@SqlResultSetMapping(
  name= Result-set-mapping-name,
  entities= Specify-the-entity-class-for-mapping-the-result-(@EntityResult-a
rray),
  columns= Specify-the-column-for-mapping-the-result-(@ColumnResult-array) )
```

name attribute

Specify the result set mapping name.

entities attribute

Specify the `@EntityResult` array. The coding format of `@EntityResult` is as follows:

```
@EntityResult(
    entityClass= Specify-the-class-for-mapping-the-result,
    fields= Specify-the-field-for-mapping-the-result-(@FieldResult-array
) )
```

In the `entityClass` attribute of `@EntityResult`, specify the entity class for storing the column value. In the `field` attribute, specify the `@FieldResult` array.

The coding format of `@EntityResult` is as follows:

```
@FiledResult(
    name= Name-of-persistent-property-(or-field)-of-class,
    column= Column-name-of-SELECT-clause-(or-optional-name) )
```

In the `name` attribute of `@FieldResult`, specify the persistence field name of the entity class specified in the `entityClass` attribute of `@EntityResult`. Also, in the `column` attribute, specify the column name.

columns attribute

The `columns` attribute specifies the `@ColumnResult` array to receive the execution results of the native query as a scalar value without being stored in the entity class. If you do not need to extract the scalar value, you need not specify the `columns` attribute. In the `name` attribute of `@ColumnResult`, specify the column name for extracting the value. The coding format of `@ColumnResult` is as follows:

```
@ColumnResult(
    name= Column-name-of-SELECT-clause-(or-optional-name) )
```

Note that you can also specify the column name with the alias name specified by AS. If the SELECT clause contains multiple columns with the same name, use the optional name of the column.

(b) Example of usage

In the following example, the query result for employee number 12003 is mapped from the employee table (Employee) and department table (Department) to any entity class (EmployeeSetmap) and the scalar value (EMP_MONTHLY_SALARY column) is received.

Specify the result set mapping name (NativeQuerySetMap) in the second argument of createNativeQuery and execute the result set mapping.

- Example of coding the native query using result set mapping

```
query = em.createNativeQuery(
                "SELECT e.EMPLOYEE_ID AS EMP_EMPLOYEE_ID, " +
                "e.EMPLOYEE_NAME AS EMP_EMPLOYEE_NAME, " +
                "d.DEPARTMENT_NAME AS DEP_DEPARTMENT_NAME, " +
                "e.MONTHLY_SALARY AS EMP_MONTHLY_SALARY  " +
                "FROM EMPLOYEE e, DEPARTMENT d " +
                "WHERE e.DEPARTMENT_ID = d.DEPARTMENT_ID " +
                "AND e.EMPLOYEE_ID = 12003",
                "NativeQuerySetMap");
```

- Optional entity class for storing the execution result of the native query

```
@Entity
public class EmployeeSetmap implements Serializable {
...
@Id
public int getEmployeeId() { return employeeId; }
public String getEmployeeName() { return employeeName; }
public String getDepartmentName() { return departmentName; }
...
}
```

- Example of coding @SqlResultSetMapping

```
@SqlResultSetMapping(
       name="NativeQuerySetmap",
       entities={ @EntityResult(
                         entityClass=EmployeeSetmap.class,
                         fields={ @FiledResult(
                                     name="employeeId",
                                     column="EMP_EMPLOYEE_ID"),
                                 @FiledResult(
                                     name="employeeName",
                                     column="EMP_EMPLOYEE_NAME"),
                                 @FiledResult(
                                     name="departmentName",
                                     column="DEP_DEPARTMENT_NAME") } ) },
       columns={ @ColumnResult(
                         name="EMP_MONTHLY_SALARY") }
```

Note that the `Object` type array for the result of executing the native query by executing `@SqlResultSetMapping` is as follows:

`Object[0]`
: Object of `EmployeeSetmap` class
: (Values of `EMP_EMPLOYEE_ID` column, `EMP_EMPLOYEE_NAME` column, and `DEP_DEPARTMENT_NAME` column are stored in each field)

`Object[1]`
: Value of `EMP_MONTHLY_SALARY` column

(3) How to specify the parameters

As in the case of JPQL, the native query can set a value dynamically by using parameters. Code a combination of question mark (`?`) and a numeric value in the location where you want to insert the parameter in the `WHERE` clause. Specify the parameter value with the `setParameter` method of the `Query` interface. However, you cannot use the named parameters of JPQL with the native query.

The coding format of the parameters is as follows:

```
Query setParameter(int location, Object value)
```

(4) Obtaining and executing the native query results

As in the case of JPQL, use the following `Query` interface methods for obtaining and executing the native query results:

- `Object getSingleResult()`
- `List getResultList()`
- `int executeUpdate()`

For details on these methods, see *9.16.1 Procedure for referencing and updating the database with JPQL*.

9.16.3 Specifying the range of query result items

You can obtain only an optionally specified item from multiple query results and only the query result specified in the starting position. To obtain this information, use the `Query` interface methods. This section describes the methods.

(1) setMaxResults method

To obtain only an optionally specified item from multiple query results, use the `setMaxResults` method of the `Query` interface. The coding format of the `setMaxResults` method is as follows:

```
Query setMaxResults(int maximum-number-of-search-results)
```

In the method argument, specify the maximum number of search results.

(2) setFirstResult method

To obtain only the query result specified in the starting position, use the `setFirstResult` method. The coding format of the `setFirstResult` method is as follows:

```
Query setFirstResult(int starting-position-of-search-results)
```

In the method argument, specify the starting position of the search results. Specify a numeric value beginning with 0 as the starting position.

(3) Example usage of the Query interface methods

The examples of usage of the `setMaxResults` method and `setFirstResult` method are as follows. In this example, five `Employee` objects are obtained sequentially from the tenth employee with the highest monthly salary (`e.monthlySalary`) from the employee data (`Employee`).

```
Query query = em.createQuery( "SELECT e FROM Employee AS e " +
                              "ORDER BY e.monthlySalary DESC" )
                              .setFirstResult(9)
                              .setMaxResults(5);
List resultList = query.getResultList();
```

(4) Notes

When you use the `setFirstResult` method to specify the starting position, the time period from the invocation of the `getResultList` method and `getSingleResult` method until the obtaining of the result values varies depending on the value specified in the argument. Typically, the time taken until the result value returns is in proportion to the value of the starting position specified in the argument.

9.16.4 Specifying the flush mode

You can specify how the query would handle an uncommitted operation executed for an entity object. This is called *specifying the flush mode*.

You set the flush mode with the `setFlushMode` method of the `javax.persistence.Query` interface. With Cosminexus JPA Provider, you can only set `AUTO` as the value. You cannot specify `COMMIT`.

In FlushModeType.AUTO

When a query is executed in a transaction, the changes in all the entities existing in the persistence context that affect the query results are applied to the query results.

This setting is applied to a query regardless of the flush mode of the `setFlushMode` method of the `EntityManager` interface.

9.16.5 Specifying a query hint

During the execution of a query, you can specify a query hint as a vendor-dependent hint. With Cosminexus JPA Provider, you specify a query hint when you use the pessimistic lock.

Specify the query hint at the following locations:

- The argument of the `setHint()` method of the `Query` object
- The argument `@Hint` of `@NamedQuery`
- The `<hint>` tag that is a lower element of the `<named-query>` tag in the O/R mapping file

If you specify a value outside the range specifiable in the query hint, an exception occurs. Note that the specified value is not case sensitive.

The time when an exception occurs differs depending on the location where the query hint is specified. The timing for the occurrence of an exception is as follows:

- In the `setHint()` method, when the application query is executed
- In the annotation, when the application is deployed
- In the O/R mapping file, when the application starts

For details on the query hints supported by CJPA provider, see *9.22 Scope of support for the annotations included in the javax.persistence package* when using annotations, and see *13.3 O/R mapping files* when using O/R mapping files.

9.16.6 Notes on executing a query

This section describes the notes on executing a query.

- In the `setMaxResults` method or `setFirstResult` method, if a query in which collection contains `FETCH JOIN` is executed, the results might be incorrect.
- The `Query` methods other than the `executeUpdate` method need not be executed in a transaction. Particularly, the `getResultList` and `getSingleResult` methods need not be executed in a transaction.
- When the query is executed with `EntityManager` of the transaction scope persistence context, the resulting entity has the `detached` state. When the query is executed with `EntityManager` of the extended persistence context, all the entities have the `managed` state.
- The runtime exceptions other than `NoResultException` and `NonUniqueResultException` thrown from the `Query` interface methods roll back the current transaction.

9.17 JPQL coding method

This section describes JPQL coding method.

9.17.1 JPQL syntax

A JPQL statement includes the `SELECT` statement, `UPDATE` statement, and `DELETE` statement.

You can dynamically specify a JPQL statement or statically define a JPQL statement using the annotation and O/R mapping file tags. You can also specify parameters in all the JPQL statements.

JPQL is a typed language and all the expressions have a type. The abstract schema type defined in the expression configuration and the identification variable, the type for evaluating the persistence fields and relationships, and the literal type configure the expression types. For details on the syntax, see *Appendix G BNF for JPQL*.

Important note

In Cosminexus JPA Provider, an exception might occur if you use JPQL that does not conform to the BNF syntax. Even if an exception does not occur, the operation might not function properly. Also, even if you use JPQL that conforms to the BNF syntax, the operations might not function properly if the relevant functionality is not supported in the database used.

(1) Abstract schema type

With JPQL, a query is issued for an entity. Therefore, the abstract schema of the target entity must be defined with the query. The abstract schema type of the entity is defined based on the entity class and O/R mapping information provided by the annotations or the O/R mapping file.

The abstract schema type indicates an entity class. The fields and the O/R mapping information of the annotation configure the entity class.

The abstract schema type of the entity has the following fields:

- **State field**
  A state field is a field or property in which a relationship-based relation does not exist in the persistence field or persistence property of the entity class.
- **Relation field**
  A relation field is a persistence field or persistence property associated by relationship with the entity class. If the relationship is OneToMany or ManyToMany, the field is collection.

(2) Abstract schema name

With JPQL, you must specify the abstract schema type in order to indicate an entity. The name used for indicating the abstract schema type is called the *abstract schema name*. You define the abstract schema name in the `name` attribute of `@Entity` (or in the `name` attribute of the `<entity>` tag in the O/R mapping file). If the `name` attribute is not specified, the name becomes the class name of the entity class (without package name).

The abstract schema name is specific to each persistence unit.

(3) Query domain

The query domain can reference the abstract schema type of all the entities defined in the persistence unit. The abstract schema type of the other related entities can be referenced using the relation field defined in the abstract schema type.

9.17.2 SELECT statement

The SELECT statement contains the following clauses:

- SELECT clause
  Specifies the value based on the type or set function of the object to be searched. Make sure that you specify the SELECT clause.
- FROM clause
  Specifies the range for which a search is applied. Make sure that you specify the FROM clause.
- WHERE clause
  Used for narrowing down the search results. You can omit the WHERE clause.
- GROUP BY clause
  Used for grouping the search results. You can omit the GROUP BY clause.
- HAVING clause
  Used for filtering the grouped search results. You can omit the HAVING clause.
- ORDER BY clause
  Used for the ordered classification of the search results. You can omit the ORDER BY clause.

If the SELECT clause and FROM clause are not specified in the SELECT statement, an exception occurs.

9.17.3 SELECT clause

The SELECT clause expresses the query results. One or more values are returned from the SELECT clause of the query. You specify the following elements, demarcated by commas, in the SELECT clause. Note that the cluster demarcated by commas is called a *select expression*.

- Identifier of the abstract schema or persistence field that is assigned an identifier
- Path expression
- Set function
- Constructor expression

Specify the DISTINCT keyword when you want to exclude the duplicated values from the query result.

An example of coding the SELECT clause is as follows:

```
SELECT e.employeeName, e.monthlySalary
FROM Employee AS e
WHERE e.monthlySalary  < 150000
```

(1) Constructor expression

The constructor expression is used in the select expression of the `SELECT` clause that returns one or more Java instances. The generated name specifies the fully qualified name.

The constructor expression can be obtained as a combination of some or all entity columns and the other related entity columns. Note that the class that stores this result need not be an entity.

Specify the syntax of the constructor expression by assigning the `NEW` operator in the select expression. If the class that stores the result is an entity, the state of the entity class instance becomes `new`. An example of coding the constructor expression is as follows:

```
SELECT NEW com.hitachi.jpa.test.entity.EmployeeTmp
(e.employeeId, e.employeeName, d.departmentName)
FROM Department AS d, d.employees AS e
WHERE e.employeeId = 12003
```

(2) Set function

You can use the set function with the `SELECT` clause. The following table lists and describes the available set functions.

Table 9–19: Set functions available in the SELECT clause of JPQL

Set function	Argument	Result type	Result when the applied value does not exist
`AVG`	Numeric type state field #	`Double` type	`null`
`MAX`	Field type that can specify the order (numeric type, string type, character type, or date type) #	Type of the applied field	`null`
`MIN`	Field type that can specify the order (numeric type, string type, character type, or date type) #	Type of the applied field	`null`
`SUM`	Numeric type state field #	• For the integer type: `Long` type • For the floating point type: `Double` type • For the `BigInteger` type: `BigInteger` type • For the `BigDecimal` type: `BigDecimal` type	`null`
`COUNT`	Identification variable (when you specify the path expression in the argument, specify the state field or relation field)	`Long` type	`0`

Note

Regardless of whether `DISTINCT` is specified, the null value is removed before the set function is applied.

#

If the path expression is specified in the argument, you cannot specify the relation field.

For details on the syntax of the set function expression, see *Appendix G BNF for JPQL*.

(3) Execution results of the SELECT clause

The type of the query results defined in the SELECT clause of the query is one of the following. If multiple types are present, the types are serialized.

- Entity abstract schema type
- Field type
- Set function results
- Constructor expression results

The result type of the SELECT clause is defined according to the result type of the select expression included in the SELECT clause. If multiple select expressions are used in the SELECT clause, the query result is Object[] type. The elements of this result match the order specified in the SELECT clause and with the result type of each select expression.

9.17.4 FROM clause

This section describes the FROM clause.

(1) Range variable declaration and identification variables

A range variable declaration is a declaration that codes the logical name of the entity class in the FROM clause and then specifies AS and the identifier (AS can be omitted). The identifier of this range variable declaration is called the *identification variable*. An example of range variable declaration and identification variable is as follows:

```
SELECT ... (omitted) ...

FROM  Department AS dep

WHERE ... (omitted) ...
```

The part Department AS dep is the range variable declaration. Also, dep is the identification variable. The syntax of the range variable declaration is as follows:

```
range_variable_declaration ::=
abstract_schema_name [AS] identification_variable
```

The syntax of the identification variable in the range variable declaration is the same as the SQL syntax. A description of the syntax is as follows:

- The use of the keyword AS is optional.
- You cannot omit the identification variable. However, you can omit the AS specified between the abstract schema and the identification variable. Specify the identification variable in the FROM clause.
- A reserved identifier cannot be used. If used, an exception occurs.
- The same name as another entity in the same persistence unit cannot be used. With Cosminexus JPA Provider, the operations might not function properly if an entity with the same name is used.
- The identification variable is not case sensitive.

- You cannot specify the same name as the abstract schema name. With Cosminexus JPA Provider, the operations might not function properly if the abstract schema with the same name is specified.
- The syntax must begin with a Java identifier character and all the other characters must be partial characters of the Java identifier. If other characters are specified, an exception occurs. For the first character, use a character by which the return value of the `Character.isJavaIdentifierStart` method becomes `true` (including underscore (_) and dollar sign (`$`) characters). For the characters other than the first character, use a character by which the return value of the `Character.isJavaIdentifierPart` method becomes `true` (however, question mark (`?`) is a reserved word in JPQL and cannot be used).
- The reserved words in JPQL are as follows:
  `SELECT`, `FROM`, `WHERE`, `UPDATE`, `DELETE`, `JOIN`, `OUTER`, `INNER`, `LEFT`, `GROUP`, `BY`, `HAVING`, `FETCH`, `DISTINCT`, `OBJECT`, `NULL`, `TRUE`, `FALSE`, `NOT`, `AND`, `OR`, `BETWEEN`, `LIKE`, `IN`, `AS`, `UNKNOWN`, `EMPTY`, `MEMBER`, `OF`, `IS`, `AVG`, `MAX`, `MIN`, `SUM`, `COUNT`, `ORDER`, `BY`, `ASC`, `DESC`, `MOD`, `UPPER`, `LOWER`, `TRIM`, `POSITION`, `CHARACTER_LENGTH`, `CHAR_LENGTH`, `BIT_LENGTH`, `CURRENT_TIME`, `CURRENT_DATE`, `CURRENT_TIMESTAMP`, `NEW`, `EXISTS`, `ALL`, `ANY`, and `SOME`
  Note that `UNKNOWN` is not used in JPA 1.0, but is a reserved word in Cosminexus JPA Provider.

(2) Path expression

A path expression is an expression in which a period (`.`) is added after the identification variable and is used for continuing the state field or relation field. Therefore, the path expression type becomes the state field or relation field type.

You can build up a path expression further from the relation fields obtained by tracing the path expression. However, if the path expression type that forms the base is a collection relation field, you cannot build a path expression. Creating the path expression from the collection type becomes a structural error.

Note that if the relation field in the middle of the path expression is a null value, the path assumes that the value does not exist, so the query result is not affected.

You can use the path expression with the syntax using `inner join`. For details on the path expression syntax, see *Appendix G BNF for JPQL*.

Reference note

The types of relation fields are as follows:

- A collection relation field (`collection_valued_association_field`) is one in which the relation field is specified using a collection. The collection relation field is indicated by OneToMany or ManyToMany relation.
- A non-collection relation field (`single_valued_association_field`) is one in which the relation field is specified using `single-valued`. The non-collection relation field is indicated by OneToOne or ManyToOne relation.
- The embedded class field is the field name of the entity corresponding to the `embedded` class.

(3) Joins expression

The Joins expression is available in the `FROM` clause. The following table lists and describes the available Joins expressions.

Table 9–20: Joins expressions available in the FROM clause

Joins expression	Contents	Syntax name of BNF syntax [#1]
`Inner Joins`	This expression binds two entity classes and extracts only the entity objects with relation, in the fields to be related.	`join`, `join_spec`
`Left Outer Joins`	This expression binds two entity classes and extracts the entity objects with relation as well as the entity objects without relation, in the fields to be related.	`join`, `join_spec`
`Fetch Joins`	This expression binds two entity classes in the related fields. Note that a relation exists between the entities due to relationship, so specify only one entity class in the `Select` clause. [#2]	`fetch_join`

#1

For details on the syntax name of the BNF syntax, see *Appendix G BNF for JPQL*.

#2

When the entity is obtained using `Fetch Joins`, the information about the relation destination entity specified on the right is obtained at the same time as the execution of the query. As a result, you can obtain the relation destination information without dependence on the fetch strategy. An example of coding `Fetch Joins` is as follows:

```
SELECT emp FROM Employee AS emp JOIN FETCH emp.company
```

Notes on using the Join expression

The notes on using the Join expression are as follows:

Notes on `Inner Joins`

The keyword `INNER` is used optionally.

Notes on `Left Outer Joins`

The keyword `OUTER` is used optionally.

Notes on `Fetch Joins`

- With Fetch Joins, the entity information of two entities is specified in one entity. The information of the specified entity and the information of another entity related to that entity are bound.

  Note that in Cosminexus JPA Provider, you specify entities with relationship because the entities are bound by one entity information. If entities without relationship are specified, an exception occurs.
- The relation referenced on the right side of `JOIN FETCH` must be a relation belonging to the entity returned as a query result. In Cosminexus JPA Provider, if the relation does not belong to the entity, an exception occurs.
- An identifier cannot be specified in the entity referenced on the right side of `JOIN FETCH`. Therefore, the entity cannot be referenced in the query.

(4) Declaring the collection members

The identification variables for declaring the collection members are declared by using the reserved identifier `IN`. You can obtain the collection value for the identification variable defined in the collection member expression, by using the path expression. An example of coding the collection member expression is as follows:

```
SELECT emp.employeeId, emp.employeeName, dep.departmentName
FROM Department AS dep, IN (dep.employees) AS emp
WHERE dep.departmentId = 3
```

For details on the syntax of the collection member expression, see *Appendix G BNF for JPQL*.

(5) Notes

This section describes the notes on the `FROM` clause.

- Effect of the identification variable
  Even if the identification variable declared in the `FROM` clause is not used in the `WHERE` clause, the declared identification variable is applied to the query result.
- Notes on polymorphism
  JPQL is a polymorphism. The specific entity class instances referenced explicitly by the `FROM` clause as well as the subclasses become the target. Therefore, the instances that can be obtained using a query include subclass instances that satisfy the query conditions.

9.17.5 WHERE clause

The `WHERE` clause is made up of conditional expressions used for searching the objects or variables that satisfy the expression. The `WHERE` clause specifies the result of the `select` statement and the scope of the `update` and `delete` operations.

(1) Conditional expressions that can be used in the WHERE clause

The following table describes the conditional expressions that can be used in the `WHERE` clause.

Table 9–21: Conditional expressions that can be used in the WHERE clause

Expression	Contents	Syntax name of BNF syntax#
`BETWEEN`	Evaluates whether the value is included in the scope specified for the field.	`between_expression`
`IN`	Evaluates that the value matches with some value specified in the field.	`in_expression`
`LIKE`	Evaluates that the value matches with the string after a wild card sign is allocated in the field.	`like_expression`
`IS [NOT] NULL`	Tests whether the value is a null value.	`null_comparison_expression`
`IS [NOT] EMPTY`	Tests whether the specified collection value is empty.	`empty_collection_comparison_expression`
`[NOT] MEMBER [OF]`	Tests whether the specified collection value is a collection member. When an empty collection is expressed, the value of the `MEMBER OF` expression is `FALSE` and the value of the `NOT MEMBER OF` expression is `TRUE`.	`collection_member_expression`
`EXISTS`	Determines the sub-query result. If one or more values exist, `true` is returned and in other cases, `false` is returned.	`exists_expression`
`ALL`	Compares a value with all the values returned by the sub-query.	`all_or_any_expression`
`ANY (SOME)`	Compares a value with some value returned by the sub-query.	`all_or_any_expression`
Sub-query	Results can be coded based on `select`.	`subquery`
Functional expression	See *(2) Functional expressions*.	See the syntax of the relevant function

#
For details on the syntax name of the BNF syntax, see *Appendix G BNF for JPQL*.

The notes on using the `WHERE` clause are as follows:

- **IN expression**
    - If the field to be evaluated is `null` or `unknown`, the expression value becomes `unknown`.
    - The list that defines the value set, demarcated by commas, for the `IN` expression, must have at least one or more elements. In Cosminexus JPA Provider, an exception occurs if one or more elements do not exist.
    - The field to be evaluated must have a string, numeric, or enum value.
    - The literal and input parameter values, and the sub-query results must have the same abstract schema type as the field to be evaluated. In Cosminexus JPA Provider, an exception occurs if the abstract schema type is not the same.
- **LIKE expression**
    - If the value of `string_expression` or `pattern_value` indicated in the BNF syntax is `null` or `unknown`, the value of the `LIKE` expression becomes `unknown`.
    - If `escape_character` indicated in the BNF syntax is specified and if the value is `null`, the value of the `LIKE` expression becomes `unknown`.
    - In the specified pattern value, an underscore (_) corresponds to one character and a percent (`%`) character corresponds to continuous characters (including continuous null characters). Note that the other characters indicate the search string.
    - The optional escape character is a string literal or character. Use the escape character to interpret the underscore and percent characters of the pattern string as the standard characters.
- **IS [NOT] NULL expression**
  If the specified collection value is `null` or `unknown`, the value of the `IS [NOT] EMPTY` expression becomes `unknown`.
- **[NOT] MEMBER [OF] expression**
  If the specified value is `null` or `unknown`, the return value becomes `unknown`.
- **ALL expression or ANY (SOME) expression**
    - If the conditional operation is neither `true` nor `false`, the value becomes `unknown`.
    - The comparison operators that can be used together are =, <, <=, >, >=, <>.
    - The sub-query result type must be the same as the comparison operator type. In Cosminexus JPA Provider, an exception occurs if the types are different.
    - `SOME` is the synonym of `ANY`. #

  # This note is only applicable to `ANY` (`SOME`) expression.
- **Sub-query**
  Used in the `WHERE` clause and `HAVING` clause. Cannot be used in the `FROM` clause.

(2) Function expression

With JPQL, the functions listed in the following table are available in the `WHERE` clause and `HAVING` clause. If the value of the function expression argument is `null` or `unknown`, the value of the function expression becomes `unknown`.

Table 9–22: Functions available with JPQL

Category	Functions	Return value	Supplement to arguments
String function	`CONCAT`	Concatenated string of the argument	--

Category	Functions	Return value	Supplement to arguments
	`SUBSTRING`	String with the start position and length specified in the argument	The second and third arguments specify the start position and length of the returned sub-string as an integer. The position of the beginning of the string is 1.
	`TRIM`	String with a specific character removed	If no character is to be removed, a space (blank) is assumed. The optional `trim` character is the `character` input string. If the trim method is not specified, `BOTH` is used.
	`LOWER`	De-capitalized string	--
	`UPPER`	Capitalized string	--
	`LENGTH`	Integer value of the string length	--
	`LOCATE`	Integer value of the given string and the position where the string is first found after searching from a specified position (if the string is not found, `0`)	The first argument indicates the string to be searched and the second argument indicates the searched string. The optional third argument indicates the position of the character to start the search (by default, the search is performed from the beginning). The start position of the string is 1.
Arithmetical function	`ABS`	Absolute value with the same type as the function argument (`Integer`, `Float`, or `Double`)	A numerical value is passed as an argument.
	`SQRT`	Square root of the numerical value passed in the argument (real number)	A numerical value is passed as an argument.
	`MOD`	Remainder of two numerical values passed in the argument (integer)	Two integers are passed.
	`SIZE`	Number of collection elements (integer)	Collection is passed.
Date and time function	`CURRENT_DATE`	Database date	--
	`CURRENT_TIME`	Database time	--
	`CURRENT_TIMESTAMP`	Database timestamp	--

Legend:

--: Not applicable

(3) Notes

This section describes the notes on the `WHERE` clause.

(a) Priority of the operators

The priority of the operators is as follows:

1. Period (`.`)
2. Arithmetic operators
   Unary operation (`+`, `-`), multiplication and division (`*`, `/`), addition and subtraction (`+`, `-`)
3. Comparison operators
   =, >, >=, <, <=, <> (not equal), `[NOT] BETWEEN`, `[NOT] LIKE`, `[NOT] IN`, `IS [NOT] NULL`, `IS [NOT] EMPTY`, `[NOT] MEMBER [OF]`

4. Logical operators
   `NOT`, `AND`, `OR`

(b) Notes on conditional expressions

- A conditional expression is made up of a conditional expression, comparison expression, and a logical operator, and the evaluation result is made up of the path expression that forms a boolean value, boolean literal, and input parameters of `boolean` type.
- An arithmetic expression is available in the comparison expression. An arithmetic expression is made up of other arithmetic expressions and the four arithmetic operations. The result is made up of the path expression that forms a numerical value, number literals, and the input parameters of the numerical type.
- An arithmetic operation uses numeric promotion.
- To specify the evaluation order of the expression, you can enclose the evaluation order in parentheses.
- The set function is only available in the conditional expression of the `HAVING` clause.
- In a conditional expression, do not use fields mapped as a serialized format or `lobs`.

(c) Notes on using literals

- Enclose a character literal in single quotes (example: `'`*literal*`'`). For a string literal containing single quotes, use two single quotes.
- As in the case of the `JavaString` literal, use Unicode character encoding for the query string literal.
- The use of Java escape expression is not supported in the query string literal.
- For the enum literal, you can use the literal of the Java enum literal syntax. The enum literal requires an enum class name.
- The boolean literal is `TRUE` and `FALSE`. This literal is not case sensitive.

(d) Notes on identification variables

- The identification variable is an identifier declared in the `FROM` clause of the query, so the identification variable cannot be declared in another clause. If the variable is declared in a clause other than the `FROM` clause, an exception occurs. For all the identification variables used in the `WHERE` or `HAVING` clause of the `SELECT` statement or `DELETE` statement, use the identification variables defined in the `FROM` clause.
- The range of the identification variables is determined by the `WHERE` clause and `HAVING` clause. An identification variable can specify the collection members and instances of the abstract schema type of the entity, but cannot specify the collection itself. If a collection is specified, an exception occurs.

(e) Notes on path expressions

In the path expression of the collection in the `WHERE` or `HAVING` clause, do not use values other than `IS [NOT] EMPTY` expression and `[NOT] MEMBER [OF]` expression, or arguments for the `SIZE` operation as a part of the conditional expression.

(f) Notes on input parameters

- The input parameters are available in the `WHERE` clause or `HAVING` clause of the query.
- Do not mix the location parameters and named parameters in one query. If the parameters are mixed, the operations might not function properly.
- If the value of the input parameters is `null`, the value returned by the comparison operation or arithmetic operation containing the input parameters becomes `unknown`.

For details on how to use the location parameters and the named parameters, see *9.16.1(2) How to specify the parameters*.

9.17.6 GROUP BY clause and HAVING clause

With the `GROUP BY` clause, the query results are compiled for each group. With the `HAVING` clause, you can specify conditions for narrowing down the query results further. After specifying the group, specify the HAVING clause conditions.

If a query contains both the `WHERE` clause and `GROUP BY` clause, the `WHERE` clause is executed first, the format is adjusted with the `GROUP BY` clause, and then filtering is performed according to the `HAVING` clause.

The items other than the set function appearing in the `SELECT` clause must also be specified in the `GROUP BY` clause. With grouping, the null value is also included and is handled as a condition. The notes related to the `GROUP BY` clause and `HAVING` clause are as follows:

- Entity-based grouping can be performed, but serialized fields and lob fields cannot be included. If serialized fields and lob fields are specified with Cosminexus JPA Provider, an exception occurs.
- In the `HAVING` clause, search conditions are specified for the group items, so the set function applicable to the group items must be specified. With Cosminexus JPA Provider, if the search conditions are not specified, an exception occurs.
- Do not use the `HAVING` clause when the `GROUP BY` clause does not exist. If used, an exception occurs.

An example of coding the `GROUP BY` clause and `HAVING` clause is as follows:

```
SELECT e.department.departmentId
FROM Employee AS e
GROUP BY e.department.departmentId
HAVING COUNT(e.department.departmentId) <= 2
```

For details on the syntax of the `GROUP BY` clause and `HAVING` clause, see *Appendix G BNF for JPQL*.

9.17.7 ORDER BY clause

With the `ORDER BY` clause, the objects and values are placed in order and then the query results are returned. An example of coding the `ORDER BY` clause is as follows:

```
SELECT e
FROM Employee AS e
ORDER BY e.monthlySalary DESC
```

A description of the `ORDER BY` clause is as follows:

- If one or more `order` items are specified, the priority is determined from the left to the right of the `orderby` item elements and the left-most `orderby` item has the highest priority.
- The `ORDER BY` clause is specified when `ASC` is ascending and `DESC` is descending. Note that the default value is `ASC`.
- The SQL rules are applied to the order when the `null` value exists.

- When the ORDER BY clause is used, the order of the query results is stored in the result of the query method.

For details on the syntax of the ORDER BY clause, see *Appendix G BNF for JPQL*.

Furthermore, the ORDER BY clause must satisfy the following conditions:

- The order items specified in the ORDER BY clause can be placed in order.
- The order items specified in the ORDER BY clause can be traced from the select expression of the SELECT clause.

If these conditions are not satisfied, an exception might occur in Cosminexus JPA Provider. However, even if an exception does not occur, the operations might not function properly.

9.17.8 Bulk UPDATE statement and Bulk DELETE statement

Update of multiple records in a batch is called *bulk update*. To perform bulk update, you use the Bulk UPDATE statement and Bulk DELETE statement. The bulk UPDATE and DELETE operations are applied to a stand-alone entity class or an entity class matching with the subclass. With the UPDATE statement, you define the identification variable in the UPDATE clause, and with the DELETE statement, you define the identification variable in the FROM clause and specify one type of entity.

Notes on using the Bulk UPDATE statement and Bulk DELETE statement:

- The delete operation is only applied to the entities of the specified class and the subclasses. The related entities are not cascaded.
- The update value specified in the update operation (new_value) must be compatible with the allocated field type. If the update value is not compatible, an exception occurs.
- The Bulk UPDATE statement executes the database update operation without setting the optimistic lock, so the processing related to the optimistic lock is not executed. Therefore, the value in the version column must be referenced and updated manually.

Important note

When you execute the Bulk UPDATE statement and the Bulk DELETE statement, note that mismatch occurs between the database and the entities of the activated persistence context. The operations in the Bulk UPDATE statement and the Bulk DELETE statement must be executed when separated from the transaction or when the transaction is started.

An example of coding the Bulk UPDATE statement and Bulk DELETE statement is as follows:

```
UPDATE EMPLOYEE
SET MONTHLY_SALARY = MONTHLY_SALARY + 1000
WHERE DEPARTMENT_ID = 3

DELETE FROM EMPLOYEE e
WHERE e.EMPLOYEE_NAME IS NULL AND e.monthlySalary  IS EMPTY
```

For details on the syntax of the Bulk UPDATE statement and Bulk DELETE statement, see *Appendix G BNF for JPQL*.

9.17.9 Notes on using JPQL

This section describes the notes on using JPQL.

(1) Notes on the null value

- **null value in the query result**
  - If the query result value corresponds to a relation field or state field with a `null` value, that `null` value is returned as the result of the query method.
  - You can use the `IS NOT NULL` syntax to remove the `null` value from the query result set.
  - In the fields defined using the numeric series primitive type of Java, a `null` value cannot be generated as a query result.
- **null value in the comparison and conditional expressions**
  - If the reference target does not exist in the database, the value is assumed to be `null`. To search a `null` value, you can only use the comparison conditions `IS NULL` and `IS NOT NULL`.
  - If the `null` value is used in a condition other than `IS NULL` and `IS NOT NULL` and if that result depends on the `null` value, the result becomes `unknown`.
  - The result of the comparison or arithmetic operation of the `null` value is always an `unknown` value.
  - The result of comparing two `null` values is an `unknown` value.
  - The result of the comparison or arithmetic operation with an `unknown` value is always an `unknown` value.
  - The boolean operator uses three-valued-logic. The following table describes the three-valued-logic expressions for `AND`, `OR`, and `NOT`.

Table 9–23: Three-valued-logic expression for AND (result of A AND B)

A	B		
	True	False	Unknown
`True`	`True`	`False`	`Unknown`
`False`	`False`	`False`	`False`
`Unknown`	`Unknown`	`False`	`Unknown`

Table 9–24: Three-valued-logic expression for OR (result of A OR B)

A	B		
	True	False	Unknown
`True`	`True`	`True`	`True`
`False`	`True`	`False`	`Unknown`
`Unknown`	`True`	`Unknown`	`Unknown`

Table 9–25: Three-valued-logic expression for NOT (result of NOT A)

A	Result of NOT A
`True`	`False`
`False`	`True`
`Unknown`	`Unknown`

(2) Notes on using JPQL in HiRDB

- You cannot specify location parameters and named parameters in the arguments of the following JPQL functions: `TRIM`, `SQRT`, `ABS`, `LENGTH`, `LOWER`, `MOD`, `LOCATE`, `UPPER`, `CONCAT`, `SUBSTRING`, `IS [NOT] NULL`

  If location parameters and named parameters are specified in the arguments of these functions, the operations might not function properly. Note that if the location parameters and named parameters are specified, a HiRDB SQL syntax error occurs and the `PersistenceException` exception containing the `SQLException` exception might be thrown.

  If you want to use the functionality of these functions, use the native query.
- You cannot specify the question mark (`?`) parameter on both sides of the four arithmetic operators (`+`, `-`, `*`, `/`) and both sides of the comparison operators (=, >, >=, <, <=, <>). Also, you cannot specify the question mark (`?`) parameter on one side of the comparison operators and a literal on the other side. If such a value is specified, the operations might not function properly. If such a value is specified, a HiRDB SQL syntax error occurs and the `PersistenceException` exception containing the `SQLException` exception might be thrown.

  Instead of executing the four arithmetic operations and comparison operations in JPQL, execute the four arithmetic operations and comparison operations before using JPQL and then use JPQL.

The examples of coding queries that cannot be used when JPQL is used in HiRDB are as follows:

Example 1 of query that cannot be used: Specifying a location parameter in the function argument

```
Query query1 = em.createQuery(
"SELECT  o FROM TestEntity o "+
                "WHERE o.name=TRIM(LEADING FROM ?1)")
              .setParameter(1, "  HitachiTaro");
```

Example 2 of query that cannot be used: Specifying a location parameter in the four arithmetic operators

```
int no_A=2;
int no_B=4;
Query query2 = em.createQuery(
"SELECT  o FROM TestEntity o WHERE o.id = ?1 + ?2")
               .setParameter(1, no_A)
               .setParameter(2, no_B);
```

Example 3 of query that cannot be used: Specifying a location parameter in the comparison operators

```
int cmp_no=3;
Query query3 = em.createQuery(
"SELECT  o FROM TestEntity o WHERE o.id = ?1 AND ?1 < 9")
               .setParameter(1, cmp_no);
```

9.17.10 Exceptions thrown when queries are used

In Cosminexus JPA Provider, if there is a syntactic error in a query-related annotation, an exception occurs when the application is deployed. Also, if the arguments of the query-related method are invalid and if the specified string is not a valid JPQL string, the `IllegalArgumentException` exception occurs or the execution of the query fails.

If the query is not valid for the database query that uses a native query or if the defined result set is not compatible with the query result, the execution of the query fails. In Cosminexus JPA Provider, the `PersistenceException` exception is thrown during the execution of the query.

(1) Exceptions thrown in the API functions of the query-related interfaces in EntityManager

The exceptions thrown in the API functions of the query-related interfaces in `EntityManager` are as follows:

- If the query string of the `createQuery` method is incorrect, the `IllegalArgumentException` exception is thrown.
- If the query is not defined with the name specified in the `createNamedQuery` method, the `IllegalArgumentException` exception is thrown.

(2) Exceptions thrown in the API functions of the Query interface

The exceptions thrown in the API functions of the Query interface are as follows:

- If the JPQL `UPDATE` statement or `DELETE` statement is invoked with the `getResultList` method, the `IllegalStateException` exception is thrown.
- If the query result does not exist, the `NoResultException` exception is thrown in the `getSingleResult` method. Note that if there are two or more query results, the `NonUniqueResultException` exception is thrown. If the JPQL `UPDATE` statement or `DELETE` statement is invoked, the `IllegalStateException` exception is thrown.
- If the JPQL `SELECT` statement is invoked with the `executeUpdate` method, the `IllegalStateException` exception is thrown. At this time, if the transaction does not exist, the `TransactionRequiredException` exception is thrown.
- If the second argument of the `setHint` method is not correct for implementation, the `IllegalArgumentException` exception is thrown.
- If the position of the arguments for the `setParameter` method does not match with the location parameter of the query, or if the parameter name of the argument does not match with the parameter of the query string, the `IllegalArgumentException` exception is thrown.
- If the arguments of the `setMaxResults` or `setFirstResult` method are negative, the `IllegalArgumentException` exception is thrown.

For details on the API functions, see the *Java documentation*.

9.18 Defining persistence.xml

This section describes the definition for the cache functionality of the entity objects, which is a Cosminexus JPA Provider-specific functionality, and the notes on data source specification for defining `persistence.xml`.

9.18.1 Defining the cache functionality of the entity objects

You define the cache functionality of the entity objects provided with Cosminexus JPA Provider in the `<property>` tag of `persistence.xml`. The following table describes the definition of the cache functionality of the entity objects in `persistence.xml`.

Table 9–26: Definition of the cache functionality of the entity objects in persistence.xml

Specified properties	Settings
`cosminexus.jpa.cache.size.`*ENTITY*	Specify the cache size for caching the entity.
`cosminexus.jpa.cache.size.default`	Specify the default cache size for caching the entity.
`cosminexus.jpa.cache.type.`*ENTITY*	Specify the cache type of the entity.
`cosminexus.jpa.cache.type.default`	Specify the default cache type of the entity.
`cosminexus.jpa.target-database`	Specify the name of the database to be connected to.

For details on tags, see *13.2.2 Cosminexus JPA Provider-specific properties that can be specified in the <property> tag*.

Reference note

The properties described here are properties unique to Cosminexus JPA Provider. In `persistence.xml`, you can specify other properties defined in the JPA specifications. However, in Cosminexus JPA Provider, you cannot use properties beginning with `javax` that are defined in the JPA specifications.

9.18.2 Notes on data source specification

With the data source specification in `persistence.xml`, you can use the user-specified Namespace functionality, which is an Application Server functionality, to assign an optional name to the resource adapter. If you set an optional name for the resource adapter in the `persistence.xml` settings, the optional name of the resource adapter must also be defined in the HITACHI Connector Property file. For details, see *9.19 Settings in the execution environment*.

9.19 Settings in the execution environment

To use Cosminexus JPA Provider, you must specify the J2EE server settings and the DB Connector settings.

9.19.1 J2EE server settings

You implement the J2EE server settings with the Easy Setup definition file. Specify the settings for the log files output by Cosminexus JPA Provider in the `<configuration>` tag of the logical J2EE server (`j2ee-server`) in the Easy Setup definition file.

The following table describes the settings in the Easy Setup definition file for the log files output by Cosminexus JPA Provider.

Table 9–27: Settings in the Easy Setup definition file for the log files output by Cosminexus JPA Provider

Specified parameters	Settings
`ejbserver.logger.channels.define.JPAOperationLogFile.filenum`	Specify the number of operation log files.
`ejbserver.logger.channels.define.JPAMaintenanceLogFile.filenum`	Specify the number of maintenance log files.
`ejbserver.logger.channels.define.JPAOperationLogFile.filesize`	Specify the size of the operation log.
`ejbserver.logger.channels.define.JPAMaintenanceLogFile.filesize`	Specify the size of the maintenance log.
`cosminexus.jpa.logging.level.operation.`*category*	Specify the log level of the operation log.

For details on the Easy Setup definition file and parameters to specify, see *12.2.1 Parameters used for setting up the user properties for the J2EE server*.

9.19.2 DB Connector settings

You implement the DB Connector settings with the server management commands and the HITACHI Connector Property file.

The following table describes the settings in the HITACHI Connector Property file.

Table 9–28: Settings in the HITACHI Connector Property file

Specified tags	Settings
`<resource-external-property>-<optional-name>` tag	If the optional name of the resource adapter is used in the data source specification of `persistence.xml`, the optional name of the resource adapter is set up. If the optional name of the resource adapter is not used in `persistence.xml`, the settings need not be specified.
`<resource-external-property>-<res-auth>` tag	If the optional name of the resource adapter is used in the data source specification of `persistence.xml`, make sure you specify `Container` as the resource authentication method.

Specified tags	Settings
`<resource-external-property>-<res-sharing-scope>` tag	If the optional name of the resource adapter is used in the data source specification of `persistence.xml`, make sure you specify `Shareable` to specify whether the resource connections will be shared.
`<property>-<property-name>` tag `<property>-<property-value>` tag	Specify the information for connecting to the database. • Specifying the user name Specify `User` in the `<property-name>` tag and the user name to be used for the database connection in the `<property-value>` tag. • Specifying the password Specify `Password` in the `<property-name>` tag and the password to be used for the database connection in the `<property-value>` tag.
`<connection-definition>-<transaction-support>` tag	Specify the transaction support level. Match the value specified here with the value specified in the `transaction-type` attribute of the `<persistence-unit>` tag in `persistence.xml`.

For details on the HITACHI Connector Property file and the specified tags, see *20.6 HITACHI Connector Property file*.

9.20 Precautions on application development

9.20.1 Precautions when using LAZY fetch in @OneToOne and @ManyToOne in Cosminexus 09-60 or a later environment

When developing an application in Cosminexus 09-60 or a later environment, create an entity class that meets the following conditions as the Java SE 6 source file. Also, specify settings so that Java SE 6 class files are generated when the application is compiled.

- The entity class includes `@OneToOne` or `@ManyToOne`.
- `FetchType.LAZY` is specified for the fetch attribute of `@OneToOne` or `@ManyToOne`.

Note:

When compiling by using the `javac` command, you can generate Java SE 6 class files by using `-target` and `-source`.

9.21 If a problem occurs in the JPA Application

This section describes the analysis procedure when a trouble occurs in an application using JPA.

This section describes an example of the model failure and the analysis procedure for the cause.

Note that the information, between each log output by Cosminexus JPA provider, is based on time, Thread ID/Process ID, and PersistenceUnit name. You can raise a log level and acquire the data, when you cannot specify the cause in the acquired log and the log is repeated.

9.21.1 Exception occurrence by user application

This point describes an analysis procedure implemented by a user when an exception occurs in a user application. The timing, when the exception occurs, might be when starting an application (including deploy) or when executing the application. This section describes the analysis procedure when an exception occurs in their respective timings.

(1) Exception occurrence when starting an application

The analysis procedure when an exception occurs while starting an application is as follows:

1. Checking message logs

   Reference the message output by a message log. See the message manual and confirm the handling procedure of the message ID that is output. Moreover, examine the countermeasures.

2. Checking application definitions

   Review the contents of the definition information of annotation in the application, `persistence.xml`, settings of O/R mapping files based on the contents of the message log and check whether there is any problem in the setup contents. When the method of handling the message log involves contacting the maintenance personnel, and when the cause of situations, such as not having any problem in the application process is not revealed, contact maintenance personnel.

(2) Exception occurred when executing an application

This point describes an analysis procedure when an exception occurs while executing an application.

1. Checking message logs

   Check the event of a failure from the message output to a message log. For details on the messages, see the manual *uCosminexus Application Server Messages*.

2. Checking exception logs

   Specify the point where an exception has occurred from the stack trace output to the exception log. To find out the cause of the exception occurrence, check whether there is any problem in the process contents of the application. Modify the application if there is any problem in the process.

   When the method of handling the message log involves contacting the maintenance personnel, and when the causes of situations, such as not having any problem in the application process are not revealed, contact maintenance personnel.

9.21.2 Errors occurred in a performance screen

This point describes an analysis procedure when a performance related failure occurs. For example, certain processes require time. The performance related problem might occur when starting an application or when executing the application.

When you start an application, if any performance related problem occurs, contact maintenance personnel. When you execute the application, and if any performance related problem occurs, specify the location of the problem as per the following procedure:

1. Checking PRF trace.

   Specify the location where the process takes time from the contents of a PRF trace. Specify whether the cause of a failure is in the user-implemented location or in the Cosminexus JPA Provider. Modify the application if there is any problem in the user-implemented location. When the processes in the internal process of Cosminexus JPA provider take time, contact maintenance personnel.

2. Checking operation logs

   For checking the issued SQL, check operation logs.

   For example, you can reference the issued SQL and confirm whether it is relevant to the following case or not:

   - `JOIN` operation is executed repeatedly by reading the entity because `JOINED` is used in inheritance strategy.
   - The entity managed by collection is operated, and therefore, `SELECT` is generated for each element with extension.

For details on the operation log output to Cosminexus JPA provider, see *16.1.1 Cosminexus JPA Provider operation log*.

9.21.3 Data used in troubleshooting

The following table shows the troubleshooting data and the acquisition source of the data that is required in the Cosminexus JPA Provider, when a failure occurs in the system.

Table 9–29: Troubleshooting data and data acquisition source required in Cosminexus JPA Provider

Troubleshooting data		Acquisition source	File name of the data
Message log		snapshot (primary)	Message log
Exception log		snapshot (primary)	Exception information when an error occurs
Operation log of Cosminexus JPA Provider		snapshot (primary)	Operation log of CJPA
J2EE server definition information		snapshot (primary)	• Option definition file for J2EE server • User property file for J2EE server • Security policy file for J2EE server
Definition information of applications included in the operation directory of J2EE server	`persistence.xml`	snapshot (secondary)	• Contents of EJB server operation directory • Contents of Web container operation directory
	O/R mapping file		
	Property file of EJB, WAR		
Connector property file of DB Connector		snapshot (secondary)	Contents of Web container operation directory

Troubleshooting data		Acquisition source	File name of the data
DB Connector log		snapshot (primary)	Operation log of the resource adaptor that is deployed and used as a J2EE resource adaptor
Trace based Performance Analysis		snapshot (primary)	PRF daemon and PRF command log
J2EE server thread dump		Thread dump acquisition command	
SQL trace of a database to be connected (However, only when the trace information is collected)	• HiRDB SQL trace	See the manual *HiRDB SQL Reference*	
	• For the Oracle Trace file that is acquired when setting `sql_trace=true` in SQL trace `init.ora`	See the manual of Oracle.	

9.22 Scope of support for the annotations included in the javax.persistence package

This section describes the scope of support for annotations included in the `javax.persistence` package. Annotation is a language specification that allows you to annotate source code.

The components in which the annotations of the `javax.persistence` package can be used differ based on the dependability on the JPA Provider. The annotations that depend on the JPA Provider and the annotations that do not depend on the JPA Provider are described separately:

9.22.1 Annotations that depend on the JPA Provider

This point describes the applicability of the annotations that depend on the JPA Provider. The annotations that can be coded in each component are as follows:

(1) WAR file (Servlet 3.0 compliant)

The following table lists the annotations that you can code in a WAR file:

Table 9–30: Annotations (javax. persistence package) that can be coded in WAR file (Servlet 3.0 compliant)

Annotation name	Servlet specifications						JSP specifications				Exception class	ManagedBean (JSF)	Other class
	Servlet	Servlet (API)	Servlet filter	Servlet filter (API)	Event listener	Event listener (API)	JSP FILE	Tag handler		Tag library event listener			
								Classic tag handler	Simple tag handler				
@PersistenceContext	Y	--	Y	--	Y	--	--	Y	Y	N	--	Y	--
@PersistenceContexts	Y	--	Y	--	Y	--	--	Y	Y	N	--	Y	--
@PersistenceProperty	Y	--	Y	--	Y	--	--	Y	Y	N	--	Y	--
@PersistenceUnit	Y	--	Y	--	Y	--	--	Y	Y	N	--	Y	--
@PersistenceUnits	Y	--	Y	--	Y	--	--	Y	Y	N	--	Y	--

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not supported by standard specifications.

(2) WAR file (Supported by Servlet 2.5)

The following table lists the annotations that you can code in a WAR file:

Table 9–31: Annotations (javax.persistence package) that can be coded in a WAR file (Supported by Servlet 2.5)

Annotation name	Servlet specifications			JSP specifications				Other class
	Servlet	Servlet filter	Event listener	JSP file	Tag handler		Tag library event listener	
					Classic tag handler	Simple tag handler		
`@PersistenceContext`	Y	Y	Y	--	Y	Y	N	--
`@PersistenceContexts`	Y	Y	Y	--	Y	Y	N	--
`@PersistenceProperty`	Y	Y	Y	--	Y	Y	N	--
`@PersistenceUnit`	Y	Y	Y	--	Y	Y	N	--
`@PersistenceUnits`	Y	Y	Y	--	Y	Y	N	--

Legend:
Y: Supported.
N: Not supported by Application Server.
--: Not applicable.

(3) EJB-JAR file (EJB3.1/EJB3.0 compliant)

The following table lists the annotations that you can code in an EJB-JAR file:

Table 9–32: Annotations (javax.persistence package) that can be coded in an EJB-JAR file (Supported by EJB3.0)

Annotation name	Enterprise Bean						Exception class	Other class
	Interface	Session Bean	Entity Bean	Message-driven Bean	Interceptor			
					Other than default Interceptor	Default Interceptor		
`@PersistenceContext`	--	Y	--	N	Y	Y	--	--
`@PersistenceContexts`	--	Y	--	N	Y	Y	--	--
`@PersistenceProperty`	--	Y	--	N	Y	Y	--	--
`@PersistenceUnit`	--	Y	--	N	Y	Y	--	--
`@PersistenceUnits`	--	Y	--	N	Y	Y	--	--

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not applicable.

(4) Library JAR file (Servlets or JSPs)

The following table lists the annotations that you can code in a servlet or JSP of a library JAR file.

Table 9–33: Annotations (javax.persistence package) that can be coded in a library JAR file (Servlets or JSPs)

Annotation name	Servlet specifications						JSP specifications			
	Servlet	Servlet (API)	Servlet filter	Servlet filter (API)	Event listener	Event listener (API)	JSP FILE	Tag handler		Tag library event listener
								Classic tag handler	Simple tag handler	
@PersistenceContext	--	--	Y	--	Y	--	--	Y	Y	N
@PersistenceContexts	--	--	Y	--	Y	--	--	Y	Y	N
@PersistenceProperty	--	--	Y	--	Y	--	--	Y	Y	N
@PersistenceUnit	--	--	Y	--	Y	--	--	Y	Y	N
@PersistenceUnits	--	--	Y	--	Y	--	--	Y	Y	N

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not applicable.

(5) Library JAR file (Enterprise Bean, exception class, or other classes)

The following table lists the annotations that you can code in the Enterprise Beans, exception classes, or the other classes of a library JAR file:

Table 9–34: Annotations (javax.persistence package) that can be coded in a library JAR file (Enterprise Beans, exception classes, or other classes)

Annotation name	Enterprise Bean					Exception class
	Interface	Session Bean	Entity Bean	Message-driven Bean	Interceptor	
@PersistenceContext	--	--	--	N	Y	--
@PersistenceContexts	--	--	--	N	Y	--
@PersistenceProperty	--	--	--	N	Y	--
@PersistenceUnit	--	--	--	N	Y	--
@PersistenceUnits	--	--	--	N	Y	--

Legend:

Y: Supported.

N: Not supported by Application Server.

--: Not applicable.

9.22.2 Annotations that do not depend on the JPA Provider

Irrespective of the file type, you can use the annotations that do not depend on the JPA Provider, in the entity class.

For details on the list of annotations included in the `javax.persistence` package, see *8.12 javax.persistence package*.

10 Web Container

This chapter describes the functionality that can be used only in V9 compatibility mode and precautions related to the Web container (server infrastructure for executing servlets and JSP). Use the Web container functionality when you execute a J2EE application with servlets or JSPs.

10.1 Functionality of filtering requests and responses

This section describes the functionality for filtering the requests and responses.

The following table describes the organization of this section.

Table 10–1: Organization of this section (Functionality for filtering requests and responses)

Category	Title	Reference
Description	Servlet filter provided by Application Server (built-in filter)	*10.1.1*
	Examples of recommended filter chain	*10.1.2*
Implementation	Definition in the DD	*10.1.3*
Settings	Execution environment settings (Web application settings)	*10.1.4*

Note:

There is no specific description of *Operations* and *Notes* for this functionality.

The filtering functionality available on Application Server is the functionality defined in the Servlet specifications and the functionality provided with Application Server. Both of the above functions filter requests and responses of servlets or JSPs.

The filtering functionality defined in the Servlet specifications wraps the requests before executing the servlets and JSPs or the responses after executing the servlets and JSPs. As a result, you can perform operations such as changing the data and acquiring the trace for the resources.

With the filtering functionality provided with Application Server, you can inherit the session information and compress the HTTP responses. In Application Server, a servlet filter is provided for using this filtering functionality. The servlet filter provided in Application Server is called a *built-in filter*. The following section explains the built-in filter provided in Application Server.

10.1.1 Servlet filter provided by Application Server (built-in filter)

In Application Server, a servlet filter (built-in filter) is provided to use the following functionality:

- Compressing the HTTP responses

  To compress the HTTP responses for the HTTP requests, Application Server provides an HTTP response compression filter as the built-in filter.

The following table describes the types of built-in filters. Further, references of the functionality that you can use by embedding the built-in filter in a Web application are also described.

Table 10–2: Types of built-in filters

Type of built-in filter	Description of functionality	Reference manual	Reference
HTTP response compression filter	This functionality compresses the HTTP responses to the HTTP requests for servlets, JSPs, and static contents, in the `gzip` format.	This manual	*10.2*

Note that with Servlet 3.0 or later, you can define a filter by using an API and not the `web.xml` file. However you cannot use an API to define a built-in filter.

The action of the built-in filter on HTTP requests and HTTP responses, the restrictions on the operation conditions of the built-in filter are explained below:

(1) Action on the HTTP requests and HTTP responses

The built-in filter acts on the request header and the request body of the HTTP requests sent from the client, and may delete, add, and change the information. In the same way, the built-in filter may also act on the response header and the response body of the HTTP responses sent from the server. The following table describes the action of the built-in filter on HTTP requests and HTTP responses:

Table 10–3: Action of the built-in filter on HTTP requests and HTTP responses

Type of built-in filter	Action on the HTTP request	Action on the HTTP response
HTTP response compression filter	--	When the response body is compressed, `gzip` is specified in the Content-Encoding header. The response body is compressed in `gzip` format.

Legend:
--: Not applicable

(2) Restriction on operation conditions

When you use the user filter and the built-in filter simultaneously, there are restrictions on the order in which the built-in filter is invoked in the filter chain.

The following restrictions are explained for each built-in filter:

- Restriction on the location
  This is the restriction on the location (invocation order) of the built-in filter in the filter chain.
- Restriction on operation conditions
  This is a restriction on the operation conditions, such as the preconditions for operation of the built-in filter.
- Restriction on the other servlet filters deployed before and after the built-in filter
  This is a restriction with respect to servlet filters that are deployed before and after the built-in filter.

(a) Restrictions on an HTTP response compression filter

The following table describes the restrictions on the HTTP response compression filter:

Table 10–4: Restrictions on the HTTP response compression filter

Type of restriction	Description
Restriction on the location	--
Restriction on operation conditions	--
Restriction on the other servlet filters deployed before and after the built-in filter	In the case of concurrent use of the HTTP response compression filter and servlet filter that changes the settings of the Content-Length header or the Content-Encoding header by using the `setHeader` method, `addHeader` method, `setIntHeader` method, or `addIntHeader` method of the `javax.servlet.http.HttpServletResponse` interface, you need to deploy the servlet filter after the HTTP response compression filter.

Legend:
--: Not applicable

10.1.2 Examples of recommended filter chain

The examples of recommended filter chain are explained below. Deploy the filters to form a chain in the following order:

Note that the general filters that act on the request body and the response body are assumed as user filters explained in the examples.

- **When using an HTTP response compression filter and a user filter (Filter A)**
    1. HTTP response compression filter
    2. User filter (Filter A)
- **When using an HTTP response compression filter and a user filter (Filter D)**
    1. HTTP response compression filter
    2. User filter (Filter D)

10.1.3 Definition in the DD

Define the functionality for filtering requests and responses in `web.xml`.

For details on definitions of the filtering functionality for requests and responses in DD, see the following:

- For details on the HTTP response compression filter, see *10.2 HTTP response compression functionality*.

10.1.4 Execution environment settings (Web application settings)

To define the functionality for filtering requests and responses you must set up the Web applications. Reference this section only if you want to set or change the properties of Web applications that do not contain `cosminexus.xml`.

Implement the Web application settings in the execution environment by using the server management commands and property files. To define the filtering for requests and responses, use the filter property file and WAR property file.

The tags specified in the filter property file and WAR property file correspond to the DD. For the definitions in the DD (`web.xml`), see *10.1.3 Definition in the DD*.

10.2 HTTP response compression functionality

This section describes the HTTP response compression functionality.

The HTTP response compression functionality compresses the HTTP responses for the HTTP requests to the servlets, JSPs, and static contents in the `gzip` format. By using this functionality to compress the HTTP responses, you can reduce the time required for communicating the HTTP responses between the Web container and Web client (browser).

This functionality is provided as a servlet filter that is embedded and runs in the Web application. This is called the *HTTP response compression filter*.

The following table describes the organization of this section.

Table 10–5: Organization of this section (HTTP response compression functionality)

Category	Title	Reference
Description	Overview of HTTP response compression filter	*10.2.1*
	Conditions for using the HTTP response compression filter	*10.2.2*
Implementation	Executing the applications that use the HTTP response compression filter	*10.2.3*
	Definition in the DD	*10.2.4*
	Examples of the DD definitions	*10.2.5*
Settings	Execution environment settings (Web application settings)	*10.2.6*

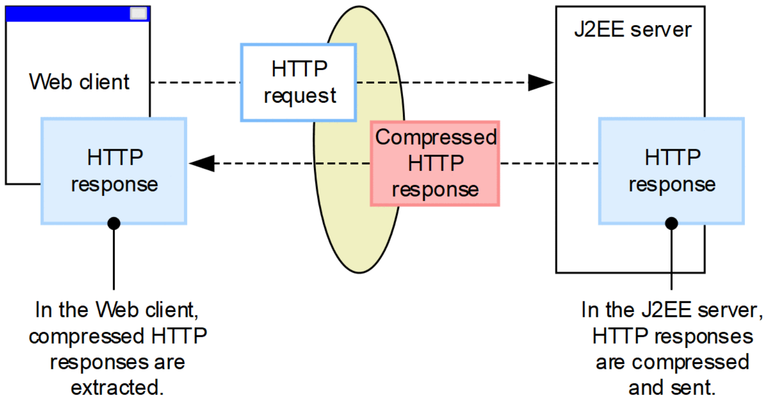
Note:

There is no specific description of *Operations* and *Notes* for this functionality.

10.2.1 Overview of HTTP response compression filter

If you enable the HTTP response compression functionality, the response body of the HTTP response is compressed in a `gzip` format. The following figure shows an overview of HTTP response compression functionality.

Figure 10–1: Overview of HTTP response compression functionality


Web client
HTTP request
J2EE server
HTTP response
Compressed HTTP response
HTTP response
In the Web client, compressed HTTP responses are extracted.
In the J2EE server, HTTP responses are compressed and sent.

To enable the HTTP response compression functionality, you need to embed the HTTP response compression filter provided by Application Server, in the Web application. In the case of applying the HTTP response compression functionality, add the filter definition of the HTTP response compression filter and the definition of the filter mapping to the DD (`web.xml`) of Web application. In the case of applying the HTTP response compression functionality to a Web application that is already deployed on a J2EE server, use the server management commands to add the filter definition of the HTTP response compression filter and the definition of filter mapping.

10.2.2 Conditions for using the HTTP response compression filter

This section describes the conditions and precautions for using the HTTP response compression filter.

(1) Preconditions

To use the HTTP response compression functionality, the following preconditions need to be satisfied:

- **gzip format compliant Web client**
  When the HTTP response compression functionality is enabled, you need to decompress HTTP responses that are compressed in `gzip` format with the Web client. The Web client, therefore must support the `gzip` format. If the Web client does not support the `gzip` format, HTTP responses are not compressed, even if the HTTP response compression functionality is enabled.
- **HTTP/1.1 compliant Web client**
  In HTTP response compression functionality, the value specified in the Accept-Encoding header of the HTTP request determines whether the Web client supports the `gzip` format. The Web client is therefore required to support HTTP/1.1 as defined in the specifications of the Accept-Encoding header.

(2) Required memory size

The memory required for the HTTP response compression functionality is obtained with the following formula:

Memory-required-for-the-HTTP-response-compression-functionality (bytes) = *Number-of-concurrent-executions-of-the-HTTP-requests-that-enable-the-HTTP-response-compression-functionality* × *Response-compression-threshold* (bytes)

The compression threshold is used for determining whether to compress the HTTP response depending on the size of the HTTP response body. Only when the size of the HTTP response body exceeds the size defined in the compression threshold, the HTTP response will be compressed. Note that the compression threshold is specified for HTTP requests.

Define the compression threshold in the DD (`web.xml`). When the size of the HTTP response is small, by defining the compression threshold you ensure that the time required for HTTP response compression is not longer than the time required for communication.

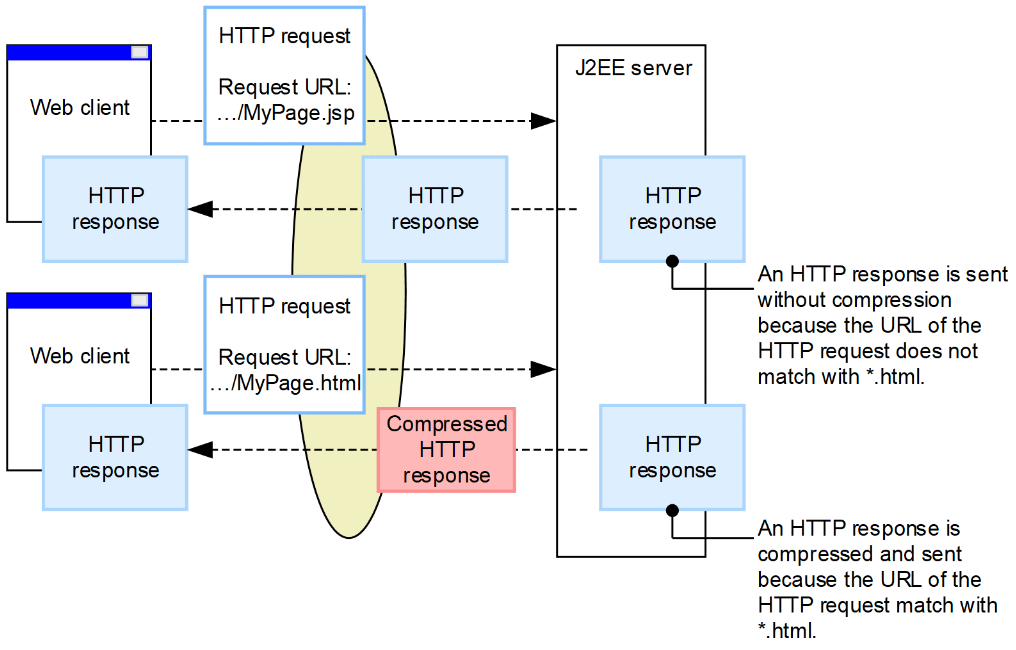
Decide an appropriate size for the compression threshold depending on the type of resources you want to compress and the speed of communications line. Hitachi recommends that you acquire the size defined in the compression threshold using the actual measurements and define the appropriate size.

(3) Conditions for enabling the HTTP response compression functionality

You can specify conditions for enabling the HTTP response compression functionality. The conditions that can be specified are explained below.

- **URL pattern of HTTP request**
  If the URL of the request to a Web application (in which the HTTP response compression filter is installed) matches with the specified URL pattern, compress the response to that request.
  The following figure shows an example with `*.html` specified as the URL pattern of the HTTP request that executes the compression of HTTP response:

Figure 10–2: Example with '*.html' specified as the UTL pattern of the HTTP request that executes HTTP response compression


Web client
HTTP request
Request URL: .../MyPage.jsp
J2EE server
HTTP response
HTTP response
HTTP response
An HTTP response is sent without compression because the URL of the HTTP request does not match with *.html.
Web client
HTTP request
Request URL: .../MyPage.html
Compressed HTTP response
HTTP response
HTTP response
An HTTP response is compressed and sent because the URL of the HTTP request match with *.html.

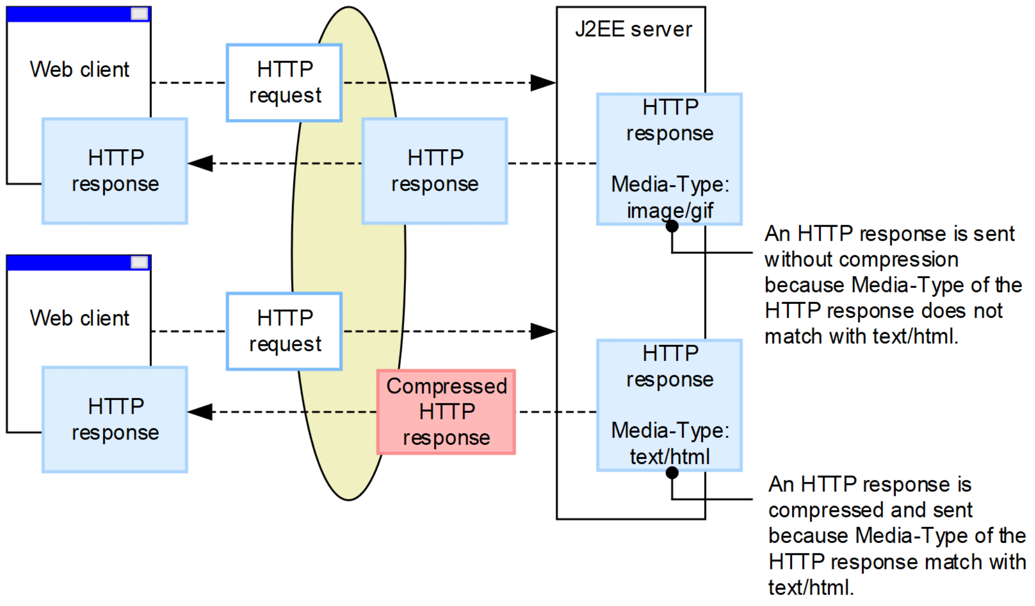
- **Media-Type of HTTP response**

  If the value of the Media-Type included in the Content-Type header of the HTTP response matches with the specified value, compress the HTTP response.

  In the case of servlets or JSPs, the value of the Media-Type of HTTP response is set by the `setContentType` method of a J2EE application. In the case of static contents, the Media-Type is the MIME type associated with the extension.

  The following figure shows an example with 'text/html' specified as the Media-Type of HTTP response that executes the compression of HTTP response:

Figure 10–3: Example with 'text/html' specified as the Media-Type of HTTP response that executes HTTP response compression

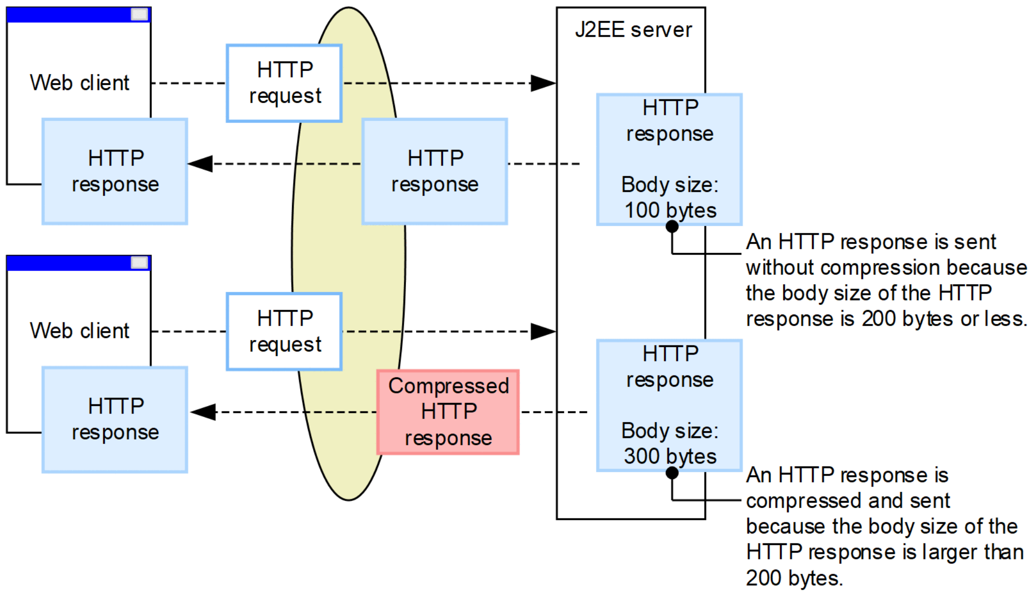

Web client
HTTP request
J2EE server
HTTP response
HTTP response
HTTP response
Media-Type: image/gif
An HTTP response is sent without compression because Media-Type of the HTTP response does not match with text/html.
Web client
HTTP request
Compressed HTTP response
HTTP response
HTTP response
Media-Type: text/html
An HTTP response is compressed and sent because Media-Type of the HTTP response match with text/html.

- **Body size of HTTP response**

Set a threshold value to execute the compression of an HTTP response. If the body size exceeds this threshold value, compress the HTTP response.

The following figure shows an example in which the HTTP response compression functionality is enabled by specifying '200 bytes' as the body size of the HTTP response that executes the compression of HTTP response:

Figure 10–4: Example with '200 bytes' specified as the body size of the HTTP response that executes HTTP response compression


Web client
HTTP request
J2EE server
HTTP response
HTTP response
HTTP response
Body size: 100 bytes
An HTTP response is sent without compression because the body size of the HTTP response is 200 bytes or less.
Web client
HTTP request
HTTP response
Compressed HTTP response
HTTP response
Body size: 300 bytes
An HTTP response is compressed and sent because the body size of the HTTP response is larger than 200 bytes.

(4) Notes

Precautions related to the definition of the HTTP response compression filter

When using the HTTP response compression filter, after considering the action of the built-in filter on the HTTP requests and HTTP responses and the restrictions on the order of the filter chain, you need to embed HTTP response compression filter in a Web application. For details on the built-in filter, see *10.1.1 Servlet filter provided by Application Server (built-in filter)*.

Note that with Servlet 3.0 or later, you can define a filter by using an API and not the `web.xml` file. However, you cannot use an API to define a built-in filter.

Precautions related to error pages

In the Web applications that use the HTTP response compression functionality, you can customize the error pages by using the following functionality:

- Error page customization with the Web server functionality
- Error page customization based on in-process HTTP server
- Error page customization with the `<error-page>` tag of `web.xml`

When using the error page with the `<error-page>` tag of `web.xml`, specify the servlet that acquires and uses `javax.servlet.ServletOutputStream` from the static contents or response in the error page.

In the HTTP response compression functionality, you use `javax.servlet.ServletOutputStream` acquired from the response object to output the compressed data. Therefore, `java.io.PrintWriter` cannot be acquired from the response object in the servlet or JSP that generates an error page.

10.2.3 Executing the applications that use the HTTP response compression filter

This section describes the precautions to be taken when developing applications that use the HTTP response compression filter.

(1) Order of invocation when the HTTP response compression filter is combined with other filters

The HTTP response compression filter must be invoked before the other filters specified in the HTTP response header. When you use the `setHeader` method, `addHeader` method, `setIntHeader` method, and `addIntHeader` method of `javax.servlet.http.HttpServletResponse` to use other filters that set Content-Length header and Content-Encoding header, deploy the other filters after the HTTP response compression filter.

(2) Precautions related to HTTP response buffer

When the HTTP response compression functionality is enabled, the buffer with the size specified in the compression threshold is installed before the HTTP response buffer. Data is written in the HTTP response buffer when the output data exceeds the compression threshold.

Unless the compressed data size exceeds the HTTP response buffer size, the HTTP response is not written in the Web client. If you are required to write the HTTP response in the Web client before the output data size exceeds the compression threshold, you must explicitly invoke the `flush` method of the stream for response output#. However, if the `flush` method or `flushBuffer` method of the `javax.servlet.ServletResponse` interface is invoked before the output data size exceeds the compression threshold, the output data is written in the Web client without being compressed.

#

The stream for response output indicates the following objects:

- `javax.servlet.ServletOutputStream` acquired by the `getOutputStream` method of the `javax.servlet.ServletResponse` interface
- `java.io.PrintWriter` acquired by the `getWriter` method of the `javax.servlet.ServletResponse` interface
- `javax.servlet.jsp.JspWriter` implicitly available in JSP

(3) Precautions related to the response header of the HTTP response

When the response body of the HTTP response is compressed with the HTTP response compression functionality, `gzip` is specified in the Content-Encoding header and Accept-Encoding is specified in the Vary header of this HTTP response. Nothing is specified in the Content-Length header.

Therefore, note the following points when you use the `setContentLength` method of the `javax.servlet.ServletResponse` interface and when you use API# for adding and changing the response header of the `javax.servlet.http.HttpServletResponse` interface:

- When you use one of the following APIs to add the filter for setting the Content-Length header and Content-Encoding header, define in the DD (`web.xml`) that the API be executed after the filter for response compression:
    - `setContentLength` method of the `ServletResponse` class
    - API# for adding and changing the response header of the `HttpServletResponse` class

- When the response body of the HTTP response is compressed, the Content-Length header of the HTTP response is not added even though the `setContentLength` method of the `ServletResponse` class and the API# for adding and changing the response header of the `HttpServletResponse` class are used. The HTTP response for which the Content-Length header is not added is sent in the chunk format to the client by the Web container.
- When the response body of the HTTP response is compressed, a value is not set in the Content-Encoding header even though the API# for adding and changing the response header of the `HttpServletResponse` class is used. When the response body is compressed, `gzip` is specified in the Content-Encoding header by the Web Container.

#

The API for adding and changing the response header indicates the following methods of the `javax.servlet.http.HttpServletResponse` interface:

- `setHeader` method
- `addHeader` method
- `setIntHeader` method
- `addIntHeader` method

(4) Precautions related to data output for HTTP response

Note the following points when `ServletOutputStream` or `PrintWriter` is acquired with the `getOutputStream` method or the `getWriter` method of the `javax.servlet.ServletResponse` interface and the HTTP response is output:

- When you invoke the `setContentType` method of `ServletResponse` while the data is being written in the HTTP response buffer by using `ServletOutputStream` or `PrintWriter`, even if HTTP response with Media-Type specified for compression exists, the HTTP response is not compressed.
  However, if an asterisk (*) is specified in the Media-Type to be compressed, the HTTP response is compressed.
- To compress the JSP output by specifying Media-Type, either specify the `contentType` attribute of the `Page` directive or invoke the `setContentType` method of the `ServletResponse` class before the JSP buffer is exceeded.

(5) Precautions for compressing HTTP responses in applications

For HTTP responses compressed in applications, specify settings so that the HTTP response compression functionality is not enabled. If the HTTP response compression functionality is enabled for the HTTP responses compressed in applications, the operations might not function properly.

10.2.4 Definition in the DD

This section describes the DD definitions required for using the HTTP response compression functionality.

To enable the HTTP response compression functionality, you must add the filter definition and filter mapping definition in the DD of the Web applications. The HTTP response compression functionality is enabled only when requests exist for the resources with the URL pattern mapped by the filter mapping definition.

Define the HTTP response compression functionality in `web.xml`.

The following table lists the definition of HTTP response compression functionality in the DD:

Table 10–6: Definition of HTTP response compression functionality in the DD

Settings	Specified tags	Setting contents
Filter definition	`<filter-name>` tag in the `<filter>` tag	Specify the name of the filter you want to add. The set value is a fixed value. **Set value (fixed value)** `com.hitachi.software.was.web.ResponseCompressionFilter`
	`<filter-class>` tag in the `<filter>` tag	Specify the class name of the filter you want to add. The set value is a fixed value. **Set value (fixed value)** `com.hitachi.software.was.web.ResponseCompressionFilter`
Mapping of the URL pattern and HTTP response compression rules	`<param-name>` tag and `<param-value>` tag in the `<filter-class><init-param>` tag	Specify the mapping of the URL pattern and the HTTP response compression functionality. For details, see *10.2.4(1) Mapping of the URL pattern and HTTP response compression rules (url-mapping)*.
HTTP response compression rules		Specify the compression rule name, Media-Type, and compression threshold as the compression rules for HTTP response. For details, see *10.2.4(2) HTTP response compression rules*.
Filter mapping definition	`<filter-name>` tag in the `<filter-mapping>` tag	Specify the filter name. The set value is a fixed value. **Set value (fixed value)** `com.hitachi.software.was.web.ResponseCompressionFilter`
	`<url-pattern>` tag in the `<filter-mapping>` tag	Specify the mapping of the URL pattern or servlet class and the HTTP response compression filter. The HTTP response compression functionality is enabled only when requests exist for the resources with the URL pattern mapped by the filter mapping definition. The set value is optional.

The definition contents of the DD are described in the following examples of the DD definitions:

```
 ...
<filter>
   <filter-name>com.hitachi.software.was.web.ResponseCompressionFilter</filt
er-name>
   <filter-class>com.hitachi.software.was.web.ResponseCompressionFilter</fil
ter-class>
   <init-param>
     <param-name>url-mapping</param-name>
     <param-value>
       /*=rule1;
     </param-value>
    </init-param>
    <init-param>
       <param-name>rule1</param-name>
     <param-value>
        *: 1000;
     </param-value>
   </init-param>
</filter>
    ...
<!-- The filter mappings for Response Compression Filter -->
<filter-mapping>
   <filter-name>com.hitachi.software.was.web.ResponseCompressionFilter</filt
```

```
er-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>
```

The part enclosed in the `<filter>` tag is the filter definition and the part enclosed in the `<filter-mapping>` tag is the filter mapping definition. The value specified in the `<filter-name>` tag and `<filter-class>` tag of the filter definition is fixed in the following package name:

```
com.hitachi.software.was.web.ResponseCompressionFilter
```

The contents defined in the `<init-param>` tag of the DD are as follows:

(1) Mapping of the URL pattern and HTTP response compression rules (url-mapping)

The `<init-param>` tag of the DD specifies the mapping of the URL pattern that enables the HTTP response compression functionality and the name of the HTTP response compression rule applied to the specified URL pattern.

The rules for specifying parameters and the mapping rules for URL pattern are as follows:

(a) Rules for specifying parameters

- The URL pattern is case-sensitive.
- The name of the HTTP response compression rule is case-sensitive.
- The URL pattern and HTTP response compression rule name are delimited by one-byte equal sign (=).
- Delimit multiple specifications with one-byte semicolon (`;`).
- If multiple URL patterns are applicable, the URL pattern specified earlier is used.
- The linefeed, tabs, and spaces in the parameter value are ignored.
- The one-byte semicolon (`;`) at the end of the parameter value is ignored.

(b) Mapping rules for URL pattern

The path match, extension match, and exact match mapping rules are applicable to the URL pattern defined in the `url-mapping` parameter. The following shows the mapping rules:

- **Path match**

  If a string beginning with `/` and ending with `/*` is specified as the URL pattern and if the relative path from the context root of the request URL begins with the string excluding `*` from the URL pattern, the string is considered as matching. Also, if `/*` is specified as the URL pattern, all the request URLs are considered to be matching.

  Examples:

  In the URL pattern `/jsp/*`, if the relative path from the context root of the request URL is `/jsp/index.jsp`, the string is considered as matching.

- **Extension match**

  If a string beginning with `*.` is specified as the URL pattern and if the string is the same as the string continuing after `*.` of the request URL extension and URL pattern, the string is considered as matching.

  Example:

  In the URL pattern `*.jsp`, if the relative path from the context root of the request URL is `/jsp/index.jsp`, the string is considered to be matching.

- **Exact match**

  If a string other than those mentioned above begins with `/` as the specified URL pattern and if the relative path from the context root of the request URL is exactly same as this URL pattern, the string is considered as matching.

  Example:

  In the URL pattern `/jsp/index.jsp`, if the relative path from the context root of the request URL is `/jsp/index.jsp`, the string is considered as matching.

(2) HTTP response compression rules

The `<init-param>` tag of the DD specifies the Media-Type of the HTTP response to be compressed and the compression threshold.

Media-Type

If the Media-Type specified as the condition is included in the HTTP response, the HTTP response will be compressed with the HTTP response compression functionality.

Compression threshold

The compression threshold is specified as an integer value from 100 to 2,147,483,647. By default, compression threshold `100` is applied to the all Media-Type.

The compression threshold is used for determining whether to compress the HTTP response depending on the size of the HTTP response body. When the size of the HTTP response body exceeds the size defined in the compression threshold, the HTTP response will be compressed with the HTTP response compression functionality.

When the size of the HTTP response is small, by defining the compression threshold you ensure that the time required for HTTP response compression is not longer than the time required for communication.

An appropriate size is decided for the compression threshold depending on the type of resources you want to compress and the speed of communications line; therefore, it is recommended that the size defined in the compression threshold be acquired using actual measurements and be defined appropriately.

The following shows the rules for specifying parameters:

- If an asterisk (`*`) is specified in Media-Type, all Media-Type are shown. However, if each Media-Type is specified, the specification for each Media-Type is given priority.
- Media-Type is case-insensitive.
- Media-Type and compression threshold are delimited with one-byte colon (`:`).
- Delimit multiple specifications with one-byte semicolon (`;`).
- If the same Media-Type is specified multiple times, the Media-Type specified later is used.
- The linefeed, tabs, and spaces in the parameter value are ignored.
- The one-byte semicolon (`;`) at the end of the parameter value is ignored.

(3) Notes

Take the following precautions when you set up the HTTP response compression filter:

- Check the validity of the filter initialization parameter when you initialize the filter for response compression. If a problem occurs in the value defined in the initialization parameter, an error occurs in the filter initialization process and the Web application will not start.
- If the request URL matches the URL pattern specified in the `url-mapping` parameter, but if the response compression filter does not match the mapped URL pattern, the response compression rules specified in the `url-mapping` parameter are not applied.

- If multiple URL patterns are mapped to the response compression filter, you need to specify the filter mapping definition in such a way so that the request URL does not match with multiple URL patterns simultaneously. To specify different response compression functionality for multiple URL patterns, specify multiple URL patterns in the `url-mapping` parameter.
- If the Web application version is Servlet 2.4 or later versions, do not specify the `<dispatcher>` tag of `<filter-mapping>`. Also, if `FORWARD`, `INCLUDE`, and `ERROR` is specified in the `<dispatcher>` tag, note that an error will occur while starting the Web application and the application cannot be started.

10.2.5 Examples of the DD definitions

This section describes the examples of the DD definitions that use the HTTP response compression functionality as the examples for each of the following cases:

- When the compression condition is specified for the body size of the HTTP response
- When the compression condition is specified for the URL pattern
- When the compression condition is specified for the Media-Type of the HTTP response
- When the compression conditions are combined and specified

(1) When the compression condition is specified for the body size of the HTTP response

The following is an example of definition when the compression condition is specified for the body size of the HTTP response:

```
<web-app>
   ...
<!-- The filter for Response Compression Filter -->
<filter>
   <filter-name>com.hitachi.software.was.web.ResponseCompressionFilter</filt
er-name>
   <filter-class>com.hitachi.software.was.web.ResponseCompressionFilter</fil
ter-class>
   <init-param>
     <param-name>url-mapping</param-name>
     <param-value>
        /*=rule1;
     </param-value>
     </init-param>
     <init-param>
       <param-name>rule1</param-name>
       <param-value>
         *:1000;
       </param-value>
     </init-param>
</filter>
   ...
<!-- The filter mappings for Response Compression Filter -->
<filter-mapping>
   <filter-name>com.hitachi.software.was.web.ResponseCompressionFilter</filt
er-name>
   <url-pattern>/*</url-pattern>
```

```
</filter-mapping>
   ...
</web-app>
```

In the definition example, the HTTP response for accessing the URL pattern `/*` and the body size exceeding 1,000 bytes is compressed.

(2) When the compression condition is specified for the URL pattern

The following is an example of definition when the compression condition is specified for the URL pattern:

```
<web-app>
   ...
<!-- The filter for Response Compression Filter -->
<filter>
   <filter-name>com.hitachi.software.was.web.ResponseCompressionFilter</filt
er-name>
   <filter-class>com.hitachi.software.was.web.ResponseCompressionFilter</fil
ter-class>
   <init-param>
     <param-name>url-mapping</param-name>
     <param-value>
       /app/dir/*=rule1;
       *.html=rule1;
     </param-value>
   </init-param>
   <init-param>
     <param-name>rule1</param-name>
     <param-value>
       *:100;
     </param-value>
   </init-param>
</filter>
   ...
<!-- The filter mappings for Response Compression Filter -->
<filter-mapping>
   <filter-name>com.hitachi.software.was.web.ResponseCompressionFilter</filt
er-name>
   <url-pattern>/app/*</url-pattern>
</filter-mapping>
   ...
</web-app>
```

In this definition example, the HTTP response for accessing the URL pattern `/app/*` and fulfilling the following conditions is compressed:

- HTTP response for which the HTTP request URL pattern is `/app/dir/*` and the body size exceeds 100 bytes
- HTTP response for which the HTTP request URL pattern is `*.html` and the body size exceeds 100 bytes

(3) When the compression condition is specified for the Media-Type of the HTTP response

The following is an example of definition when the compression condition is specified for the Media-Type of HTTP response:

```
<web-app>
   ...
<!-- The filter for Response Compression Filter -->
<filter>
   <filter-name>com.hitachi.software.was.web.ResponseCompressionFilter</filt
er-name>
   <filter-class>com.hitachi.software.was.web.ResponseCompressionFilter</fil
ter-class>
   <init-param>
     <param-name>url-mapping</param-name>
     <param-value>
       /*=rule1;
     </param-value>
   </init-param>
   <init-param>
     <param-name>rule1</param-name>
     <param-value>
       text/html:500;
       application/pdf:1000;
     </param-value>
   </init-param>
</filter>
   ...
<!-- The filter mappings for Response Compression Filter -->
<filter-mapping>
   <filter-name>com.hitachi.software.was.web.ResponseCompressionFilter</filt
er-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>
   ...
</web-app>
```

In this definition example, the HTTP response for accessing the URL pattern `/*` and fulfilling the following conditions is compressed:

- HTTP response for which Media-Type is text or html and the body size exceeds 500 bytes
- HTTP response for which Media-Type is application or pdf and the body size exceeds 1,000 bytes

(4) When the compression conditions are combined and specified

You can also combine and define the compression conditions as shown in the following example:

```
<web-app>
   ...
<!-- The filter for Response Compression Filter -->
<filter>
   <filter-name>com.hitachi.software.was.web.ResponseCompressionFilter</filt
er-name>
   <filter-class>com.hitachi.software.was.web.ResponseCompressionFilter</fil
ter-class>
   <init-param>
     <param-name>url-mapping</param-name>
     <param-value>
       /app/dir1/*=rule1;
       /app/dir2/*=rule2;
```

```
      *.html=rule3;
    </param-value>
  </init-param>
  <init-param>
    <param-name>rule1</param-name>
    <param-value>
      *:500;
      application/pdf:1000;
    </param-value>
  </init-param>
  <init-param>
    <param-name>rule2</param-name>
    <param-value>
      application/pdf:2000;
    </param-value>
  </init-param>
  <init-param>
    <param-name>rule3</param-name>
    <param-value>
      *:100;
    </param-value>
  </init-param>
</filter>
   ...
<!-- The filter mappings for Response Compression Filter -->
<filter-mapping>
   <filter-name>com.hitachi.software.was.web.ResponseCompressionFilter</filt
er-name>
   <url-pattern>/app/*</url-pattern>
</filter-mapping>
   ...
</web-app>
```

In this definition example, the HTTP response for accessing the URL pattern `/app/*` and fulfilling the following conditions is compressed:

- HTTP response for which the HTTP request URL pattern is `/app/dir1/*`, Media-Type is other than application or pdf, and the body size exceeds 500 bytes
- HTTP response for which the HTTP request URL pattern is `/app/dir1/*`, Media-Type is application or pdf, and the body size exceeds 1,000 bytes
- HTTP response for which the HTTP request URL pattern is `/app/dir2/*`, Media-Type is application or pdf, and the body size exceeds 2,000 bytes
- HTTP response for which the HTTP request URL pattern is `*.html` and the body size exceeds 100 bytes

10.2.6 Execution environment settings (Web application settings)

To use HTTP response compression based on filtering, you must set up the Web application.

Reference the Web application settings only to set or change the properties of Web applications that do not contain `cosminexus.xml`.

Implement the Web application settings in the execution environment using the server management commands and property files. To define HTTP response compression based on filtering, use the filter property file and WAR property file.

The tags specified in the filter property file and WAR property file correspond to the DD. For details on definitions in the DD (`web.xml`), see *10.2.4 Definition in the DD* and *10.2.5 Examples of the DD definitions*.

10.3 Precautions related to the Web container

The maximum size of POST data that can be handled by the Web container in V9 compatibility mode is 2 GB. Also, the maximum size of the POST data that can be handled is regulated depending on the settings of the Web container, and the settings of the Web server and redirector in front of the Web container.

10.4 Precautions for implementing servlets and JSPs

This section describes the precautions for implementing servlets and JSPs.

The following table describes the organization of this section.

Table 10–7: Organization of this section (Precautions for implementing servlets and JSPs)

Titles	Reference
Common precautions for implementing servlets and JSPs	*10.4.1*
Precautions related to the specifications that are added or changed in the EL2.2 specifications	*10.4.2*

10.4.1 Common precautions for implementing servlets and JSPs

This section describes the common precautions for implementing servlets and JSPs as the application programs running on the Application Server.

(1) J2EE application version requirements for Web application operations

For each Web application version, the following table lists the J2EE version to which the J2EE application conforms and that forms the prerequisite for Web application operations:

Table 10–8: J2EE version to which the J2EE application conforms

Version of J2EE specifications to which the J2EE application conforms	Servlet specifications corresponding to the Web applications				
	3.0	2.5	2.4	2.3	2.2
Java EE 6	Y	Y	Y	Y	Y
Java EE 5	N	Y	Y	Y	Y
J2EE1.4	N	N	Y	Y	Y
J2EE1.3	N	N	L	Y	Y
J2EE1.2	N	N	L	L	Y

Legend:

Y: Available

L: Limited (since the version of the J2EE specifications is updated to 1.4 when importing the J2EE application)

N: Not available

(2) Supported range of Web applications

The Web application version is identified by the version information of the Servlet specifications described in `web.xml`. A Web application of a higher version can use the functionality of a lower version. A Web application of a lower version cannot use the functionality of a higher version.

The following table describes the range of functionality available for each Web application version:

Table 10–9: Supported range of Web applications

Version of the Web application	Servlet					JSP					Tag library[#1]			
	3.0	2.5	2.4	2.3	2.2	2.2	2.1	2.0	1.2	1.1	2.1	2.0	1.2	1.1
3.0	Y	Y	Y	Y	Y	N[#2]	Y	Y	Y	Y	Y	Y	Y	Y
2.5	N	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y
2.4	N	N	Y	Y	Y	N	N	Y	Y	Y	N	Y	Y	Y
2.3	N	N	N	Y	Y	N	N	N	Y	Y	N	N	Y	Y
2.2	N	N	N	N	Y	N	N	N	N	Y	N	N	N	Y

Legend:

Y: Available

N: Not available

#1

The tag library version indicates the version of the tag library descriptor (TLD file).

#2

A `web.xml` compatible with JSP 2.2 can be read. However, any tag added to JSP 2.2 is ignored.

Note that if the functionality of a later version is used from a Web application of an earlier version, an error might occur. The following table describes the errors in each version:

Table 10–10: Errors that occur if functionalities of Servlet 3.0 are used from a Web application that is compatible with Servlet 2.2, 2.3, 2.4, or 2.5 specifications

Specifications	Functionality used	Operation in the event of an error
Servlet 3.0	Invoking new API	The program does not check whether an API of the Servlet 3.0 specification is being used. To avoid any abnormal operation, do not call the API.
	Using new annotations	For handling errors occurring due to annotations, see *14. Using Annotations* in the *uCosminexus Application Server Common Container Functionality Guide.*
JSP 2.2	New EL API	The program does not check whether an API of EL2.2 specification is being used. To avoid any abnormal operation, do not call the API.

For details on operation and precautions when upgrading the Web application version from the Servlet 2.2 specifications, Servlet 2.3 specifications, Servlet 2.4 specifications, or Servlet 2.5 specifications to the Servlet 3.0 specifications, see the manual *uCosminexus Application Server Web Container Functionality Guide*.

Table 10–11: Errors when Servlet 2.5 functionality is used from a Web application corresponding to Servlet 2.2, 2.3, or 2.4 specifications

Specifications	Functionality used	Error processing
Servlet 2.5	Invoking new API	The check whether the API added in the Servlet 2.5 specifications is used, is not performed. The operations when the API is invoked do not function properly, so do not invoke the API.
	Using new annotations	For the processing when an annotation is used and an error occurs, see *14. Using annotations* in the *uCosminexus Application Server Common Container Functionality Guide*.
JSP 2.1	Attributes of the new directive[#1]	KDJE39145-E message is output in the servlets log and KDJE39186-E message is output in the message log[#2] and a translation error occurs.

Specifications	Functionality used	Error processing
	TLD 2.1	If the following TLD file exists when the Web application starts, the KDJE39293-W message is output in the message log and the processing is not performed: • TLD file specified in the `<tablib-location>` element in the `<taglib>` element of `web.xml` • TLD file deployed under the `/META-INF` directory in the Jar file under the `/WEB-INF/lib` directory If a TLD file other than the above exists when the Web application starts, the KDJE39293-W message is output in the message log during JSP compilation and the processing is not performed. If JSP compilation occurs when the application is accessed the first time, the KDJE39145-E message is output in the servlets log and the KDJE39186-E message is output in the message log#2 and a translation error occurs.
	EL addition functionality	• Dedicated processing is not implemented for the `Enum` type added in JSP 2.1. The processing is same as for the general classes. However, if the EL API of the JSP 2.0 specifications that is deprecated in the JSP 2.1 specifications is used, regardless of the Web application version, the API is processed in the EL functionality range of the JSP 2.0 specifications. • EL with `#{}` format is displayed as a string. • `"\#"` is not handled as an escape sequence, and is displayed as a string `"\#"`.

#1

If an undefined directive is specified in the JSP specifications for XXX using the format `<jsp:directive.XXX/>` on the JSP page or if an undefined standard action is specified in the JSP specifications for XXX using the format `<jsp:XXX>`, the definition contents are output as is.

#2

The details of JSP compilation error are output in KDJE39145-E and the reporting of translation error is output in KDJE39186-E.

For details on operation and precautions when upgrading the Web application version from the Servlet 2.2 specifications, Servlet 2.3 specifications, or Servlet 2.4 specifications to the Servlet 2.5 specifications, see the manual *uCosminexus Application Server Web Container Functionality Guide*.

Table 10–12: Errors when using the functionality of Servlet 2.4 specifications from a Web application corresponding to Servlet 2.2 or 2.3 specifications

Specifications	Functionality used	Error processing
Servlet 2.4	Invoking new API	The check for whether the API added in the Servlet 2.4 specifications is used, is not performed. The operations when the API is invoked do not function properly, so do not invoke the API.
	Registering new listener	The KDJE39297-W message is output in the message log when the Web application starts and the listener definition is ignored.
JSP 2.0	New directive and new standard action#1	The KDJE39145-E message is output in the servlets log and the KDJE39186-E message is output in the message log#2, and a translation error occurs.
	Tag files	When TLD is not used The `tagdir` attribute that is a new attribute is assumed to be invalid in the `taglib` directive, the KDJE39145-E message is output in the servlets log and the KDJE39186-E message is output in the message log#2, and a translation error occurs. When TLD is used A TLD 2.0 usage error occurs.

Specifications	Functionality used	Error processing
	TLD 2.0	The following TLD files are checked when the Web application starts. When applicable, the KDJE39293-W message is output in the message log and the file is ignored: • TLD file specified in `<taglib><tablib-location>` of `web.xml` • TLD file deployed under `/META-INF` in the Jar file under `/WEB-INF/lib` The TLD files other than above-mentioned are checked during the JSP compilation. When the JSP file is compiled, such as during the initial access, the KDJE 39145-E message is output in the servlets log and the KDJE39186-E message is output in the message log#2, and a translation error occurs.
	Simple Tag Handler	The KDJE39145-E message is output in the servlets log and the KDJE39186-E message is output in the message log#2, and a translation error occurs.

#1

If an undefined directive is specified in the JSP specifications for XXX using the format `<jsp:directive.XXX/>` on the JSP page or if an undefined standard action is specified in the JSP specifications for XXX using the format `<jsp:XXX>`, the definition contents are output as it is.

#2

The details of JSP compilation error are output in KDJE39145-E and the reporting of translation error is output in KDJE39186-E.

For details on operation and precautions when upgrading the Web application version from the Servlet 2.2 specifications or Servlet 2.3 specifications to the Servlet 2.4 specifications, see the manual *uCosminexus Application Server Web Container Functionality Guide*.

Note that even if you use the Servlet 2.3 functionality from the Web applications corresponding to the Servlet 2.2 specifications, when imported, the application is rewritten by the Web application conforming to the Servlet 2.3 specifications, and therefore, the application is processed normally and error is not reported.

(3) Notes on using the transaction and JDBC connection

To use a transaction in the servlets and JSPs, acquire the JDBC connection with the applicable service method and release the connection before the applicable service method ends. In the servlets and JSPs where the transaction is running, the use of the following JDBC connections is not supported:

- Use of the JDBC connection on the thread generated by the service methods of the servlets and JSPs.
- Use of the JDBC connection in the service methods of the other servlets and JSPs invoked from the service methods of the servlets and JSPs.
- Use of the JDBC connection acquired with the `init` method of the service methods of the servlets and JSPs.
- Use of the JDBC connection stored in the instance variable.#

  #

  When the servlets and JSPs of `SingleThreadModel` are used, the JDBC connection can be stored in the instance variable.

(4) Notes related to package name specification

If a class with an invalid package name is used in the servlets and JSPs, an error with status code 500 occurs when the class is accessed from the browser. For example, even if a created class file is deployed correctly and accessed from the browser, if the package name declaration is invalid, the applicable class is not found. In this case, an error with status code 500 is returned.

(5) Notes for using cookies

- Do not use cookies containing double-byte codes such as Japanese characters. If such cookies are used, the HTTP session used in the servlets and JSPs might be lost.
- When managing a session with a cookie, the session generated using the servlets or JSPs accessed in the URL by the host name is not inherited in the servlets or JSPs accessed in the URL with the IP address specified instead of the host name (and vice versa).

(6) Notes related to display of input values with special meanings

When the input values of characters with special meanings such as "<" and ">" in forms are displayed as it is, malicious users might use the tags such as `<SCRIPT>`, `<OBJECT>`, `<APPLET>`, `<EMBED>` that can execute scripts and cause important security-related problems. A processing must be added for the application developer to inspect the data entered by a user and the characters with special meanings must be excluded.

(7) Notes related to error page display after response commit

After the response is committed in the servlets or JSPs, even if an error such as an exception occurs, the following error pages are not displayed in the browser:

- Error pages specified in `web.xml`
- Error pages specified in the `errorPage` attribute of the JSP page directive

The Web container commits the response automatically when the response buffer becomes full apart from the case in which the user commits the response by invoking the `flushBuffer` method of the `ServletResponse` class explicitly.

To check whether the response is committed in the servlets or JSPs, use the `isCommitted` method of the `ServletResponse` class. Also, you can change the buffer size using the `setBufferSize` method of the `ServletResponse` class in the case of servlets and by specifying the buffer attribute of the page directive for JSPs.

(8) Improving the performance when using the PrintWriter and JSPWriter class

By reducing the frequency of invoking the `print` method and the `println` method of the `PrintWriter` class and the `JSPWriter` class, you can reduce the access frequency and improve the performance. For example, use the `StringBuffer` class and invoke the `println` method finally to reduce the frequency of invoking the `print` and `println` methods.

(9) Notes on referencing the error information by javax.servlet.error.XXXXX

The `javax.servlet.error.XXXXX` attributes defined in the Servlet 2.3 specifications are used for referencing the error information that causes the execution of the error page in the servlets or JSPs specified in the `<error-page>` tag of `web.xml`. Do not reference these attributes from servlets or JSPs other than those specified in the `<error-page>` tag of `web.xml`.

(10) Notes on accessing files

To access a file, make sure you specify the absolute path. If you specify a relative path, the J2EE server tries to use the relative path to find the target path from the execution directory. If you specify the relative path in the `getRealPath` method of the `ServletContext` class, the relative path in the directory that deploys the WAR file is acquired.

Furthermore, when you access a file, make sure that you close the file. If you access a file in the WAR file deployment directory and do not close the file, you cannot perform the normal un-deployment on the J2EE server. If the file not closed even when the path under the WAR file deployment directory is not specified, events such as the files cannot be deleted when the J2EE server is running occur.

(11) Error page settings when an exception occurs

If an exception occurs while accessing the JSPs and servlets, the exception status code is returned to the browser by the default processing in the Web Container. To change this default processing, specify the JSP errorPage or set the error page in `web.xml`.

(12) Notes related to acquisition of the class loader

To use the following methods by acquiring the class loader of the Cosminexus Component Container from the code in the J2EE application, use the `java.net.JarURLConnection` class:

- `getResource(String).openConnection().getInputStream()`
- `getResource(String).openStream()`

In the process in which these methods are invoked, the `openConnection` method of the `java.net.JarURLConnection` class is invoked and the JAR file specified in the applicable URL is opened. This JAR file remains open unless the `close` method is explicitly invoked and cannot be deleted. Do not use these methods in the J2EE application. Also, when the operations for the JAR file are required and when using the `openConnection` method of the `java.net.JarURLConnection` class, make sure that you invoke the `close` method of the `JarFile` instance that the `getJarFile` method of the `java.net.JarURLConnection` returns.

(13) Notes on using the URLConnection class

When the `java.net.URLConnection` class uses the `setUseCaches(boolean)` method to acquire the connection for the specified URL, you can specify whether or not to use the cache information. When the `setUseCaches(false)` method is specified for the `URLConnection` class, the target object is generated for each connection. When the `URLConnection` class is used from the code in the J2EE application, memory might be insufficient for the J2EE server JavaVM.

(14) Notes related to the loading of the native library

Do not use the `System.loadLibrary` method to load the native library from the servlets and JSPs. If the native library is loaded with the servlets and JSPs, the error `java.lang.UnsatisfiedLinkError` might occur depending on the constraints of the JNI specifications. If you must load the native library, create the container extension library that invokes the `System.loadLibrary` method and implement settings so that the container extension library is referenced from the servlets and JSPs. For creating the container extension library, see *16. Container Extension Library* in the *uCosminexus Application Server Common Container Functionality Guide*.

(15) Using user thread

You can generate and use threads from the servlets and JSPs that form the application. The thread that a user explicitly generates in a program is called a *user thread*.

The user threads are classified as follows depending on the operation method after generating the threads (lifecycle):

- Threads operating within the scope of the `service` method and the `init` method.
- Threads operating in the background of the `service` method and the `init` method.

Usage conditions of the user thread

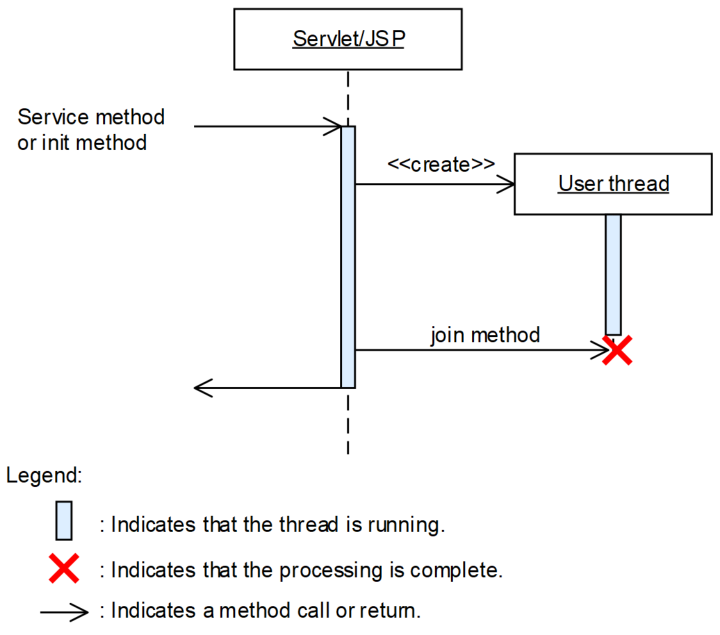
- The user thread cannot be used in the Enterprise Bean (since the generation of threads from the Enterprise Bean is prohibited in the EJB specifications).

The following points describe the lifecycle when using the user threads:

(a) When the threads operate within the scope of the service method and init method

In this model, the processing of the user thread is completed with the `service` method and the `init` method. The following figure shows the flow of processing in this model:

Figure 10–5: Processing when the thread operates within the scope of the service method and the init method

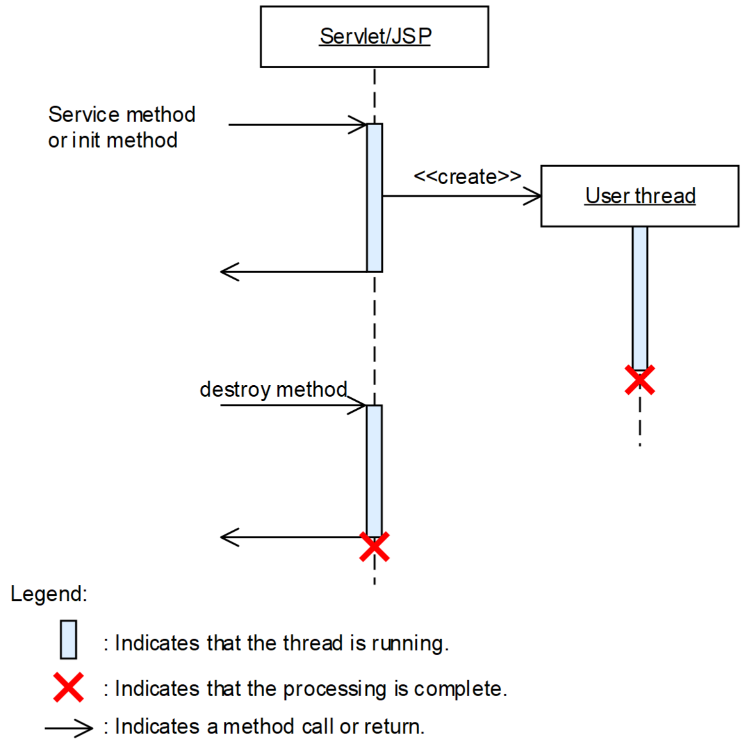

Servlet/JSP
Service method or init method
<<create>>
User thread
join method
Legend:
: Indicates that the thread is running.
: Indicates that the processing is complete.
: Indicates a method call or return.

The user thread will be generated within the invocation scope of the `service` method and the `init` method. The `service` method and the `init` method wait for the processing of the user thread by the `join` method to complete, and then the methods are returned.

(b) When the threads operate in the background of the service method and the init method

In this model, the user thread is generated with the `service` method and the `init` method, and then the user thread is operated in the background. The following figure shows the flow of processing in this model:

Figure 10–6: Processing when the threads operate in the background of the service method and the init method


Servlet/JSP
Service method or init method
<<create>>
User thread
destroy method
Legend:
: Indicates that the thread is running.
: Indicates that the processing is complete.
: Indicates a method call or return.

The `service` method and the `init` method that generated the user thread will be returned without waiting for the processing to complete after the user thread is generated. However, after the application is stopped, the J2EE service cannot be used from the user thread. Therefore, no problem occurs if a user thread is stopped by a `contextDestroyed` method of `javax.servlet.ServletContextListener` or a JSP or servlet `destroy` method invoked by the stopping of an application.

(16) Persistence of session information

The Web container does not support the persistence of the session information. Irrespective of whether the session information in the Web container is normal or abnormal, the information is lost once the Web container exits. If you want to maintain the session information, use the session failover functionality.

Furthermore, when the `<distributable>` tag is specified in `web.xml` or when the not `Serializable` object is registered as the session information, `IllegalArgumentException` does not occur.

(17) Messages output when the init method and destroy method are not overridden

If you initialize or terminate the servlets that do not override the `init` method and the `destroy` method, the log of the following format is output in the servlets log:

- Message ID: `KDJE39037-I`
- Message text: `path="aa....aa" :bb....bb: init`#

aa....aa

Indicates the context path starting with a forward slash (/).

bb....bb

Indicates the servlets name specified in the `<servlet-name>` tag of `web.xml`. In the case of servlets with default mapping, the name is `org.apache.catalina.INVOKER.`*class-name*.

#

In the case of the `init` method, the name is `init` and in the case of `destroy` method, the name is `destroy`. The output messages are the log output in the `init` method and `destroy` method of the `javax.servlet.GenericServlet` class respectively. Therefore, these messages are not output in the servlets that override the `init` method or the `destroy` method.

Also, in the case of JSP, if the `init` method and the `destroy` method are not overridden in the base class of JSP specified in the `extends` attribute of the `page` directive, a similar message is output. In this case, servlets name is `com.hitachi.software.web.servlet-name.jsp`. If the `extends` attribute of the `page` directive is not specified in JSP, only the `init` method log is output and the `destroy` method log is not output.

However, for both servlet and JSP, when the `init` method and the `destroy` method of the superclass are invoked by overriding the `init` method and the `destroy` method, this message is output.

(18) javax.servlet.error.exception attribute of the javax.servlet.ServletRequest object

The `javax.servlet.error.exception` attribute of the `javax.servlet.ServletRequest` object is separately described as follows for the case in which exception is thrown in the Servlet and when exception is thrown in the JSP file:

(a) When an exception is thrown in the servlet

When the exception class thrown in the servlet is java.lang.Error or an inherited class

The exception of the `javax.servlet.ServletException` class is set in the `javax.servlet.error.exception` attribute of the `javax.servlet.ServletRequest` object. The exception thrown in the servlet can be acquired using the `getRootCause` method of the `javax.servlet.ServletException` class.

When the exception class thrown in the servlet is a class other than java.lang.Error or an inherited class

The exception thrown in the servlet is set in the `javax.servlet.error.exception` attribute of the `javax.servlet.ServletRequest` object.

(b) When an exception is thrown in the JSP file

- **When the error page is a JSP file**

When the error page is specified in the <error-page> tag of web.xml

The error page specified in the `<error-page>` tag of `web.xml` is described separately for JSP 2.0 and later versions and JSP 1.2 version.

JSP 2.0 and later versions

The exception thrown in the JSP file is set in the `javax.servlet.error.exception` attribute of the `javax.servlet.ServletRequest` object.

JSP 1.2

If the exception class thrown in the JSP file is one of the following classes, the exception thrown in the JSP file is set in the `javax.servlet.error.exception` attribute of the `javax.servlet.ServletRequest` object.

- `java.io.IOException` or an inherited class
- `java.lang.RuntimeException` or an inherited class
- `javax.servlet.ServletException` or an inherited class

If the exception class thrown in the JSP file is other than the above-mentioned classes, the exception of the `javax.servlet.ServletException` class is set in the `javax.servlet.error.exception` attribute of the `javax.servlet.ServletRequest` object. The exception thrown in the JSP file can be acquired using the `getRootCause` method of the `javax.servlet.ServletException` class.

When the error page is specified in the errorPage attribute of the page directive

The error page specified in the `errorPage` attribute of the `page` directive is described separately for the case in which `true` is specified and when `false` is specified in the `isErrorPage` attribute of the `page` directive in the error page.

When true is specified in the isErrorPage attribute of the page directive in the error page

The exception thrown in the JSP file is set in the `javax.servlet.error.exception` attribute of the `javax.servlet.ServletRequest` object.

When false is specified in the isErrorPage attribute of the page directive in the error page

A value is not set in the `javax.servlet.error.exception` attribute of the `javax.servlet.ServletRequest` object.

- **When the error page is a Servlet**

When the error page is specified in the <error-page> tag of web.xml

The exception thrown in a servlet is the same as in the case in which the error page is a JSP file when the error page is specified in the `<error-page>` tag of `web.xml`.

When the error page is specified in the errorPage attribute of the page directive

A value is not set in the `javax.servlet.error.exception` attribute of the `javax.servlet.ServletRequest` object.

(19) Operating Web applications containing binary data

Note the following items to the Web applications containing binary data:

- When executing the requests to the binary data sent from the client
  Do not acquire `PrintWriter` from the response object in the filter applied in the request to the binary data.
- When the servlets or JSPs that process the requests sent from the client are to be dispatched
  Do not acquire `PrintWriter` from the response object in the following locations:
  - In the filter applied in the request to the binary data
  - In the servlets or JSPs to be dispatched to the binary data

Reference note

Binary data means static contents in which the MIME type mapped to the extension does not begin with `"text/"` or static contents for which mapping does not exist.

(20) Notes related to response character encoding

If the character encoding of the response body in the JSPs or servlets is UTF-16 (16 bit UCS conversion format), the character encoding might not be displayed correctly by the browser. In this case, use UTF-16BE (big-endian byte order of the 16 bit UCS conversion format) or UTF-16LE (little-endian byte order of the 16 bit UCS conversion format) for the character encoding of the JSPs or servlets.

(21) Return values of the getServerName method and getServerPort method of the javax.servlet.ServletRequest interface

This point describes the return values of the `getServerName` method and the `getServerPort` method.

In Servlet 2.4 and later specifications, the return values of the `getServerName` method and the `getServerPort` method differ depending on the availability of the Host header. The following table lists the return values of the `getServerName` method and the `getServerPort` method in the Servlet 2.4 and later specifications:

Table 10–13: Return values of the getServerName method and getServerPort method (In Servlet 2.4 or later specifications)

Presence of Host header	Return value of the getServerName method	Return value of the getServerPort method
Yes	Part before the colon (`:`) of the Host header	Part after the colon (`:`) of the Host header
No	Resolved server name or IP address	Port number of the server that receives the connection with the client

In Application Server, the return values of the `getServerName` method and the `getServerPort` method are acquired depending on the combination of the HTTP requests and the functionality used. Note that when the Host header is not included in the HTTP 1.1 request, 400 error occurs according to the HTTP 1.1 specifications. Furthermore, the HTTP 1.1 specifications define that if the request URI of the request line is an absolute URI, use the host of the request URI for the host and ignore the contents of the Host header. Though not explicitly mentioned in the Servlet specifications, the host name included in the request line URI is given priority based on the HTTP specifications.

The following table lists the return values of the `getServerName` method and the `getServerPort` method obtained depending on the combination of the HTTP requests and the functionality used. For details on the return values of the `getServerName` method and `getServerPort` method when the gateway specification functionality is used, see *Table 10-13*.

Table 10–14: Return values of the getServerName method and getServerPort method (In the application server)

HTTP request		Functionality used	Return value of the getServerName method	Return value of the getServerPort method
Presence of the Host header	URI type of the request line			
Yes	Absolute URI	Web server integration	Host name of request line	Port number of request line
		In-process HTTP server	Host name of request line	Port number of request line
	Relative URI	Web server integration	Host name of Host header	Port number of Host header
		In-process HTTP server	Host name of Host header	Port number of Host header
No	Absolute URI	Web server integration	Host name of request line	Port number of request line
		In-process HTTP server	Host name of request line	Port number of request line
	Relative URI	Web server integration	Host name or IP address of the Web server#2	Port number of the Web server

HTTP request		Functionality used	Return value of the getServerName method	Return value of the getServerPort method
Presence of the Host header	URI type of the request line			
		In-process HTTP server	Host name or IP address of the J2EE server#1	Port number of the in-process HTTP server

#1

Return value of the `java.net.InetAddress.getLocalHost` method or the `getHostName` method.

#2

Value specified in the `ServerName` directive when Cosminexus HTTP Server is used. For details on the ServerName directive, see the *uCosminexus Application Server HTTP Server User Guide*.

The following table lists the return values of the `getServerName` method and the `getServerPort` method when you use the gateway specification functionality:

Table 10–15: Return values of the getServerName method and getServerPort method when you use the gateway specification functionality (in the Application Server)

HTTP request		Functionality used	Return value of the getServerName method	Return value of the getServerPort method
Presence of the Host header	URI type of the request line			
Yes	Absolute URI	Web server integration	Host name of request line	Port number of request line
		In-process HTTP server	Host name of request line	Port number of request line
	Relative URI	Web server integration	Host name of Host header	Port number of Host header
		In-process HTTP server	Host name of Host header	Port number of Host header
No	Absolute URI	Web server integration	Host name specified in the gateway specification functionality	Port number specified in the gateway specification functionality
		In-process HTTP server	Host name of request line	Port number of request line
	Relative URI	Web server integration	Host name specified in the gateway specification functionality	Port number specified in the gateway specification functionality
		In-process HTTP server	Host name specified in the gateway specification functionality	Port number specified in the gateway specification functionality

(22) Acquiring the root cause exception specified in the constructor of the javax.servlet.ServletException class

With the Application Server, you can acquire the root cause exception specified in the constructor `ServletException(String, Throwable)` or `ServletException(Throwable)` using the `getCause` method. Note that you can also acquire the exception using the `getRootCause` method. However, with versions prior to 07-60 version, the `getCause` method returns null.

This point describes the compatibility parameters and notes related to the acquisition of the root cause exception specified in the constructor of the `javax.servlet.ServletException` class.

- Compatibility parameters

To perform the same operations as are operated with the versions prior to 07-60 version, specify `true` in the compatibility parameter `webserver.servlet_api.exception.getCause.backcompat` in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file. For details on the parameters, see the *uCosminexus Application Server Definition Reference Guide*.

The following table lists the differences in the return values of the `getCause` method and the `getRootCause` method depending on the value specified in the compatibility parameter `webserver.servlet_api.exception.getCause.backcompat`.

Table 10–16: Differences in the return values of the getCause method and getRootCause method depending on the value specified in webserver.servlet_api.exception.getCause.backcompat

Method	Value specified in webserver.servlet_api.exception.getCause.backcompat	
	true	false
`getCause()`	N	Y
`GetRemoteUser()`	Y	Y

Legend:

Y: Returns the root cause exception

N: Returns null

Note that the contents specified in the compatibility parameter are also applied to the operations of the `getCause` method and the `getRootCause` method of the `javax.servlet.jsp.JspException` class.

- Notes

  When the root cause exception can be acquired by implementing the `getCause` method, you cannot invoke `initCause(Throwable)` for the `ServletException` object generated by the constructor `ServletException(String, Throwable)` or `ServletException(Throwable)`. If `initCause(Throwable)` is invoked, the `java.lang.IllegalStateException` exception is thrown.

(23) Notes related to the execution of the flush method for the javax.servlet.ServletOutputStream object

With the Application Server, even if the `flush` method is executed after executing the `close` method for the `javax.servlet.ServletOutputStream` object obtained from the `javax.servlet.ServletResponse` object, the `java.io.IOException` exception is not thrown.

(24) Normalizing request URIs

With Application Server, the character strings included in request URIs are used in the following matching processes after normalizing the request URIs:

- Matching context path and context root
- Matching URL pattern of servlets and JSPs
- Matching default mapping
- Matching static contents
- Matching URL pattern of filter
- Matching with the `<error-page>` tag of `web.xml` or with the `errPage` attribute of the `page` directive of JSPs
- Matching URL pattern for restricting access

- Determining URL for login authentication
- Forwarding and including requests
- Matching URL pattern for HTTP response compression filter
- Matching URL pattern to control the number of concurrently executing threads in the URL group
- Error page customization of the in-process HTTP server
- Request distribution by redirecting the in-process HTTP server

(25) Return value of the getRequestURI and getRequestURL methods of the javax.servlet.http.HttpServletRequest interface

The normalized URL is considered as a return value in the `getRequestURI` and `getRequestURL` methods of the `javax.servlet.http.HttpServletRequest` interface.

(26) Specifying a Servlet or JSP mapped to a URL in the welcome file

When a request URL must be forwarded to the `welcome` file without matching it with a Servlet or JSP mapped to the URL, the forwarding-destination `welcome` file is selected as follows in the Web container:

First of all, from the specified `welcome` file name, the static contents and JSP file candidate are selected on priority. If no corresponding item exists, the candidate of the Servlet or JSP for which URL mapping is performed is selected.

The notes on the `welcome` file are as follows:

- **Limitations based on the welcome file forwarding method**
  The `welcome` file is forwarded based on HTTP redirect (the browser redirects to HTTP status code 302). This forwarding method has some limitations that must be noted when designing the URL.
    - When a POST request is received, the information of the request body sent from the browser cannot be inherited in the `welcome` file at the forwarding destination. Only when the posted information is in the form input format (`Content-Type` is `application/x-www-form-urlencoded`), the information can be inherited by assigning to the query string of the forwarding-destination URL of the `welcome` file created by the Web container. However, even in this case, consideration must be given to the fact that when the information of the request body is large in size, the forwarding-destination URL becomes too long, and the information is visible as is in the address bar of the browser in the form of the query string.
    - When the `doGet` method is not implemented in the Servlet of the forwarding-destination `welcome` file, `400 Bad Request` (for other than HTTP/1.1) or `405 Method Not Allowed` (for HTTP/1.1) is displayed on the browser.
    - When you invoke the `include` method of the `javax.servlet.RequestDispatcher` interface from the Web application, then in spite of specifying the directory in which the `welcome` file exists as the URL to be included, the contents of the `welcome` file of the forwarding destination are not inserted.
- **Adding the welcome file in an environment for which JSP pre-compilation is complete**
  When adding the JSP file specified in the `welcome` file to a Web application for which JSP pre-compilation is complete, you must again perform JSP pre-compilation after adding the JSP file. When you do not perform JSP pre-compilation again, the `welcome` file is not forwarded properly.
- **Specifying a servlet whose servlet class cannot be referenced in the welcome file**
  Do not specify a servlet whose servlet class cannot be referenced in the `welcome` file. When you specify a servlet whose servlet class cannot be referenced, the `welcome` file is not forwarded properly.
- **Requesting the welcome file to a path in which no directory exists**

When a request is sent to the path of a directory that does not exist as a resource within the Web application, the `welcome` file is not forwarded even when the request URL ends with a forward slash (`/`).

(27) Starting and stopping order of the servlet, filter and listener

If you start a Web application, perform the initialization process in the order below before starting the receipt of requests, according to the Servlet 2.4 specifications. In Cosminexus Application Server, the initialization process is performed in the same order even in Web applications of Servlet2.3 or earlier versions. The order of starting the servlets, filters, and listeners at the time of starting the Web application is as follows:

1. Starting the listener (generating an instance#1, invoking methods of the `@PostConstruct` annotation and the `contextInitialized` method of `ServletContextListener`#2)
2. Starting the filter (generating an instance#1, invoking the methods of `@PostConstruct` annotation and the `init` method)
3. Starting Servlet/JSP specified in the load-on-startup tag (generating an instance#1, invoking method of the `@PostConstruct` annotation and the `init` method)

#1: In Servlet 3.0 or later, you can dynamically add the servlets, filters, and listeners by API calling. However, for the servlets, filters, and listeners for which the definition is added by the API calling that specifies the instances, the instance has already been generated and hence the Web container does not generate an instance.

#2: Even if an exception occurs while invoking the `contextInitialized()` method of the listener, the system outputs the message of KDJE39103-E and continues the process of starting the Web application.

Note that for servlets for which the execution of the initialization process at the time of Web application startup is not specified with the *load-on-startup* element in `web.xml`, the system generates the instances of the servlet and invokes the `init()` method at the time of the initial request execution.

At this time, the instance generation and the `init()` method invocation of the servlet is performed before the filter.

The order of stopping the servlets, filters, and listeners at the time of stopping the Web application is as follows:

1. Stopping the already started Servlet/JSP (invoking the `destroy` method or the methods of the `@PreDestroy` annotation)
2. Stopping the filter (invoking the `destroy` method or the methods of the `@PreDestroy` annotation)
3. Stopping the listener (invoking the methods of the `@PreDestroy` annotation)

(28) Accessing the static contents within a Web application

You can use any one of the methods such as `GET`, `HEAD`, `POST`, `TRACE`, and `OPTIONS`, to access the static contents within a Web application.

When you use the `POST` method, same details of the static contents are sent as a response, as when you use the `GET` method.

(29) Precautions related to character encoding

In the same Web application, use the same character encoding in the error page specified in `web.xml` as the character encoding used for the servlets and JSPs that use character encoding in the HTTP response.

(30) Return value when only the part after equal (=) sign is used in the query character string

When only the part after equal sign ("=") is specified in the query character string of a request (for example, in the case of `http://localhost/application/getparam.jsp?=param`), the return value of the Servlet API that obtains the request parameter of `javax.servlet.ServletRequest` differs depending on the type of Web server in use.

The following are the respective return values of Servlet APIs for each type of the Web server:

- When using the Web server integration functionality of a simple Web server
  - `getParameter` method
    When you specify a blank character (""), the parameter specified after the equal sign ("=") is returned.
  - `getParameterMap` method
    The `java.util.Map` object that includes the parameter for which a blank character ("") acts as a key, is returned.
  - `getParameterNames` method
    The `java.util.Enumeration` object that includes a blank character (""), is returned.
  - `getParameterValues` method
    When you specify a blank character (""), the parameter specified after the equal sign ("=") is returned.
- When using the in-process HTTP server
  - `getParameter` method
    Even if you specify a blank character (""), null is returned.
  - `getParameterMap` method
    A blank `java.util.Map` object is returned.
  - `getParameterNames` method
    A blank `java.util.Enumeration` object is returned.
  - `getParameterValues` method
    Even if you specify a blank character (""), null is returned.

(31) containsHeader method of javax.servlet.http.HttpServletResponse instance

The following response headers are sometimes automatically set to responses, by the Web container. You cannot use the `containsHeader` method of the `javax.servlet.http.HttpServletResponse` interface to check whether such response headers are set for responses.

- For Web server integration
  - `Content-Length`
  - `Content-Type`
  - `Set-Cookie`
- For the in-process HTTP server
  - `Connection`
  - `Content-Language`

- `Content-Length`
- `Content-Type`
- `Date`
- `Server`
- `Set-Cookie`
- `Transfer-Encoding`

(32) Precautions related to the libraries of Application Server

If you include the libraries of Application Server in J2EE applications, it may lead to incorrect operations during the start and execution of the application import, due to reasons like a conflict in the library version. Therefore, do not include the libraries of Application Server in a J2EE application, except for cases where inclusion of the libraries is specified as a method of using the product.

10.4.2 Precautions related to the specifications that are added or changed in the EL2.2 specifications

This section describes the points to be noted when using added or changed EL2.2 specifications with Application Server. For details on the EL2.2 specifications, see *EL2.2 Specifications*.

- Although Application Server supports EL2.2, both EL2.1 and EL2.2 implementations are available. You can switch the implementation by specifying the following parameter in the Easy Setup definition file. You must specify the parameter in the *configuration* tag of the logical J2EE server in the Easy Setup definition file.

  ```
  webserver.jsp.el2_2.enabled
  ```

  For details on the `webserver.jsp.el2_2.enabled` parameter, see *4.11.2 Parameters used for setting up the user properties for the J2EE server* in the manual *uCosminexus Application Server Definition Reference Guide*.
- If multiple methods with the same number of parameters are specified in a single Bean, the type conversion is done based on the method that is defined first. Accordingly, if the type of the argument when calling a method matches a method that is defined first, the method is called without any problem; otherwise, if the argument type does not match, ELException is thrown.

11 Files Used in J2EE Servers

This chapter describes the storage location, functionality, and format of the files used in J2EE servers and the keys that you can specify in the files. The chapter describes only the content that differs from the recommended mode.

11.1 Details on the files used on J2EE servers

11.1.1 usrconf.properties (User property file for J2EE servers)

(1) Keys beginning with cosminexus.jpa

The following table lists the specifiable keys. Note that *Default value* is the value that is assumed when a key is not specified.

VR is the version of Application Server on which the keys are introduced or changed.

Key name	Contents	Default value	VR
`cosminexus.jpa.logging.level.operation.`*category*	Specify the log level for each category of the JPA provider operation log, when you are using the JPA functionality of the J2EE server. The category name and log level are case sensitive. If this key is not specified, the log is not output to the operation log. Since this affects the security and performance, take care when you specify the output level settings. If you specify `Off`: The log is not output to the JPA provider operation log. If you specify `Information`: The JPA operation information is output to the JPA provider operation log. If you specify `Detail`: The detailed JPA operation information and the information that was output to `Information` is output to the JPA provider operation log.	`Off`	08-00
`cosminexus.jpa.exception.logging.sql`#	Specify whether the exception message will include the SQL statement that caused the exception, when the JPA provider executes the SQL statement and receives an exception from the database. If you specify `Off`: The SQL statement executed by the JPA provider and the value specified for the `?` parameter (place holder) is not included in the exception message. If you specify `Information`: The SQL statement executed by the JPA provider is included in the exception message. If you specify `Detail`: The SQL statement executed by the JPA provider and the value specified for the `?` parameter (place holder) is included in the exception message.	`Off`	08-00

#

The values specified in this property are also applied to the contents output to the exception log.

For `Information` and `Detail`, the contents of the SQL statement and the `?` parameter (place holder) is also output to the exception log; and therefore, you consider the security. Specify the values as and when required for development and maintenance.

When handling binary data, the hash value of the binary object is output to the `?` parameter (place holder).

Before you complete the preparation for issuing an SQL statement, if a failure occurs in acquiring a connection due to communication errors, you might not be able to acquire value of the `?` parameter (place holder).

(2) Keys beginning with ejbserver.jpa

The following table lists the specifiable keys. Note that *Default value* is the value that is assumed when a key is not specified.

VR is the version of Application Server on which the keys are introduced or changed.

Key name	Contents	Default value	VR
`ejbserver.jpa.defaultJtaDsName`	Specify the default JTA data source references. This property is used when `jta-data-source` is not specified in `persistence.xml` or when a space is specified.	None	08-00
`ejbserver.jpa.defaultNonJtaDsName`	Specify the default non-JTA data source references. This property is used when `non-jta-data-source` is not specified in `persistence.xml` or when a space is specified.	None	08-00
`ejbserver.jpa.defaultProviderClassName`	This property specifies the default JPA provider class name. This property is used when `provider` is not specified in `persistence.xml` or when a space is specified.	`com.hitachi.software.jpa.PersistenceProvider`	08-00
`ejbserver.jpa.disable`#	Specify this property when using the JPA functionality of Application Server. If you specify `true`: The JPA functionality of Application Server is disabled. If you specify `false`: The JPA functionality of Application Server is enabled.	`false`	08-20
`ejbserver.jpa.overrideJtaDsName`	Specify the JTA data source references that will be used with a higher priority than the values specified in `jta-data-source` in `persistence.xml` and the values specified in `ejbserver.jpa.defaultJtaDsName`.	None	08-00
`ejbserver.jpa.overrideNonJtaDsName`	Specify the non-JTA data source references that will be used with a higher priority than the values specified in `non-jta-data-source` in `persistence.xml` and the values specified in `ejbserver.jpa.defaultNonJtaDsName`.	None	08-00
`ejbserver.jpa.overrideProvider`	Specify the JPA provider class name that will be used with a higher priority than the values specified in `provider` in `persistence.xml` and the values specified in `ejbserver.jpa.defaultProviderClassName`.	None	08-00
`ejbserver.jpa.emfprop.`*property-key*	Specify the JPA provider-specific property keys. When all the persistence units are deployed, the properties with the prefix `ejbserver.jpa.emfprop.` removed will be passed to the JPA provider.	None	08-00

#

Notes for specifying `ejbserver.jpa.disable=true`

If the application includes `persistence.xml`, Application Server does not read `persistence.xml` when the application starts. Also, if you are using the reload functionality of the application, update detection is not performed for `persistence.xml`.

If the application is using a managed persistent context or persistent unit of Application Server, you will not be able to start the application.

(3) Keys beginning with ejbserver.logger

The following table lists the specifiable keys. Note that *Default value* is the value that is assumed when a key is not specified.

VR is the version of Application Server on which the keys are introduced or changed.

Related information is the reference location for information related to the specified key. *uCosminexus Application Server* is omitted from the manual names.

Key name	Contents	Default value	VR	Related information
`ejbserver.logger.channels.define.`*channel-name*[#]`.filenum`	Specify an integer from 1 to 16 for the number of log files of a J2EE server.	• `16`, when the channel name is `WebAccessLogFile` • `4`, when the channel name is `MaintenanceLogFile` or `WebServletLogFile` • `2`, when the channel name is other than above	--	
`ejbserver.logger.channels.define.`*channel-name*[#]`.filesize`	Specify an integer from 4096 to 2147483647 (units: bytes) for the size of the log files of a J2EE server.	• `2097152`, when the channel name is `WebAccessLogFile` • `16777216`, when the channel name is `MaintenanceLogFile` • `4194304`, when the channel name is `WebServletLogFile` • `1048576`, when the channel name is other than above	--	

Legend:

--: Indicates a version earlier than the version 08-00.

Blank cell: Related information does not exist.

#

You can set the following names as channel name:

- Channels for output of Cosminexus system log:
  `MessageLogFile`, `MaintenanceLogFile`, `ExceptionLogFile`, `ConsoleLogFile`, `EJBContainerLogFile`, `WebContainerLogFile`, `WebServletLogFile`, `UserOutLogFile`, `UserErrLogFile`, `WebAccessLogFile`, `JPAOperationLogFile`, `JPAMaintenanceLogFile`
- Channels for output of resource depletion monitoring log
  `MemoryWatchLogFile`, `FileDescriptorWatchLogFile`, `ThreadWatchLogFile`, `ThreaddumpWatchLogFile`, `RequestQueueWatchLogFile`, `HttpSessionWatchLogFile`, `ConnectionPoolWatchLogFile`

For details about the acquisition of documents, see *2.3 Acquiring the Data* in the manual *uCosminexus Application Server Maintenance and Migration Guide*.

(4) Keys beginning with ejbserver.server

The following table lists the specifiable keys. Note that *Default value* is the value that is assumed when a key is not specified.

VR is the version of Application Server on which the keys are introduced or changed.

Related information is the reference location for information related to the specified key. *uCosminexus Application Server* is omitted from the manual names.

Key name	Contents	Default value	VR	Related information
`ejbserver.server.eheap.ajp13.enabled`	Specify whether to deploy the objects for communication with the redirector on the Explicit heap. If you specify `true`: The objects for communication with the redirector will be deployed on the Explicit heap. If you specify `false`: The objects for communication with the redirector will be deployed on the Java heap area. However, if JavaVM option `HitachiUseExplicitMemory` is disabled, this property is disabled (same as for the case when `false` is specified).	`true`	08-00	

Legend:

Blank cell: Related information does not exist.

(5) Keys beginning with webserver.connector

The following table lists the specifiable keys. Note that *Default value* is the value that is assumed when a key is not specified.

VR is the version of Application Server on which the keys are introduced or changed.

Related information is the reference location for information related to the specified key. *uCosminexus Application Server* is omitted from the manual names.

Key name	Contents	Default value	VR	Related information
`webserver.connector.ajp13.backlog`	Specify the maximum number of rows and columns awaiting connection requests from the redirector. Specify the value using an integer from 1 to 2147483647. The valid maximum value becomes the maximum value of the `Listen` queue in the Socket specifiable on the platform to be used. The actual maximum value of the `Listen` queue depends on the OS. For details, see the manuals for the listen functionality in each OS. The value specified in this key is set in the `backlog` argument of `java.net.ServerSocket` class constructor. If, however, this specified value exceeds the limit value set for the OS, it is interpreted as the limit value of the OS, and it therefore does not result in an error. The limit value is different for every OS. For details on how to extend the limit value, see the OS manual.	`100`	--	
`webserver.connector.ajp13.bind_host`	Specify the IP address or the host name used for Web server integration. The single-byte space before and after the IP address or the host name is ignored. If you do not specify a value, the wild card address is used. When specifying this property, you need to specify the local host name or the IP address even in the worker host name.	None	--	*5.7 Specifying the IP address (Web server integration)*

Key name	Contents	Default value	VR	Related information
	In a configuration wherein a Web server is integrated with another Web server running on the same host, the Web Container cannot receive requests from the Web server if either of the following settings is specified: • The local host name or IP address is specified in the `webserver.connector.ajp13.bind_host` property, and the loopback address of a host, such as localhost, is specified in the worker host name of the redirector. • The loopback address of a host, such as localhost, is specified in the `webserver.connector.ajp13.bind_host` property, and the local host name or IP address is specified in the worker host name of the redirector.			
`webserver.connector.ajp13.max_threads`	Specify the concurrently executing number of requests being processed by the Web container.#1 Specify the value using an integer from 1 to 1024. Number of threads equal to the concurrently executing number of specified requests is generated when the server is started.	`10`	--	*Web Container Functionality Guide*
`webserver.connector.ajp13.port`	Specify the port number used for communication with the Web server. Specify the value using an integer from 1 to 65535. You cannot specify a port number that is already being used or secured for another application. Furthermore, do not specify the same value in port numbers of the ports to be used to communicate with the Web server in multiple J2EE servers. The `cjstartsv` command cannot start up multiple J2EE servers in which identical port numbers are specified.	`8007`	--	
`webserver.connector.ajp13.receive_timeout`	Specify an integer from 0 to 3600 (units: seconds) for the period to await a response (communication timeout value) from the redirector, while the data request is being processed in the redirector (that processes requests). If you specify `0`, the waiting period continues until a response is received from the redirector and the timeout will not occur.	`600` (seconds)	--	
`webserver.connector.ajp13.send_timeout`	Specify an integer from 0 to 3600 (units: seconds) for the timeout value for sending response. If you specify a non-numeric value, or a numeric value outside the range, a message will be output and the default value will be used. If `0` or a time period longer than the resend timer of data transmission present in the TCP is set, the timeout value will be the timeout value of the TCP. In such a case, a message indicating that an invalid timeout value has been specified will not be output.	`600`	--	
`webserver.connector.inprocess_http.backlog`	Specify an integer from 1 to 2147483647 for the length of the TCP listen queue that saves the connection requests from the Web client.	`511`	--	

Key name	Contents	Default value	VR	Related information
	The maximum valid specified value or the length of the TCP listen queue that is actually set, is different for every OS. If you specify a non-numeric value, a numeric value outside the range, a null character string, or a whitespace#2, a message is output and the default value will be set.			
webserver.connector.inprocess_http.bind_host	Specify the IP address or the host name used in an in-process HTTP server. The single-byte space before and after the IP address or the host name is ignored. If you do not specify a value, the wild card address is used. If the specified host name or the IP address cannot be resolved and if you specify the name or IP address of a host that is not local, a message is output and the wild card address is used.	None	--	
webserver.connector.inprocess_http.enabled	Specify whether to enable the in-process HTTP server functionality. If you specify `true`: The in-process HTTP server functionality will be enabled. If you specify `false`: The in-process HTTP server functionality will be disabled. If you specify a string other than `true` or `false` or if you specify a null character string or a whitespace#2, a message is output and the default value will be set. Note that, if the in-process HTTP server functionality is enabled, Web server integration cannot be used.	`false`	--	
webserver.connector.inprocess_http.enabled_methods	Specify the HTTP methods that have access permission. When specifying multiple methods, demarcate them with a comma (`,`). In the method name, specify the method that has been defined in HTTP/1.1. If you specify an asterisk (`*`), all methods are permitted. HTTP methods are case-sensitive so the value specified in this property is also case-sensitive. In the method name, you need to use the value provided in RFC2616. You cannot, however, specify the string "`*`" as a method name. The whitespace#2 before and after each method name is ignored. If you specify an invalid value, a null character string, or a whitespace#2, a message is output and the default value will be set.	`GET`, `HEAD`, `POST`, `PUT`, `DELETE`, `OPTIONS`	--	
webserver.connector.inprocess_http.error_custom.list	Specify the definition name for error page customization used in the error page customization functionality. The maximum length of the value that can be specified is 1024 characters. Specify a string consisting of alphanumeric characters (A-Z, a-z, 0-9) or underscores (_). The string length of one definition name is from 1 to 32 characters.	None	--	

Key name	Contents	Default value	VR	Related information
	When specifying multiple definition names, demarcate them with a comma (`,`). The whitespace[#2] before and after the comma is ignored. The same definition name for error page customization cannot be specified multiple times. If you specify an invalid value, a message is output and all the definitions for error page customization are disabled.			
`webserver.connector.inprocess_http.error_custom.`*error-page-customization-definition-name*`.file`	Specify an absolute path for the file used as the response body during error page customization with the error page customization functionality. Use a forward slash (`/`) as the demarcation sign in the path. If this property is set by using the definition name for error page customization that has not been specified in `webserver.connector.inprocess_http.error_custom.list`, the property is disabled. In the definition name for error page customization specified in `webserver.connector.inprocess_http.error_custom.list`, make sure that you specify either this property, or `webserver.connector.inprocess_http.error_custom.`*error-page-customization-definition-name*`.redirect_url`. If you specify both the properties or if you do not specify any of the properties, or if you do not specify as an absolute path, or if you specify a file that does not exist, or a file for which there is no read permission, a message is output and this definition for error page customization is disabled. If you specify a null character string, or a whitespace[#2], the property is disabled.	None	--	
`webserver.connector.inprocess_http.error_custom.`*error-page-customization-definition-name*`.file.content_type`	Specify the value of Content-Type header of the response during error page customization using the error page customization functionality. If this property is set by using the definition name for error page customization that is not specified in `webserver.connector.inprocess_http.error_custom.list`, the property is disabled. If `webserver.connector.inprocess_http.error_custom.`*error-page-customization-definition-name*`.file` is not set, the property is disabled.	`text/html`	--	
`webserver.connector.inprocess_http.error_custom.`*error-page-customization-definition-name*`.redirect_url`	Specify the redirect URL as an absolute path using the error page customization functionality. If this property is set by using the definition name for error page customization that is not specified in `webserver.connector.inprocess_http.error_custom.list`, the property is disabled. In the definition name for error page customization specified in `webserver.connector.inprocess_http.error_custom.list`, make sure that you specify	None	--	

Key name	Contents	Default value	VR	Related information
	either this property or `webserver.connector.inprocess_http.error_custom.`*error-page-customization-definition-name*`.file`. Whether the value is correct is not verified, so you need to check through actual operations.			
`webserver.connector.inprocess_http.error_custom.`*error-page-customization-definition-name*`.request_url`	Specify an absolute path beginning with a forward slash (`/`) for the request URL that applies the error page customization with the error page customization functionality. You can specify the wild card (`*`) only once, immediately after the forward slash. An asterisk (`*`) is always interpreted as the wild card, so it cannot be used as a normal character. The value specified in this property and the value specified in `webserver.connector.inprocess_http.`*error_custom.error-page-customization-definition-name*`.status`, must not match with any other definition for error page customization. If this property is set by using the definition name for error page customization that is not specified in `webserver.connector.inprocess_http.error_custom.list`, the property is disabled. If you specify an invalid value, a message is output and this definition for error page customization is disabled.	`/*`	--	
`webserver.connector.inprocess_http.error_custom.`*error-page-customization-definition-name*`.status`	Use the error page customization functionality and specify an integer from 400 to 599 for the response status code that performs error page customization. The value specified in this property and the value specified in `webserver.connector.inprocess_http.error_custom.`*error-page-customization-definition-name*`.request_url`, must not match with any other definition for error page customization. If this property is set by using the definition name for error page customization that is not specified in `webserver.connector.inprocess_http.error_custom.list`, the property is disabled. Make sure that you specify this property for setting the definition name for error page customization that has been specified in `webserver.connector.inprocess_http.error_custom.list`. If you do not specify this property, or if you specify an invalid value, a message is output and this definition for error page customization is disabled.	None	--	
`webserver.connector.inprocess_http.gateway.host`	Specify the host name or the IP address of the gateway. When requests without a Host header are redirected to files, such as the `welcome` file, the host name of the URL specified in the Location header becomes the specified value.	None	--	
`webserver.connector.inprocess_http.gateway.port`	Specify the port number of the gateway by using an integer from 1 to 65535.	None	--	

Key name	Contents	Default value	VR	Related information
	If a request has no Host header and the request is to be redirected to a location such as a welcome file, the port number portion of the URL that is specified in the `Location` header becomes the specified value. This property specification is ignored when `webserver.connector.inprocess_http.gateway.host` is not specified. If `webserver.connector.inprocess_http.gateway.host` is specified and this property is omitted, the following value is set: • If you specify `true` for `webserver.connector.inprocess_http.gateway.https_scheme`: `443` • If you specify `false` for `webserver.connector.inprocess_http.gateway.https_scheme`: `80` If a string other than a numeric value or a numeric value outside the valid range is specified, a message appears and the value is considered not to have been specified.			
`webserver.connector.inprocess_http.gateway.https_scheme`	When a client request uses https as a scheme, and the scheme for a Web server will become http by using an SSL accelerator, specify `true`. If you specify `true`: https is assumed to be used as the scheme for requests sent to the Web server. If you specify `false`: No action occurs. If you specify a string other than `true` or `false` or if you specify a null character string or a whitespace#2, a message is output and the default value will be set.	`false`	--	
`webserver.connector.inprocess_http.hostname_lookups`	Specify whether the Web container should perform reverse lookup of the host name and convert the IP address of the client to the host name, for a request received by the in-process HTTP server. The throughput, however, will decline In the case of reverse lookup of the host name. If the host name is not resolved, the result of the `getRemoteHost()` method of `javax.servlet.ServletRequest` interface and the client IP address output to the log file will be in the format wherein a dot (`.`) is used for demarcation. If you specify `true`: The IP address will be converted to the host name. If you specify `false`: The IP address will not be converted to the host name. If you specify a string other than true or false or if you specify a null character string or a whitespace#2, a message is output and the default value will be set.	`false`	--	

Key name	Contents	Default value	VR	Related information
`webserver.connector.inprocess_http.init_threads`	Specify an integer from 1 to 1024 for the number of request processing threads of in-process HTTP server, generated when the server is started. The specified value must be lower than the maximum number of connections with the Web client (value specified in `webserver.connector.inprocess_http.max_connections`). If you specify a value greater than the maximum number of connections with the Web client, a message is output and the maximum number of connections with the Web client is set as the value. Furthermore, the maximum valid value differs based on the OS. If you specify a non-numeric value, a numeric value outside the range, a null character string or a whitespace#2, a message is output and the default value will be set.	`10`	--	
`webserver.connector.inprocess_http.keep_start_threads`	Specify whether to maintain the number of threads that are created when the server is started. If you specify `true`: The number of threads created when the server is started will be maintained. Even if the number of spare threads maintained in the pool exceeds the maximum number of spare threads (value specified in `webserver.connector.inprocess_http.max_spare_threads`), the number of threads created during the startup of the server will not be reduced. If you specify `false`: The number of threads created when the server is started will not be maintained. Adjustment will be made on the basis of the maximum number and the minimum number of threads maintained as spare threads. If the number of threads created when the server is started is lesser than the minimum number of spare threads (value specified in `webserver.connector.inprocess_http.min_spare_threads`), the number of threads is maintained as per the value specified in the minimum number of spare threads, irrespective of the settings in this property. If you specify `false` in this property, adjustment is done so that the threads created when the server is started become lesser than the maximum number of spare threads. If the request-processing threads created when a server is started are greater than the maximum number of spare threads, the threads exceeding the maximum number of spare threads get destroyed one by one at an interval of one second after the server is started. If you specify a string other than `true` or `false` or if you specify a null character string or a whitespace#2, a message is output and the default value will be set.	`false`	--	

Key name	Contents	Default value	VR	Related information
`webserver.connector.inprocess_http.limit.max_headers`	Specify an integer from 0 to 32767 for the upper limit of the number of HTTP headers included in the HTTP requests. If you do not want to set the upper-limit value, specify `0`. Even if the number of HTTP headers specified in this property is not met, an error occurs if the size specified in `webserver.connector.inprocess_http.limit.max_request_header` is exceeded. If you specify a non-numeric value, a numeric value outside the range, a null character string or a whitespace#2, a message is output and the default value will be set.	`100`	--	
`webserver.connector.inprocess_http.limit.max_request_body`	Specify an integer from -1 to 2147483647 (units: bytes) for the maximum size of the request body of an HTTP request. If you do not want to set the upper-limit value, specify `-1`. If the request body is sent in chunk format, the size of the chunk header also needs to be included in the specified size. If you specify a non-numeric value, a numeric value outside the range, a null character string or a whitespace#2, a message is output and the default value will be set.	`-1`	--	
`webserver.connector.inprocess_http.limit.max_request_header`	Specify an integer from 7 to 65536 (units: bytes) for the maximum size of the request header of an HTTP request. Even if the maximum size of the request header that has been set in this property is not met, an error occurs if the HTTP header specified in `webserver.connector.inprocess_http.limit.max_headers` is exceeded. The linefeed characters (double bytes of `CR(0x0d)+LF(0x0a)`) indicating the end of the HTTP header also need to be included in the specified size. If you specify a non-numeric value, a numeric value outside the range, a null character string or a whitespace#2, a message is output and the default value will be set.	`16384`	--	
`webserver.connector.inprocess_http.limit.max_request_line`	Specify either -1 or an integer from 7 to 8190 for the maximum length (units: bytes) of the request line. If you do not specify the upper-limit, specify `-1`. The request line includes URI and HTTP version that include the HTTP methods and the query strings. The value to be specified needs to be lower than the maximum size of the request header (value specified in `webserver.connector.inprocess_http.limit.max_request_header`). If you specify a value greater than the maximum size of the request header, a message is output, and the maximum size of the request header is set as the maximum length of the request line. The linefeed characters (double bytes of `CR(0x0d)+LF(0x0a)`) indicating the end of the HTTP header also need to be included in the size to be specified.	`8190`	--	

Key name	Contents	Default value	VR	Related information
	If you specify a non-numeric value, a numeric value outside the range, a null character string or a whitespace#2, a message is output and the default value will be set.			
`webserver.connector.inprocess_http.max_connections`	Specify an integer from 1 to 1024 for the maximum number of connections with the Web client. The valid maximum value is different for every operating OS. The value specified in this parameter becomes the maximum value for the request-processing threads. If you specify a non-numeric value, a numeric value outside the range, a null character string or a whitespace#2, a message is output and the default value will be set.	`100`	--	
`webserver.connector.inprocess_http.max_execute_threads`	Specify an integer from 1 to 1024 for the number of concurrently executing requests processed by the Web container. The specified value must be lower than the maximum number of connections with the Web client (value specified in `webserver.connector.inprocess_http.max_connections`). If you specify a value greater than the maximum number of connections with the Web client, a message is output and the maximum number of connections with the Web client is set in the value. If you specify a non-numeric value, a numeric value outside the range, a null character string or a whitespace#2, a message is output and the default value will be set.	`10`	--	
`webserver.connector.inprocess_http.max_spare_threads`	Specify an integer from 1 to 1024 for the maximum number of spare threads stored in a pool. The specified value must be lower than the maximum number of connections with the Web client (value specified in `webserver.connector.inprocess_http.max_connections`). If you specify a value greater than the maximum number of connections with the Web client, a message is output and the maximum number of connections with the Web client is set in the value. If you specify a non-numeric value, a numeric value outside the range, a null character string or a whitespace#2, a message is output and the default value will be set.	`20`	--	
`webserver.connector.inprocess_http.min_spare_threads`	Specify an integer from 1 to 1024 for the minimum number of spare threads stored in a pool. The specified value needs to be lower than the maximum number of spare threads stored in the pool (value specified in `webserver.connector.inprocess_http.max_spare_threads`). If a value greater than the maximum number of spare threads stored in the pool is set, a message is output, and the maximum number of spare threads stored in the pool is set as the minimum number of spare threads stored in the pool.	`5`	--	

Key name	Contents	Default value	VR	Related information
	If you specify a non-numeric value, a numeric value outside the range, a null character string or a whitespace#2, a message is output and the default value will be set.			
`webserver.connector.inprocess_http.permitted.hosts`	Specify the IP address (decimals) or the name of the host that has access permission for the in-process HTTP server. If you specify multiple hosts, demarcate the IP addresses or the host names with a comma (`,`). If there are no access restrictions, specify only an asterisk (`*`). Note that even if the local host (the address#3 associated with `localhost`) is not explicitly specified, access is always allowed. If you specify a null character string or a whitespace#2, a message is output and the default value will be set. If the specified host name cannot be resolved, a message is output and only access from the local host (the address#3 associated with `localhost`) is allowed. Note that the single-byte space before and after the IP address or the host name is ignored.	`*`	--	
`webserver.connector.inprocess_http.persistent_connection.max_connections`	Specify an integer from 0 to 1024 for the maximum number of TCP connections maintained in a Persistent Connection. The value to be set must be lower than the maximum number of connections with the Web client (value specified in `webserver.connector.inprocess_http.max_connections`). If you set a value greater than the maximum number of connections with the Web client, a message is output and the maximum number of connections with the Web client is set as the maximum number of TCP connections stored in the Persistent Connection. If you specify a non-numeric value or a numeric value outside the range, a message is output and the value specified in `webserver.connector.inprocess_http.max_connections` is set as the default value. If you specify a null character string or a whitespace#2, the value specified in `webserver.connector.inprocess_http.max_connections` is set as the default value.	Value specified in `webserver.connector.inprocess_http.max_connections`	--	
`webserver.connector.inprocess_http.persistent_connection.max_requests`	Specify an integer from 0 to 2147483647 for the upper limit of the number of serial connections when TCP connections are sustained by a Persistent Connection. If you do not want to set the upper-limit value, specify `0`. If you specify a non-numeric value, a numeric value outside the range, a null character string or a whitespace#2, a message is output and the default value will be set.	`100`	--	

Key name	Contents	Default value	VR	Related information
`webserver.connector.inprocess_http.persistent_connection.timeout`	Specify an integer from 0 to 3600 for the request wait period (units: seconds), when the TCP connections are sustained in a Persistent Connection. If you specify `0`, the timeout does not occur. If you specify a non-numeric value, a numeric value outside the range, a null character string or a whitespace#2, a message is output and the default value will be set.	`3`	--	
`webserver.connector.inprocess_http.port`	Specify an integer from 1 to 65535 for the port number used by the in-process HTTP server. You cannot specify a port number that is already being used by another application. If you specify a port number that is in use or has been secured by another application, a message is output and the J2EE server does not start. If you specify a non-numeric value, a numeric value outside the range, a null character string or a whitespace#2, a message is output and the default value will be set.	`80`	--	
`webserver.connector.inprocess_http.receive_timeout`	Specify an integer from 0 to 3600 for the period until timeout (units: seconds), when requests are received from the Web client. If you specify `0`, the timeout does not occur. If you specify a non-numeric value, a numeric value outside the range, a null character string or a whitespace#2, a message is output and the default value will be set.	`300`	--	
`webserver.connector.inprocess_http.redirect.`*redirect-definition-name*`.file`	Specify an absolute path for the file used as the response body during redirection with the redirect functionality. Use a forward slash (`/`) as the demarcation sign in the path. If you specify `200` in `webserver.connector.inprocess_http.redirect.`*redirect-definition-name*`.status`, always specify this property. If you specify `200` in `webserver.connector.inprocess_http.redirect.`*redirect-definition-name*`.status`, and this property is not specified, a message is output and this redirect definition is disabled. If you set this property by using a redirect definition name that is not specified in `webserver.connector.inprocess_http.redirect.list` and if you specify a null character string or a whitespace#2, the property is disabled. If you specify a value that is not an absolute path, a message is output and this redirect definition name is disabled. If you specify a file that does not exist or a file that does not have read permission, a message is output and this redirect definition is disabled.	None	--	
`webserver.connector.inprocess_http.redirect.`*redirect-definition-name*`.file.content_type`	Specify the value of the response Content-Type header during redirection, using the redirect functionality. If this property is set by using a redirect definition name that is not set in `webserver.connector.inprocess_http.redirect.list`, the property is disabled.	`text/html`	--	

Key name	Contents	Default value	VR	Related information
	If the `webserver.connector.inprocess_http.redirect.`*redirect-definition-name*`.file` is not set, this property is disabled.			
`webserver.connector.inprocess_http.redirect.`*redirect-definition-name*`.redirect_url`	Specify the redirect URL as an absolute URL, using the redirect functionality. If you specify `200` in `webserver.connector.inprocess_http.redirect.`*redirect-definition-name*`.status`, this property cannot be set. If you specify `200` in `webserver.connector.inprocess_http.redirect.`*redirect-definition-name*`.status`, and you specify this property, a message is output and the redirect definition name is disabled. If you specify a value other than 200 in `webserver.connector.inprocess_http.redirect.`*redirect-definition-name*`.status`, make sure that you specify this property. If you specify a value other than 200 in `webserver.connector.inprocess_http.redirect.`*redirect-definition-name*`.status` and this property is not specified, a message is output and the redirect definition is disabled. If this property is set by using a redirect definition name that is not set in `webserver.connector.inprocess_http.redirect.list`, the property is disabled. Whether the value is correct is not verified, so you need to check through actual operations.	None	--	
`webserver.connector.inprocess_http.redirect.`*redirect-definition-name*`.request_url`	Specify an absolute path beginning with a forward slash (`/`) for the request URL performing redirection with the redirect functionality. You can specify the wild card (`*`) only once, immediately after the forward slash. The wild card indicates any string of 0 or more characters. An asterisk (`*`) is always interpreted as the wild card, so it cannot be used as a normal character. You also cannot specify a value same as that specified in another redirect definition. For the redirect definition name that has been specified in `webserver.connector.inprocess_http.redirect.list`, you always need to specify this property. If you do not specify this property, a message is output and the redirect definition is disabled. If this property is set by using a redirect definition name that is not set in `webserver.connector.inprocess_http.redirect.list`, the property is disabled. If you specify an invalid value, a message is output and the redirect definition is disabled.	None	--	
`webserver.connector.inprocess_http.redirect.`*redirect-definition-name*`.status`	Specify the response status code (`200`, `300`, `301`, `302`, `303`, `305`, `307`) during redirection with the redirect functionality. If this property is set by using a redirect definition name that is not set in	`302`	--	

Key name	Contents	Default value	VR	Related information
	`webserver.connector.inprocess_http.redirect.list`, the property is disabled. If you specify an invalid value, a null character string, or a whitespace#2, a message is output and the default value will be set.			
`webserver.connector.inprocess_http.redirect.list`	Specify the redirect definition name used in the redirect functionality. The maximum length of the value that can be specified in this property is 1024 characters. Specify the redirect definition name with a string consisting of alphanumeric characters (A-Z, a-z, 0-9) or underscores (_). The string length of one redirect definition name is from 1 to 32 characters. When specifying multiple redirect definition names, demarcate them with a comma (`,`). The whitespace#2 before and after the comma is ignored. You cannot specify the same redirect definition name multiple times. If you specify an invalid value, a message is output and all redirect definitions are disabled.	None	--	
`webserver.connector.inprocess_http.rejection_threads`	Specify an integer from 0 to 1023 for the number of request-processing threads that are denied access. The value to be specified must be lesser than the maximum number of request-processing threads (value specified in `webserver.connector.inprocess_http.max_connections`). If a value greater than the maximum number of connections with the Web client is set, a message is output and a value that is 1 less than the maximum number of connections with the Web client is set as the number of request-processing threads that are denied access. If you specify a non-numeric value, a numeric value outside the range, a null character string or a whitespace#2, a message is output and the default value will be set.	`1`	--	
`webserver.connector.inprocess_http.response.header.server`	Specify the value of the Server header that is automatically added to the response. If you specify a null character string or a whitespace#2, a message is output and the default value will be set.	`CosminexusComponentContainer`	--	
`webserver.connector.inprocess_http.send_timeout`	Specify an integer from 0 to 3600 for the period until timeout (units: seconds) when a response is sent to the Web client. When you specify `0`, the timeout is not enabled. If you specify a non-numeric value, a numeric value outside the range, a null character string or a whitespace#2, a message is output and the default value will be set.	`300`	--	

Legend:

--: Indicates a version earlier than the version 08-00.

#1

- The valid maximum value depends on the operating platform.

- Some of the requests arriving in the Web server are passed to the Web container, therefore, the maximum number of concurrent connections of the Web server to be set up must be greater than the total size of the default pending queue and the pending queue of each URL group and Web application + maximum number of concurrently executing threads in each Web container.

  In a servlet or JSP performing the database operations, since it is not possible to obtain greater multiplicity than the number of database connections, you need to increase the number of usable database connections, when increasing the concurrently executing number of Web containers.

  When tuning the performance, always consider the following relation and adjust the value of each parameter:

  Maximum-number-of-concurrent-connections-of-Web-server > Total-size-of-the-pending-queues-of-each-URL-group-and-Web-application-and-the-default-pending-queue + Maximum-number-of-concurrently-executing-threads-in-each-Web-container

  Maximum-number-of-concurrently-executing-threads-in-each-Web-container ≥ Number-of-database-connections

  For details on controlling the number of concurrently executed threads in Web containers, see *2.14 Controlling the number of concurrently executed threads in Web containers* in the manual *uCosminexus Application Server Web Container Functionality Guide*.

  For details about the number of concurrent connections used for processing in the Web server, reference the manual of the Web server.

#2

Whitespace imply single-byte spaces, tabs, `LF(0x0a)`, `CR(0x0d)` or `FF(0x0c)`.

#3

This is the address associated with `localhost` when the J2EE server started.

(6) Keys beginning with webserver.container

The following table lists the specifiable keys. Note that *Default value* is the value that is assumed when a key is not specified.

VR is the version of Application Server on which the keys are introduced or changed.

Related information is the reference location for information related to the specified key. *uCosminexus Application Server* is omitted from the manual names.

Key name	Contents	Default value	VR	Related information
`webserver.container.ac.logEnabled`	Specify whether to output the trace log for Web container maintenance. If you specify `true`: The trace log will be output. If you specify `false`: The trace log will not be output.	`false`	--	

Legend:

--: Indicates a version earlier than the version 08-00.

12 Files Used by the Smart Composer Functionality

This chapter describes the formats, storage locations, and the functionality of the files used by the Smart Composer functionality, and the keys specifiable in the files. The chapter describes only the content that differs from the recommended mode.

12.1 Parameters applicable to logical Web servers

This section describes the parameters applicable to logical Web servers.

12.1.1 Parameters that set up the redirector action definition for Cosminexus HTTP Server

The following table describes the parameters used for setting up the redirector action definition for Cosminexus HTTP Server:

In the table below, *Default value* means the value that is assumed when the parameter is not specified. *VR* is the version of Application Server on which parameters are introduced or changed. For details about the contents to be specified in param-value corresponding to 'Value of param-name', see *14.2.2 mod_jk.conf (Redirector action definition file for Cosminexus HTTP Server)*.

Note:

To enable the specification of parameters that set up the redirector action definition for Cosminexus HTTP Server when specifying the AllText parameter, include the following code in the setup file:

```
Include "Cosminexus-installation-directory/CC/web/redirector/servers/Logic
al-Web-server/mod_jk.conf"
```

Table 12–1: Parameters that set up the redirector action definition for Cosminexus HTTP Server

Value of param-name	Specifiable value	Default value	VR
`JkConnectTimeout`	Specify the value using an integer from 0 to 3600.	`30`	07-00
`JkGatewayHost`	Specify one of the following values: • IPv4 address • Host name Specify a string within 255 characters using alphanumeric characters, underscore (_), period (`.`), and hyphen (`-`) as a host name.	None	07-50
`JkGatewayHttpsScheme`	Specify one of the following values: • `On` • `Off` For details on this parameter, see *5.10 Notification of gateway information to a Web container*.	`Off`	06-50
`JkGatewayPort`	Specify the value using an integer from 1 to 65535.	None	07-50
`JkLogFileDir`	Specify any string.	`Logs`	06-50
`JkLogFileNum`	Specify the value using an integer from 1 to 64.	`8`	06-50 07-50
`JkLogFileSize`	Specify the value using an integer from 4096 to 2147483647.	`4194304`	06-50 07-50
`JkLogLevel`	Specify one of the following values:	`error`	06-50

Value of param-name	Specifiable value	Default value	VR
	• `emerg` • `error` • `info` • `debug`		
`JkModulePriority`	Specify one of the following values: • `REALLY_FIRST` (corresponds to integer value -10) • `FIRST` (corresponds to integer value 0) • `MIDDLE` (corresponds to integer value 10) • `LAST` (corresponds to integer value 20) • `REALLY_LAST` (corresponds to integer value 30) • Integers from -10 to 30	`FIRST`	07-00
`JkMount`	Specify the definition file in the following format: *<URL-pattern><worker-name>* Use the name specified in the worker definition and worker.list in worker-name. For details on the worker definition and `worker.list`, see *12.1.2 Parameters used for setting up the worker definition*. Note: Do not define the same J2EE server in multiple worker names. The operations cannot be guaranteed if such a worker is specified in JkMount.	In the combined-tier and http-tier# `/*` *J2EE-server-name* Note: Concurrently, *J2EE-server-name* is set up as worker (type: ajp13).	07-50
`JkPrfId`	Specify a string within of 31 alphanumeric characters. If a string beginning with '`TSC`' and '`tsc`' or '`CTM`' and '`ctm`' is specified, an error will occur.	None The value is set automatically except for free-tier.	07-50
`JkRequestRetryCount`	Specify the value using an integer from 1 to 16.	`3`	07-00
`JkSendTimeout`	Specify the value using an integer from 0 to 3600.	`100`	07-00
`JkTraceLog`	Specify one of the following values: • `On` • `Off`	`On`	06-50
`JkTraceLogFileDir`	Specify any string.	`Logs`	06-50
`JkTraceLogFileNum`	Specify the value using an integer from 1 to 64.	`4`	06-50 07-50
`JkTraceLogFileSize`	Specify the value using an integer from 4096 to 2147483647.	`16777216`	06-50 07-50
`JkTranslateBackcompat`	Specify one of the following values:	`Off`	06-50

Value of param-name	Specifiable value	Default value	VR
	• `On` • `Off`		

Note:

If you specify multiple keys, the value specified last will be applied.

#

If this parameter is omitted in combined-tier and http-tier, values are set up in the related parameters as follows:

```
<param>
<param-name>JkMount</param-name>
<param-value>/* (J2EE server name)<param-value>
</param>
<param>
<param-name>worker.list</param-name>
<param-value>(J2EE server name)<param-value>
</param>
<param>
<param-name>worker.(J2EE server name).host</param-name>
<param-value>(host name)<param-value>
</param>
<param>
<param-name>worker.(J2EE server name).port</param-name>
<param-value>(Port number)<param-value>
</param>
<param>
<param-name>worker.(J2EE server name).type</param-name>
<param-value>ajp13</param-value>
</param>
```

12.1.2 Parameters used for setting up the worker definition

The following table describes the parameters used for setting a worker definition.

In the table below, *Default value* means the value that is assumed when the parameter is not specified. *VR* is the version of Application Server on which parameters are introduced or changed. For details about the contents to be specified in param-value corresponding to 'Value of param-name', see *14.2.4 workers.properties (Worker definition file)*.

Note:

To enable the specification of parameters that set up the worker definition when you specify the AllText parameter, include the following code in the setup file:

```
Include "Cosminexus-installation-directory/CC/web/redirector/servers/Logic
al-Web-server/mod_jk.conf"
```

(1) Keys specifiable in the worker definition file

These keys define workers, and the parameters for each worker. If an invalid value is specified for this key, the operations might not execute properly.

Value of param-name	Specifiable value	Default value	VR
`worker.list`	Specify any string.	None	07-50

Value of param-name	Specifiable value	Default value	VR
`worker.`*worker-name*`.`*parameter*	For details about the defined parameters, see *(2) Parameters defined for each worker*.	None	07-50

(2) Parameters defined for each worker

Value of param-name	Specifiable value	Default value	VR
`worker.`*worker-name*`.balanced_workers`	Specify any string.	None	07-50
`worker.`*worker-name*`.cachesize`#	Specify the value using an integer from 1 to 2147483647.	`64`	07-50
`worker.`*worker-name*`.default_worker`	Specify the worker names delimited with commas (`,`). The spaces at the beginning and end are ignored.	None	07-50
`worker.`*worker-name*`.delegate_error_code`	Specify 400 to 417, 422 to 424, 500 to 505, 507, and 510 by using comma (`,`) as the delimiter.	None	07-50
`worker.`*worker-name*`.host`	You can specify the following values: • IPv4 address • Host name • `@myhost` Note: If you specify an IPv4 address or a host name, specify the value of the `<host-name>` tag in the host definition. If you specify a different value, a warning message (KEOS24195-W) is output and an unintended setting might be set.	None	07-50
`worker.`*worker-name*`.lbfactor`#	Specify the value using an integer from 0 to 9999999999.	`1`	07-50
`worker.`*worker-name*`.port`	Specify an integer from 1 to 65535.	None	07-50
`worker.`*worker-name*`.post_data`	Specify a string within 10 numeric characters (0 to 9) and a string with `m`, `M`, `k`, and `K` continuing 0 to one time.	None	07-50
`worker.`*worker-name*`.post_size_workers`	Specify any string.	None	07-50
`worker.`*worker-name*`.receive_timeout`#	Specify the value using an integer from 0 to 3600.	`3600`	07-50
`worker.`*worker-name*`.type`	Specify one of the following values: • ajp13 • lb • post_size_lb	None	07-50

Note:

When you define a new worker, make sure that you define the following parameters:

For details on JkMount, see *12.1.1 Parameters that set up the redirector action definition for Cosminexus HTTP Server*.

- worker.list
- worker.*worker-name*.host
- worker.*worker-name*.port
- worker.*worker-name*.type
- JkMount

#

When set up on combined-tier or http-tier, the default value is applied to the worker with the worker name defined in the J2EE server name. If you want to change the default value, define the worker described in Note above.

12.2 Parameters applicable to logical J2EE servers

This section describes the parameters that are applicable to logical J2EE servers.

The following table shows the reference location for parameters that can be specified and for details.

Table 12–2: Correspondence between the server to be used and references of the parameters to be specified

Server to be used	Parameters to be specified
J2EE server	Parameters used for setting up the user properties for the J2EE server (see *12.2.1*)

12.2.1 Parameters used for setting up the user properties for the J2EE server

This section describes the parameters used for setting up the user properties for the J2EE server.

For details about the contents to be specified in param-value corresponding to 'Value of param-name', see *11.1.1 usrconf.properties (User property file for J2EE servers)*. When you reference the section, read the key as parameter. In the table below, *Default value* means the value that is assumed when the parameter is not specified. *VR* is the version of Application Server on which parameters are introduced or changed.

Note that the specification method of parameters for which "Value in param-name" is specified is different from `usrconf.properties` (user property file for J2EE servers).

The specification format is as follows.

Specification format:

```
<param-name>parameter</param-name>
<param-value>value</param-value>
```

From among the parameters that can be specified for `usrconf.properties` (the user properties file used for J2EE server), use the following format when specifying a parameter that is not written in the tables in this section.

Specification format:

```
<param-name>ex.properties</param-name>
<param-value>parameter=value</param-value>
```

To specify multiple values, specify `<param-value>` multiple times.

Specification format when specifying multiple values:

```
<param-name>ex.properties</param-name>
<param-value>parameter=value</param-value>
<param-value>parameter=value</param-value>
```

(1) Parameters beginning with cosminexus.jpa

The following table describes the parameters beginning with `cosminexus.jpa`. For details about the contents to be specified in param-value corresponding to 'Value of param-name', see *11.1.1(1) Keys beginning with cosminexus.jpa*. When you reference the section, read the key as parameter.

Table 12–3: Parameters beginning with cosminexus.jpa

Value of param-name	Specifiable value	Default value	VR
`cosminexus.jpa.logging.level.operation.`*category*	The following strings can be specified: • `Off` • `Information` • `Detail`	`Off`	08-00
`cosminexus.jpa.exception.logging.sql`	The following strings can be specified: • `Off` • `Information` • `Detail`	`Off`	08-00

(2) Parameters beginning with ejbserver.jpa

The following table describes the parameters beginning with `ejbserver.jpa`. For details about the contents to be specified in param-value corresponding to 'Value of param-name', see *11.1.1(2) Keys beginning with ejbserver.jpa*. When you reference the section, read the key as parameter.

Table 12–4: Parameters beginning with ejbserver.jpa

Value of param-name	Specifiable value	Default value	VR
`ejbserver.jpa.defaultJtaDsName`	Specify any string.	None	08-00
`ejbserver.jpa.defaultNonJtaDsName`	Specify any string.	None	08-00
`ejbserver.jpa.defaultProviderClassName`	Specify any string.	com.hitachi.software.jpa.PersistenceProvider	08-00
`ejbserver.jpa.disable`	The following strings can be specified: • `true` • `false`	`false`	08-50
`ejbserver.jpa.overrideJtaDsName`	Specify any string.	None	08-00
`ejbserver.jpa.overrideNonJtaDsName`	Specify any string.	None	08-00
`ejbserver.jpa.overrideProvider`	Specify any string.	None	08-00
`ejbserver.jpa.emfprop.`*property-key*	Specify any string.	None	08-00

(3) Parameters beginning with ejbserver.logger

The following table describes the parameters beginning with `ejbserver.logger`. For details about the contents to be specified in param-value corresponding to 'Value of param-name', see *11.1.1(3) Keys beginning with ejbserver.logger*. When you reference the section, read the key as parameter.

Table 12–5: Parameters beginning with ejbserver.logger

Value of param-name	Specifiable value	Default value	VR
`ejbserver.logger.channels.define.JPAOperationLogFile.filenum`	Specify the value using an integer from 1 to 16.	`2`	08-00
`ejbserver.logger.channels.define.JPAOperationLogFile.filesize`	Specify an integer from 4096 to 2147483647 (unit: bytes).	`1048576`	08-00
`ejbserver.logger.channels.define.JPAMaintenanceLogFile.filenum`	Specify the value using an integer from 1 to 16.	`2`	08-00
`ejbserver.logger.channels.define.JPAMaintenanceLogFile.filesize`	Specify an integer from 4096 to 2147483647 (unit: bytes).	`1048576`	08-00
`ejbserver.logger.channels.define.WebAccessLogFile.filenum`	Specify the value using an integer from 1 to 16.	`16`	07-50
`ejbserver.logger.channels.define.WebAccessLogFile.filesize`	Specify an integer from 4096 to 2147483647 (unit: bytes).	`4194304`	07-50

(4) Parameters beginning with ejbserver.server

The following table describes the parameters beginning with `ejbserver.server`. For details about the contents to be specified in param-value corresponding to 'Value of param-name', see *11.1.1(4) Keys beginning with ejbserver.server*. When you reference the section, read the key as parameter.

Table 12–6: Parameters beginning with ejbserver.server

Value of param-name	Specifiable value	Default value	VR
`ejbserver.server.eheap.ajp13.enabled`	The following strings can be specified: • `true` • `false`	`true`	08-00

(5) Parameters beginning with webserver.connector

The following table describes the parameters beginning with `webserver.connector`. For details about the contents to be specified in param-value corresponding to 'Value of param-name', see *11.1.1(5) Keys beginning with webserver.connector*. When you reference the section, read the key as parameter.

Table 12–7: Parameters beginning with webserver.connector

Value of param-name	Specifiable value	Default value	VR
`webserver.connector.ajp13.backlog`	Specify the value using an integer from 1 to 2147483647.	`100`	06-50
`webserver.connector.ajp13.bind_host`	The following strings can be specified: • Host name • IPv4 address • `@myhost` Note: If you specify an IPv4 address or a host name, specify the value of the `<host-name>` tag in the host definition. If you specify a different value, a warning message (KEOS24186-W) is output and an unintended setting might be set.	None	07-50
`webserver.connector.ajp13.max_threads`	Specify the value using an integer from 1 to 1024.	`10`	06-50
`webserver.connector.ajp13.port`	Specify the value using an integer from 1 to 65535.	`8007`	06-50 07-00
`webserver.connector.ajp13.receive_timeout`	Specify an integer from 0 to 3600 (unit: seconds).	`600`	06-50 07-00
`webserver.connector.ajp13.send_timeout`	Specify an integer from 0 to 3600 (unit: seconds).	`600`	07-00
`webserver.connector.inprocess_http.backlog`	Specify the value using an integer from 1 to 2147483647.	`511`	07-50
`webserver.connector.inprocess_http.bind_host`	The following strings can be specified: • Host name • IPv4 address • `@myhost` Note If you specify an IPv4 address or a host name, specify the value of the `<host-name>` tag in the host definition. If you specify a different value, a warning message (KEOS24186-W) is output and an unintended setting might be set.	None	07-50
`webserver.connector.inprocess_http.enabled`	The following strings can be specified: • `true` • `false`	`false`	07-50
`webserver.connector.inprocess_http.enabled_methods`	The values that can be specified are as follows: • `GET` • `HEAD` • `POST` • `PUT` • `DELETE` • `OPTIONS` • `TRACE` • `CONNECT` • `PATCH` • `LINK` • `UNLINK` • Asterisk (`*`)	`GET,HEAD,POST,PUT,DELETE,OPTIONS`	07-50

Value of param-name	Specifiable value	Default value	VR
`webserver.connector.inprocess_http.error_custom.`*error-page-customize-definition-name*`.file`	File name	None	07-50
`webserver.connector.inprocess_http.error_custom.`*error-page-customize-definition-name*`.file.content_type`	Specify any string.	`text/html`	07-50
`webserver.connector.inprocess_http.error_custom.`*error-page-customize-definition-name*`.redirect_url`	Specify any string.	None	07-50
`webserver.connector.inprocess_http.error_custom.`*error-page-customize-definition-name*`.request_url`	Specify any string.	`/*`	07-50
`webserver.connector.inprocess_http.error_custom.`*error-page-customize-definition-name*`.status`	Specify the value using an integer from 400 to 599.	None	07-50
`webserver.connector.inprocess_http.error_custom.list`	Specify a string within 32 characters using alphanumeric characters and underscore (_). Also, when specifying multiple values, delimit with commas (`,`).	None	07-50
`webserver.connector.inprocess_http.gateway.host`	Specify one of the following values: • Host name • IPv4 address • `@myhost`	None	07-50
`webserver.connector.inprocess_http.gateway.https_scheme`	The following strings can be specified: • `true` • `false`	`false`	07-50
`webserver.connector.inprocess_http.gateway.port`	Specify the value using an integer from 1 to 65535.	None	07-50
`webserver.connector.inprocess_http.init_threads`	Specify the value using an integer from 1 to 1024.	`10`	07-50
`webserver.connector.inprocess_http.keep_start_threads`	The following strings can be specified: • `true` • `false`	`false`	07-50
`webserver.connector.inprocess_http.limit.max_headers`	Specify the value using an integer from 0 to 32767.	`100`	07-50
`webserver.connector.inprocess_http.limit.max_request_body`	Specify an integer from -1 to 2147483647 (unit: bytes).	`-1`	07-50
`webserver.connector.inprocess_http.limit.max_request_header`	Specify an integer from 7 to 65536 (unit: bytes).	`16384`	07-50
`webserver.connector.inprocess_http.limit.max_request_line`	You can specify the following values (unit: bytes): • -1 • 7-8190	`8190`	07-50

Value of param-name	Specifiable value	Default value	VR
`webserver.connector.inprocess_http.max_connections`	Specify the value using an integer from 1 to 1024.	`100`	07-50
`webserver.connector.inprocess_http.max_execute_threads`	Specify the value using an integer from 1 to 1024.	`10`	07-50
`webserver.connector.inprocess_http.max_spare_threads`	Specify the value using an integer from 1 to 1024.	`20`	07-50
`webserver.connector.inprocess_http.min_spare_threads`	Specify the value using an integer from 1 to 1024.	`5`	07-50
`webserver.connector.inprocess_http.permitted.hosts`	Specify one of the following values: • Host name • IPv4 address • `@myhost` • Asterisk (`*`)	`*`	07-50
`webserver.connector.inprocess_http.persistent_connection.max_connections`	Specify the value using an integer from 0 to 1024.	`100`	07-50
`webserver.connector.inprocess_http.persistent_connection.max_requests`	Specify the value using an integer from 0 to 2147483647.	`100`	07-50
`webserver.connector.inprocess_http.persistent_connection.timeout`	Specify an integer from 0 to 3600 (unit: seconds).	`3`	07-50
`webserver.connector.inprocess_http.port`	Specify the value using an integer from 1 to 65535.	`80`	07-50
`webserver.connector.inprocess_http.receive_timeout`	Specify an integer from 0 to 3600 (unit: seconds).	`300`	07-50
`webserver.connector.inprocess_http.redirect.`*redirect-definition-name*`.file`	Specify the file name.	None	07-50
`webserver.connector.inprocess_http.redirect.`*redirect-definition-name*`.file.content_type`	Specify any string.	`text/html`	07-50
`webserver.connector.inprocess_http.redirect.`*redirect-definition-name*`.redirect_url`	Specify any string.	None	07-50
`webserver.connector.inprocess_http.redirect.`*redirect-definition-name*`.request_url`	Specify any string.	None	07-50
`webserver.connector.inprocess_http.redirect.`*redirect-definition-name*`.status`	The values that can be specified are as follows: • `200` • `300` • `301` • `302` • `303` • `305` • `307`	`302`	07-50

Value of param-name	Specifiable value	Default value	VR
`webserver.connector.inprocess_http.redirect.list`	Specify a string within 32 characters using alphanumeric characters and underscore (_). Also, when specifying multiple values, delimit with commas (`,`).	None	07-50
`webserver.connector.inprocess_http.rejection_threads`	Specify the value using an integer from 0 to 1023.	`1`	07-50
`webserver.connector.inprocess_http.response.header.server`	Specify any string.	Cosminexus ComponentContainer	07-50
`webserver.connector.inprocess_http.send_timeout`	Specify an integer from 0 to 3600 (unit: seconds).	`300`	07-50

(6) Parameters beginning with webserver.container

The following table describes the parameters beginning with `webserver.container`. For details about the contents to be specified in param-value corresponding to 'Value of param-name', see *11.1.1(6) Keys beginning with webserver.container*. When you reference the section, read the key as parameter.

Table 12–8: Parameters beginning with webserver.container

Value of param-name	Specifiable value	Default value	VR
`webserver.container.ac.logEnabled`	The following strings can be specified: • `true` • `false`	`false`	06-50

13 Files Used with JPA

This chapter describes the formats, storage locations, and functionality of files used in Cosminexus JPA Provider and the keys specifiable in these files.

13.1 List of files used in Cosminexus JPA Provider

The following table lists and describes the files used in Cosminexus JPA Provider:

Table 13–1: List of files used in Cosminexus JPA Provider

File name	Overview	Reference
`persistence.xml`	This file is used for setting the persistence unit information of Cosminexus JPA.	*13.2*
`O/R mapping files`	This file is used for setting the O/R mapping information.	*13.3*

13.2 persistence.xml

The following table describes the configuration of persistence.xml:

Tag name	Occurrence pattern	Description
`<persistence>`	Once	Indicates the root tag.
`<persistence-unit>`	0 or more times	Defines the persistence unit.
`<description>`	0 or once	Describes the persistence unit.
`<provider>`	0 or once	Specifies the implementation class name of `javax.persistence.spi.PersistenceProvider`.
`<jta-data-source>`	0 or once	Specifies the references for the data source corresponding to the JTA transaction.
`<non-jta-data-source>`	0 or once	Specifies the references for the data source not corresponding to the JTA transaction.
`<mapping-file>`	0 or more times	Specifies the O/R mapping file.
`<jar-file>`	0 or more times	Codes a JAR file name containing the `entity` class, `embeddable` class, and `mappedsuper` class.
`<class>`	0 or more times	Codes the `entity` class, `embeddable` class, and `mappedsuper` class.
`<exclude-unlisted-classes>`	0 or once	Specifies the `Persistence` class.
`<properties>`	0 or once	Defines the Cosminexus JPA Provider-specific properties.
`<property>`	0 or more times	Defines various properties.

For details about the respective tags, see *13.2.1 Details of persistence.xml*.

13.2.1 Details of persistence.xml

(1) <persistence>

The `<persistence>` tag is the root tag indicating the start of definition for the persistence unit.

You must set the `xmlns` element that specifies the XML namespace in the `<persistence>` tag.

The following table lists the specifiable attributes:

Table 13–2: Attributes of <persistence>

Attribute name	Type	Optional/Required	Description
`version`	persistence:versionType	Required	Specifies the XML schema version "1.0".

(2) <persistence-unit>

The `<persistence-unit>` tag defines the persistence unit. The following table lists the specifiable attributes:

Table 13–3: Attributes of <persistence-unit>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Specifies the name of the persistence unit. The name that will be specified in the `name` attribute must be unique in the persistence unit packaging range. If a persistence unit is defined with duplicate names, the operations of Cosminexus JPA Provider cannot be guaranteed. If persistence units with the same names are defined in the Java EE environment, a warning message is output by the container. In the Java EE environment, the value specified in the `name` attribute should not be a null character. If a null character is specified, an exception occurs in the container.

`transaction-type`	persistence:persistence-unit-transaction-type	Optional	Specifies the transaction that will be used by the EntityManager.
			`JTA`
			The JTA transaction will be used.
			`RESOURCE_LOCAL`
			The transaction will be managed uniquely without using the JTA transaction.
			If the value of the `transaction-type` attribute is not specified, the default value '`JTA`' will be applied.
			For details on attributes that can be specified in the Java EE environment, see *8.8.1 Attributes specified in the <persistence-unit> tag*.

(3) <description>

Describes the persistence unit.

(4) <provider>

Specifies the implementation class name of `javax.persistence.spi.PersistenceProvider`.

When using Cosminexus JPA Provider, you specify `com.hitachi.software.jpa.PersistenceProvider`. You code this tag for explicit specification such as when other JPA providers exist. Note that when the value is not specified, the tag behavior depends on the container.

Precautions

If you insert a space in the middle of the `<provider>` tag, the processing is same as the case when the element is not specified.

(5) <jta-data-source>

Specifies the references for the data source corresponding to the JTA transaction.

Specify this tag when the value specified for `transaction-type` in the `<persistence-unit>` tag is JTA. Note that when the value is not specified, the tag behavior depends on the container.

(6) <non-jta-data-source>

Specifies the references for the data source not corresponding to the JTA transaction. Specify this tag when the value specified for `transaction-type` in the `<persistence-unit>` tag is `RESOURCE_LOCAL`.

If a value specified for `transaction-type` is JTA, the value is ignored in Cosminexus JPA Provider even if that value is specified for the `<non-jta-data-source>` tag. When the value is not specified, the tag behavior depends on the container.

(7) <mapping-file>

Specifies the O/R mapping file.

You must store the specified file at the location specified in the class path. The `<mapping-file>` tag need not be coded when the O/R mapping file is not used or when `orm.xml` is deployed at a defined location and used.

If the specified file is not found, the application will fail to start.

(8) <jar-file>

The `<jar-file>` tag codes a JAR file name that includes the `entity` class, `embeddable` class, and `mappedsuper` class. Specify the JAR file path using the relative path from the root of the persistence unit.

If the specified file is not found, the tag behavior depends on the container.

(9) <class>

Codes the `entity` class, `embeddable` class, and `mappedsuper` class.

If the specified class is not found, the tag behavior depends on the container.

Note that Cosminexus JPA Provider does not implement the check to verify whether the value specified in the `<class>` tag is the `entity` class, `embeddable` class, and `mappedsuper` class. Therefore, if you specify a class other than the `entity` class, `embeddable` class, and `mappedsuper` class, the operation is performed without throwing an exception.

(10) <exclude-unlisted-classes>

The `<exclude-unlisted-classes>` tag defines the Persistence class.

The specifiable values and the behavior when the values are specified are as follows:

`true`

Only the classes explicitly specified by the `class` element, the `jar-file` element, and the `mapping-file` element are handled as the Persistence classes.

`false`

If the `exclude-unlisted-class` element is not specified, the class files under the persistence unit root will be searched to find out if the class is a JPA target class.

(11) <properties>

Defines the Cosminexus JPA Provider-specific properties. You specify the `property` element under this element to define the property.

(12) <property>

Defines various properties.

For details about the properties, see *13.2.2 Cosminexus JPA Provider-specific properties that can be specified in the <property> tag*.

The following table lists the specifiable attributes:

Table 13–4: Attributes of <property>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Property name.
`value`	xsd:string	Required	Property value.

13.2.2 Cosminexus JPA Provider-specific properties that can be specified in the <property> tag

This section describes the Cosminexus JPA Provider-specific properties specified in the `<property>` tag of `persistence.xml`.

Cosminexus JPA Provider does not have properties that can be specified as arguments for `createEntityManager()` of `EntityManagerFactory`. Note that even if the attribute values of the `persistence.xml` file properties are numeric, you make sure that the value is enclosed in quotation marks (double quotations or single quotations).

Important note

In Cosminexus JPA Provider, you cannot specify a property beginning with `javax` included in JPA specifications, within the `<property>` tag. The operations might not execute normally if a property beginning with `javax` included in JPA specifications is specified in the `<property>` tag of `persistence.xml`.

Also, you do not specify a property beginning with `javax` included in JPA specifications as a system property. The properties beginning with `javax` included in JPA specifications are as follows:

- `javax.persistence.transactionType`
- `javax.persistence.jtaDataSource`
- `javax.persistence.nonJtaDataSource`
- `javax.persistence.provider`

The following is the description related to the Cosminexus JPA Provider-specific properties:

You specify these properties in the `<property>` tag of the `persistence.xml` file.

If a value outside the specifiable range is set for a property, an exception occurs during the deployment. Also, the value is case sensitive.

The following table describes the Cosminexus JPA Provider-specific properties:

Table 13–5: Cosminexus JPA Provider-specific properties

Property name	Contents	Specifiable value	Default
`cosminexus.jpa.cache.size.<ENTITY>`	Specify the initial size to be set for the entity cache specified in `<ENTITY>`. If a cache type is `Full`, set the initial size for the entity cache. If a cache type is `HardWeak` or `SoftWeak`, set the maximum size for the entity cache. Specify the cache size using the maximum ID count (record count) of the entities invoked in all as a standard.	0 to 2147483647	`1000`
`cosminexus.jpa.cache.size.default`	Specify the default cache size used when the entity is cached. If a cache type is `Full`, set the initial size for the entity cache. If a cache type is `HardWeak` or `SoftWeak`, set the maximum size for the entity cache. Specify the cache size using the maximum ID count (record count) of the entities invoked in all as a standard.	0 to 2147483647	`1000`
`cosminexus.jpa.cache.type.<ENTITY>`	Specify the entity cache type specified in `<ENTITY>`.	You can specify the following strings: • `Full` • `Weak` • `HardWeak` • `SoftWeak` • `NONE`	`SoftWeak`
`cosminexus.jpa.cache.type.default`	Specify the default cache type.	You can specify the following strings: • `Full` • `Weak` • `HardWeak` • `SoftWeak` • `NONE`	`SoftWeak`
`cosminexus.jpa.target-database`	Specify the name of a database you want to connect to. To classify the database-specific processing, the class implementing the database-specific part is read according to the specified value. When `Auto` is specified, the database is automatically identified from the resource information specified for `<jta-data-source>` or	The following strings can be specified: • `Auto` • `HiRDB` • `Oracle`	`Auto`

Property name	Contents	Specifiable value	Default
	`<non-jta-data-source>` in `persistence.xml` when a request is sent to the CJPA provider for the first time. For this reason, the processing time is slightly longer than when the database name is specified. Normally, Hitachi recommends you to specify the name of the database without using `Auto`.		

13.3 O/R mapping files

The following table describes the configuration of the O/R mapping files:

Tag name	Occurrence pattern	Description
`<entity-mappings>`	Once	Indicates the root tag.
`<description>`	0 or once	Adds the description.
`<persistence-unit-metadata>`	0 or once	Specifies the definition related to the entire `PersistenceUnit`.
`<xml-mapping-metadata-complete>`	0 or once	Specifies whether to control the mapping metadata of the persistence unit.
`<persistence-unit-defaults>`	0 or once	Specifies the default value of the persistence unit.
`<schema>`	0 or once	Defines the schema.
`<catalog>`	0 or once	Defines the catalog.
`<access>`	0 or once	Specifies the access type.
`<cascade-persist>`	0 or once	Adds the cascade persistence option.
`<entity-listeners>`	0 or once	Defines the default entity listener of the persistence unit.
`<entity-listener>`	0 or more times	Specifies the entity listener.
`<pre-persist>`	0 or once	Specifies the lifecycle callback method.
`<post-persist>`	0 or once	Specifies the lifecycle callback method.
`<pre-remove>`	0 or once	Specifies the lifecycle callback method.
`<post-remove>`	0 or once	Specifies the lifecycle callback method.
`<pre-update>`	0 or once	Specifies the lifecycle callback method.
`<post-update>`	0 or once	Specifies the lifecycle callback method.
`<post-load>`	0 or once	Specifies the lifecycle callback method.
`<package>`	0 or once	Specifies the class package described in the elements and attributes within the same mapping file.
`<schema>`	0 or once	Defines the schema.
`<catalog>`	0 or once	Defines the catalog.
`<access>`	0 or once	Defines the access method.
`<sequence-generator>`	0 or more times	Adds the sequence generator.
`<table-generator>`	0 or more times	Defines the table generator.
`<unique-constraint>`	0 or more times	Adds a unique constraint to DDL.
`<column-name>`	One or more times	Specifies the name of the column in which the unique constraint will be added.
`<named-query>`	0 or more times	Defines a named query.
`<query>`	Once	Specifies the query string.

Tag name	Occurrence pattern	Description
<hint>	0 or more times	Adds a hint to the query.
<named-native-query>	0 or more times	Defines the named native query.
<query>	Once	Specifies the query string.
<hint>	0 or more times	Adds a hint to the query.
<sql-result-set-mapping>	0 or more times	Defines the SQL result set mapping.
<entity-result>	0 or more times	Specifies the `entity` class used for mapping the native SQL query result.
<field-result>	0 or more times	Specifies the field used for mapping the native SQL query result.
<column-result>	0 or more times	Specifies the column used for mapping the native SQL query result.
<mapped-superclass>	0 or more times	Defines the mapped superclass of the persistence unit.
<description>	0 or once	Adds the description for the mapped superclass of the persistence unit.
<id-class>	0 or once	Overwrites `@IdClass` specified in the mapped superclass.
<exclude-default-listeners>	0 or once	Defines whether to control the default entity listener of the mapped superclass and the sub class.
<exclude-superclass-listeners>	0 or once	Defines whether to control the superclass listener of the mapped superclass and the sub class.
<entity-listeners>	0 or once	Specifies the callback listener class.
<entity-listener>	0 or more times	Specifies the entity listener.
<pre-persist>	0 or once	Specifies the lifecycle callback method.
<post-persist>	0 or once	
<pre-remove>	0 or once	
<post-remove>	0 or once	
<pre-update>	0 or once	
<post-update>	0 or once	
<post-load>	0 or once	
<pre-persist>	0 or once	Defines the lifecycle callback method using the corresponding annotations in the mapped superclass.
<post-persist>	0 or once	
<pre-remove>	0 or once	
<post-remove>	0 or once	
<pre-update>	0 or once	
<post-update>	0 or once	
<post-load>	0 or once	

Tag name	Occurrence pattern	Description
`<attributes>`	0 or once	The element is not defined.
`<id>`	0 or more times#1	Overwrites the mapping specified in fields and properties.
`<column>`	0 or once	Specifies the column mapping for the properties of the Persistent field.
`<generated-value>`	0 or once	Specifies the strategy for generating the primary key value.
`<temporal>`	0 or once	Specified when mapping to the DATE, TIME, and TIMESTAMP type.
`<table-generator>`	0 or once	Adds the table generator.
`<unique-constraint>`	0 or more times	Adds a unique constraint to DDL.
`<column-name>`	One or more times	Specifies the name of the column in which the unique constraint will be added.
`<sequence-generator>`	0 or once	Adds the sequence generator.
`<embedded-id>`	0 or once#1	Overwrites the mapping specified in fields and properties.
`<attribute-override>`	0 or more times	Overwrites the mapping of properties and fields.
`<column>`	Once	Specifies the column mapping for the properties of the Persistent field.
`<basic>`	0 or more times	Overwrites the mapping specified in fields and properties.
`<column>`	0 or once	Specifies the column mapping for the properties of the Persistent field.
`<lob>`	0 or once#2	Specified when mapping to the Lob type.
`<temporal>`	0 or once#2	Specified when mapping to the DATE, TIME, and TIMESTAMP type.
`<enumerated>`	0 or once#2	Specified when mapping to the enumeration type.
`<version>`	0 or more times	Overwrites the mapping specified in fields and properties.
`<column>`	0 or once	Specifies the column mapping for the properties of the Persistent field.
`<temporal>`	0 or once	Specified when mapping to the DATE, TIME, and TIMESTAMP type.
`<many-to-one>`	0 or more times	Overwrites the mapping specified in fields and properties.
`<join-column>`	0 or more times#3	Specifies the external key column of the join table corresponding to the owner-side entity.
`<join-table>`	0 or once#3	Specifies the join table to be used in many-to-many and the unilateral one-to-many relationships.

Tag name	Occurrence pattern	Description
`<join-column>`	0 or more times	Specifies the external key column of the join table corresponding to the owner-side entity.
`<inverse-join-column>`	0 or more times	Specifies the external key column of the join table corresponding to the owned-side entity.
`<unique-constraint>`	0 or more times	Adds a unique constraint to DDL.
`<column-name>`	One or more times	Specifies the name of the column in which the unique constraint will be added.
`<cascade>`	0 or once	Specifies the operations that you can cascade.
`<cascade-all>`	0 or once	Cascades all the operations.
`<cascade-persist>`	0 or once	Cascades the persist operation.
`<cascade-merge>`	0 or once	Cascades the merge operation.
`<cascade-remove>`	0 or once	Cascades the remove operation.
`<cascade-refresh>`	0 or once	Cascades the refresh operation.
`<one-to-many>`	0 or more times	Overwrites the mapping specified in fields and properties.
`<order-by>`	0 or once	Specifies the order to be applied when maintaining a relation in the collection.
`<map-key>`	0 or once	Specifies the map key as the Map type relation.
`<join-table>`	0 or once#4	Specifies the join table to be used in many-to-many and the unilateral one-to-many relationships.
`<join-column>`	0 or more times	Specifies the external key column of the join table corresponding to the owner-side entity.
`<inverse-join-column>`	0 or more times	Specifies the external key column of the join table corresponding to the owned-side entity.
`<unique-constraint>`	0 or more times	Adds a unique constraint to DDL.
`<column-name>`	One or more times	Specifies the name of the column in which the unique constraint will be added.
`<join-column>`	0 or more times#4	Specifies the external key column of the join table corresponding to the owner-side entity.
`<cascade>`	0 or once	Specifies the operations that you can cascade.
`<cascade-all>`	0 or once	Cascades all the operations.
`<cascade-persist>`	0 or once	Cascades the persist operation.
`<cascade-merge>`	0 or once	Cascades the merge operation.
`<cascade-remove>`	0 or once	Cascades the remove operation.
`<cascade-refresh>`	0 or once	Cascades the refresh operation.
`<one-to-one>`	0 or more times	Overwrites the mapping specified in fields and properties.
`<primary-key-join-column>`	0 or more times#5	Specifies the primary key column used as external key to JOIN with other tables.

Tag name	Occurrence pattern	Description
`<join-column>`	0 or more times#5	Specifies the external key column of the join table corresponding to the owner-side entity.
`<join-table>`	0 or once#5	Specifies the join table to be used in many-to-many and the unilateral one-to-many relationships.
`<join-column>`	0 or more times	Specifies the external key column of the join table corresponding to the owner-side entity.
`<inverse-join-column>`	0 or more times	Specifies the external key column of the join table corresponding to the owned-side entity.
`<unique-constraint>`	0 or more times	Adds a unique constraint to DDL.
`<column-name>`	One or more times	Specifies the name of the column in which the unique constraint will be added.
`<cascade>`	0 or once	Specifies the operations that you can cascade.
`<cascade-all>`	0 or once	Cascades all the operations.
`<cascade-persist>`	0 or once	Cascades the persist operation.
`<cascade-merge>`	0 or once	Cascades the merge operation.
`<cascade-remove>`	0 or once	Cascades the remove operation.
`<cascade-refresh>`	0 or once	Cascades the refresh operation.
`<many-to-many>`	0 or more times	Overwrites the mapping specified in fields and properties.
`<order-by>`	0 or once	Specifies the order to be applied when maintaining a relation in the collection.
`<map-key>`	0 or once	Specifies the map key as the Map type relation.
`<join-table>`	0 or once	Specifies the join table to be used in many-to-many and the unilateral one-to-many relationships.
`<join-column>`	0 or more times	Specifies the external key column of the join table corresponding to the owner-side entity.
`<inverse-join-column>`	0 or more times	Specifies the external key column of the join table corresponding to the owned-side entity.
`<unique-constraint>`	0 or more times	Adds a unique constraint to DDL.
`<column-name>`	One or more times	Specifies the name of the column in which the unique constraint will be added.
`<cascade>`	0 or once	Specifies the operations that you can cascade.
`<cascade-all>`	0 or once	Cascades all the operations.
`<cascade-persist>`	0 or once	Cascades the persist operation.
`<cascade-merge>`	0 or once	Cascades the merge operation.
`<cascade-remove>`	0 or once	Cascades the remove operation.
`<cascade-refresh>`	0 or once	Cascades the refresh operation.

Tag name						Occurrence pattern	Description
			`<embedded>`			0 or more times	Overwrites the mapping specified in fields and properties.
				`<attribute-override>`		0 or more times	Overwrites the mapping of properties and fields.
					`<column>`	Once	Specifies the column mapping for the properties of the Persistent field.
			`<transient>`			0 or more times	Overwrites the mapping specified in fields and properties.
	`<entity>`					0 or more times	Defines the entities of the persistence unit.
		`<description>`				0 or once	Adds the description for the entities of the persistence unit.
		`<table>`				0 or once	Overwrites `@Table` (including default values) of the `entity` class.
			`<unique-constraint>`			0 or more times	Adds a unique constraint to DDL.
				`<column-name>`		One or more times	Specifies the name of the column in which the unique constraint will be added.
		`<secondary-table>`				0 or more times	Overwrites all `@SecondaryTable` and `@SecondaryTables` (including default values) of the `entity` class.
			`<primary-key-join-column>`			0 or more times	Overwrites all `@PrimaryKeyJoinColumn` and `@PrimaryKeyJoinColumns` (including default values) of the `entity` class.
			`<unique-constraint>`			0 or more times	Adds a unique constraint to DDL.
				`<column-name>`		One or more times	Specifies the name of the column in which the unique constraint will be added.
		`<primary-key-join-column>`				0 or more times	Overwrites all `@PrimaryKeyJoinColumn` and `@PrimaryKeyJoinColumns` (including default values) of the `entity` class.
		`<id-class>`				0 or once	Overwrites `@IdClass` specified in the `entity` class.
		`<inheritance>`				0 or once	Overwrites `@Inheritance` (including default values) of the `entity` class.
		`<discriminator-value>`				0 or once	Overwrites `@Discriminator`Value (including default values) of the `entity` class.
		`<discriminator-column>`				0 or once	Overwrites `@DiscriminatorColumn` (including default values) of the `entity` class.
		`<sequence-generator>`				0 or once	Specifies the settings for the sequence generator that creates the primary key.
		`<table-generator>`				0 or once	Specifies the settings for the generator that creates the primary key.
			`<unique-constraint>`			0 or more times	Adds a unique constraint to DDL.

Tag name					Occurrence pattern	Description
				`<column-name>`	One or more times	Specifies the name of the column in which the unique constraint will be added.
		`<named-query>`			0 or more times	Defines a named query.
			`<query>`		Once	Specifies the query string.
			`<hint>`		0 or more times	Adds a hint to the query.
		`<named-native-query>`			0 or more times	Defines the named native query.
			`<query>`		Once	Specifies the query string.
			`<hint>`		0 or more times	Adds a hint to the query.
		`<sql-result-set-mapping>`			0 or more times	Defines the mapping of the SQL results.
			`<entity-result>`		0 or more times	Specifies the `entity` class used for mapping the native SQL query result.
				`<field-result>`	0 or more times	Specifies the field used for mapping the native SQL query result.
			`<column-result>`		0 or more times	Specifies the column used for mapping the native SQL query result.
		`<exclude-default-listeners>`			0 or once	Controls the default entity listeners of the `entity` class and the sub class.
		`<exclude-superclass-listeners>`			0 or once	Controls the superclass listeners of the `entity` class and the sub class.
		`<entity-listeners>`			0 or once	Overwrites `@EntityListeners` of the `entity` class.
			`<entity-listener>`		0 or more times	Specifies the entity listener.
				`<pre-persist>`	0 or once	Overwrites the definition of the lifecycle callback method according to the corresponding annotations, in the mapped superclass.
				`<post-persist>`	0 or once	
				`<pre-remove>`	0 or once	
				`<post-remove>`	0 or once	
				`<pre-update>`	0 or once	
				`<post-update>`	0 or once	
				`<post-load>`	0 or once	
		`<pre-persist>`			0 or once	Overwrites the definition of the lifecycle callback method according to the corresponding annotations, in the `entity` class.
		`<post-persist>`			0 or once	
		`<pre-remove>`			0 or once	
		`<post-remove>`			0 or once	
		`<pre-update>`			0 or once	
		`<post-update>`			0 or once	
		`<post-load>`			0 or once	
		`<attribute-override>`			0 or more times	Added to the value defined in `@AttributeOverride` or

Tag name							Occurrence pattern	Description
								`@AttributeOverrides` of the `entity` class.
			`<column>`				Once	Specifies the column mapping for the properties of the Persistent field.
		`<association-override>`					0 or more times	Added to the value defined in `@AssociationOverride` or `@AssociationOverrides` of the `entity` class.
			`<join-column>`				One or more times	Specifies the external key column of the join table corresponding to the owner-side entity.
		`<attributes>`					0 or once	The element is not defined.
			`<id>`				0 or more times[#1]	Overwrites the mapping specified in fields and properties.
				`<column>`			0 or once	Specifies the column mapping for the properties of the Persistent field.
				`<generated-value>`			0 or once	Specifies the strategy for generating the primary key value.
				`<temporal>`			0 or once	Specified when mapping to the DATE, TIME, and TIMESTAMP type.
				`<table-generator>`			0 or once	Adds the table generator.
					`<unique-constraint>`		0 or more times	Adds a unique constraint to DDL.
						`<column-name>`	One or more times	Specifies the name of the column in which the unique constraint will be added.
				`<sequence-generator>`			0 or once	Specifies the settings for the sequence generator that creates the primary key.
			`<embedded-id>`				0 or once[#1]	Overwrites the mapping specified in fields and properties.
				`<attribute-override>`			0 or more times	Added to the value defined in `@AttributeOverride` or `@AttributeOverrides` of the `entity` class.
					`<column>`		Once	Specifies the column mapping for the properties of the Persistent field.
			`<basic>`				0 or more times	Overwrites the mapping specified in fields and properties.
				`<column>`			0 or once	Specifies the column mapping for the properties of the Persistent field.
				`<lob>`			0 or once[#2]	Specified when mapping to the Lob type.
				`<temporal>`			0 or once	Specified when mapping to the DATE, TIME, and TIMESTAMP type.
				`<enumerated>`			0 or once[#2]	Specified when mapping to the enumeration type.
			`<version>`				0 or more times	Overwrites the mapping specified in fields and properties.

Tag name	Occurrence pattern	Description
`<column>`	0 or once	Specifies the column mapping for the properties of the Persistent field.
`<temporal>`	0 or once	Specified when mapping to the DATE, TIME, and TIMESTAMP type.
`<many-to-one>`	0 or more times	Overwrites the mapping specified in fields and properties.
`<join-column>`	0 or more times[#4]	Specifies the external key column of the join table corresponding to the owner-side entity.
`<join-table>`	0 or once[#4]	Specifies the join table to be used in many-to-many and the unilateral one-to-many relationships.
`<join-column>`	0 or more times	Specifies the external key column of the join table corresponding to the owner-side entity.
`<inverse-join-column>`	0 or more times	Specifies the external key column of the join table corresponding to the owned-side entity.
`<unique-constraint>`	0 or more times	Adds a unique constraint to DDL.
`<column-name>`	One or more times	Specifies the name of the column in which the unique constraint will be added.
`<cascade>`	0 or once	Specifies the operations that you can cascade.
`<cascade-all>`	0 or once	Cascades all the operations.
`<cascade-persist>`	0 or once	Cascades the persist operation.
`<cascade-merge>`	0 or once	Cascades the merge operation.
`<cascade-remove>`	0 or once	Cascades the remove operation.
`<cascade-refresh>`	0 or once	Cascades the refresh operation.
`<one-to-many>`	0 or more times	Overwrites the mapping specified in fields and properties.
`<order-by>`	0 or once	Specifies the order to be applied when maintaining a relation in the collection.
`<map-key>`	0 or once	Specifies the map key as the Map type relation.
`<join-table>`	0 or once[#3]	Specifies the join table to be used in many-to-many and the unilateral one-to-many relationships.
`<join-column>`	0 or more times	Specifies the external key column of the join table corresponding to the owner-side entity.
`<inverse-join-column>`	0 or more times	Specifies the external key column of the join table corresponding to the owned-side entity.
`<unique-constraint>`	0 or more times	Adds a unique constraint to DDL.
`<column-name>`	One or more times	Specifies the name of the column in which the unique constraint will be added.
`<join-column>`	0 or more times[#3]	Specifies the external key column of the join table corresponding to the owner-side entity.
`<cascade>`	0 or once	Specifies the operations that you can cascade.

Tag name							Occurrence pattern	Description
					`<cascade-all>`		0 or once	Cascades all the operations.
					`<cascade-persist>`		0 or once	Cascades the persist operation.
					`<cascade-merge>`		0 or once	Cascades the merge operation.
					`<cascade-remove>`		0 or once	Cascades the remove operation.
					`<cascade-refresh>`		0 or once	Cascades the refresh operation.
			`<one-to-one>`				0 or more times	Overwrites the mapping specified in fields and properties.
				`<primary-key-join-column>`			0 or more times[#5]	Overwrites all `@PrimaryKeyJoinColumn` and `@PrimaryKeyJoinColumns` (including default values) of the `entity` class.
				`<join-column>`			0 or more times[#5]	Specifies the external key column of the join table corresponding to the owner-side entity.
				`<join-table>`			0 or once[#5]	Specifies the join table to be used in many-to-many and the unilateral one-to-many relationships.
					`<join-column>`		0 or more times	Specifies the external key column of the join table corresponding to the owner-side entity.
					`<inverse-join-column>`		0 or more times	Specifies the external key column of the join table corresponding to the owned-side entity.
					`<unique-constraint>`		0 or more times	Adds a unique constraint to DDL.
						`<column-name>`	One or more times	Specifies the name of the column in which the unique constraint will be added.
				`<cascade>`			0 or once	Specifies the operations that you can cascade.
					`<cascade-all>`		0 or once	Cascades all the operations.
					`<cascade-persist>`		0 or once	Cascades the persist operation.
					`<cascade-merge>`		0 or once	Cascades the merge operation.
					`<cascade-remove>`		0 or once	Cascades the remove operation.
					`<cascade-refresh>`		0 or once	Cascades the refresh operation.
			`<many-to-many>`				0 or more times	Overwrites the mapping specified in fields and properties.
				`<order-by>`			0 or once	Specifies the order to be applied when maintaining a relation in the collection.
				`<map-key>`			0 or once	Specifies the map key as the Map type relation.
				`<join-table>`			0 or once	Specifies the join table to be used in many-to-many and the unilateral one-to-many relationships.
					`<join-column>`		0 or more times	Specifies the external key column of the join table corresponding to the owner-side entity.
					`<inverse-join-column>`		0 or more times	Specifies the external key column of the join table corresponding to the owned-side entity.

Tag name							Occurrence pattern	Description
					`<unique-constraint>`		0 or more times	Adds a unique constraint to DDL.
						`<column-name>`	One or more times	Specifies the name of the column in which the unique constraint will be added.
				`<cascade>`			0 or once	Specifies the operations that you can cascade.
					`<cascade-all>`		0 or once	Cascades all the operations.
					`<cascade-persist>`		0 or once	Cascades the persist operation.
					`<cascade-merge>`		0 or once	Cascades the merge operation.
					`<cascade-remove>`		0 or once	Cascades the remove operation.
					`<cascade-refresh>`		0 or once	Cascades the refresh operation.
			`<embedded>`				0 or more times	Overwrites the mapping specified in fields and properties.
				`<attribute-override>`			0 or more times	Added to the value defined in `@AttributeOverride` or `@AttributeOverrides` of the `entity` class.
					`<column>`		Once	Specifies the column mapping for the properties of the Persistent field.
			`<transient>`				0 or more times	Overwrites the mapping specified in fields and properties.
	`<embeddable>`						0 or more times	Defines the `embeddable` class of the persistence unit.
		`<description>`					0 or once	Adds the description for the `embeddable` class of the persistence unit.
		`<attributes>`					0 or once	The element is not defined.
			`<basic>`				0 or more times	Overwrites the mapping specified in fields and properties.
				`<column>`			0 or once	Specifies the column mapping for the properties of the Persistent field.
				`<lob>`			0 or once#2	Specified when mapping to the Lob type.
				`<temporal>`			0 or once#2	Specified when mapping to the DATE, TIME, and TIMESTAMP type.
				`<enumerated>`			0 or once#2	Specified when mapping to the enumeration type.
			`<transient>`				0 or more times	Overwrites the mapping specified in fields and properties.

#1

Specify any one among the `<id>` tag and the `<embedded-id>` tag.

#2

Specify any one among the `<lob>` tag, the `<temporal>` tag, and the `<enumerated>` tag.

#3

Specify any one among the `<join-column>` tag and the `<join-table>` tag.

#4

Specify any one among the `<join-table>` tag and the `<join-column>` tag.

#5

Specify any one among the `<primary-key-join-column>` tag, the `<join-column>` tag, and the `<join-table>` tag.

13.3.1 Elements below entity-mappings

(1) <entity-mappings>

Indicates the root tag.

The following table lists the specifiable attributes:

Table 13–6: Attributes of <entity-mappings>

Attribute name	Type	Optional/Required	Description
`version`	orm:versionType	Required	Specifies the JPA version.

(2) <package>

The `package` element specifies the class package described in the elements and attributes within the same mapping file. In the `package` element, the class name with a package name is specified for a class, and if the name differs from the package name specified in the `package` element, the package name is overwritten.

(3) <schema>

The `schema` element only applies to the entities that are specified within the same mapping file.

The `schema` element is overwritten by the following elements and attributes:

- The `schema` elements explicitly specified in `@Table` and `@SecondaryTable` of the `entity` class specified in the mapping file.
- The `schema`Attributes of the `table` element and the `secondary-table` element defined in the `entity` element.

(4) <catalog>

The `catalog` element only applies to the entities specified within the same mapping file.

The `catalog` element is overwritten by the following elements and attributes:

- The `catalog` element explicitly specified in `@Table` and `@SecondaryTable` of the `entity` class specified in the mapping file.
- The `catalog` attributes of the `table` lower element and the `secondary-table` lower element defined in the `entity` element.

Precautions

Depending upon a database, the `catalog` element might not exist. The `catalog` element does not exist in Oracle and HiRDB databases supported by the Cosminexus JPA Provider. Therefore, you cannot specify the `catalog` element. If specified, an exception will be thrown when you execute the application.

(5) <access>

The `access` element applies to the classes that are specified and managed within the same mapping file.

The `access` element is overwritten by the following annotations and attributes:

- The `access` type determined on the basis of the location in which the `entity` class annotations are specified.
- The `access` attributes defined in the `entity` element, the `mapped-superclass` element, and the `embeddable` element.

# Specify PROPERTY or FIELD as the specified value. For details on how to specify access methods for entity class fields, see *9.12.3 Specifying the access methods for the entity class fields*.

(6) <sequence-generator>

Adds the sequence generator.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–7: Attributes of <sequence-generator>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	See the `name` attribute in *8.12.56 @SequenceGenerator*.
`sequence-name`	xsd:string	Optional	See the `sequenceName` attribute in *8.12.56 @SequenceGenerator*.
`initial-value`	xsd:int	Optional	See the `initialValue` attribute in *8.12.56 @SequenceGenerator*.
`allocation-size`	xsd:int	Optional	See the `allocationSize` attribute in *8.12.56 @SequenceGenerator*.

13.3.2 Elements below persistence-unit-metadata

(1) <persistence-unit-metadata>

Specifies the definition related to the entire PersistenceUnit.

(2) <xml-mapping-metadata-complete>

If you specify the `xml-mapping-metadata-complete` element, the mapping metadata of the persistence unit is controlled and the annotations specified in the class are ignored.

If the `xml-mapping-metadata-complete` element is specified and the `XML` element is omitted, the default value is enabled.

Precautions

If you specify this element when an annotation is specified, the KDJE55532-W message is output.

(3) <persistence-unit-defaults>

Specifies the default value of the persistence unit.

(4) <schema>

The `schema` element applies to all `entity` classes, table generators, and join tables in the persistence unit.

The `schema` element is overwritten by the following elements and attributes:

- The `schema` element of the `entity-mappings` element.
- The `schema` attributes explicitly specified in `@Table` and `@SecondaryTable` of the `entity` class.
- The `schema` attributes specified in the `table` element and `secondary-table` element of the `entity` element.
- The `schema` elements explicitly specified in the `@TableGenerator` and `table-generator` element.
- The `schema` elements explicitly specified in the `@JoinTable` and `join-table` element.

(5) <catalog>

The `catalog` element applies to all `entity` classes, table generators, and join tables in the persistence unit.

The `catalog` element is overwritten by the following elements and attributes:

- The `catalog` element of the `entity-mappings` element.
- The `catalog` attributes explicitly specified in `@Table` and `@SecondaryTable` of the `entity` class.
- The `catalog` attributes specified in the `table` element and `secondary-table` element of the `entity` element.
- The `catalog` elements explicitly specified in the `@TableGenerator` and `table-generator` element.
- The `catalog` elements explicitly specified in the `@JoinTable` and `join-table` element.

Precautions

The `catalog` element does not exist in the Oracle and HiRDB databases supported in Cosminexus JPA Provider. Therefore, you cannot specify the `catalog` element. If specified, an exception will be thrown when the application is executed.

(6) <access>

The `access` element applies to all the managed classes in the persistence unit.

The `access` element is overwritten by the following annotations, elements, and attributes:

- The `access` type determined on the basis of the location in which the `entity` class annotations are specified.
- The `access` element of the `entity-mappings` element.

- The access attributes defined in the entity element, mapped-superclass element, and embeddable element.

Precautions

Specify PROPERTY or FIELD as the specified value. For details on how to specify access methods for entity class fields, see *9.12.3 Specifying the access methods for the entity class fields*.

(7) <cascade-persist>

The cascade-persist element applies to all the relationships in the persistence unit.

In addition to the values specified in annotations or O/R mapping files, the specification of the cascade-persist element adds the cascade persistence option to all the relationships.

Precautions

If you specify the cascade-persist element, you cannot overwrite and disable.

(8) <entity-listeners>

The entity-listeners element defines the default entity listeners of the persistence unit.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

13.3.3 Elements below table-generator

(1) <table-generator>

The generators defined by the table-generator element will be applied to the persistence unit.

The defined generator is added to the generators defined using annotations. If a generator with the same name is defined in the annotation, the generator defined by the table-generator element is overwritten.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–8: Attributes of <table-generator>

Attribute name	Type	Optional/Required	Description
name	xsd:string	Required	See the name attribute in *8.12.60 @TableGenerator*.
table	xsd:string	Optional	See the table attribute in *8.12.60 @TableGenerator*.
catalog	xsd:string	Optional	See the catalog attribute in *8.12.60 @TableGenerator*.
schema	xsd:string	Optional	See the schema attribute in *8.12.60 @TableGenerator*.
pk-column-name	xsd:string	Optional	See the pkColumnName attribute in *8.12.60 @TableGenerator*.

Attribute name	Type	Optional/Required	Description
`value-column-name`	xsd:string	Optional	See the `valueColumnName` attribute in *8.12.60 @TableGenerator*.
`pk-column-value`	xsd:string	Optional	See the `pkColumnValue` attribute in *8.12.60 @TableGenerator*.
`initial-value`	xsd:int	Optional	See the `initialValue` attribute in *8.12.60 @TableGenerator*.
`allocation-size`	xsd:int	Optional	See the `allocationSize` attribute in *8.12.60 @TableGenerator*.

13.3.4 Elements below named-query

(1) <named-query>

The named query defined by the `named-query` element applies to the persistence unit.

The defined named query is added to the named query defined using annotations. If a named query with the same name is defined in the annotation, the named query defined by the `named-query` element is overwritten.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–9: Attributes of <named-query>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	See the `name` attribute in *8.12.35 @NamedQuery*.

13.3.5 Elements below named-native-query

(1) <named-native-query>

The named native query defined by the `named-native-query` element applies to the persistence unit.

The defined named native query is added to the named native query defined using annotations. If a named native query with the same name is defined in the annotation, the named native query defined by the `named-native-query` element of the O/R mapping file is overwritten.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–10: Attributes of <named-native-query>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	See the `name` attribute in *8.12.33 @NamedNativeQuery*.

Attribute name	Type	Optional/Required	Description
`result-class`	xsd:string	Optional	See the `resultClass` attribute in *8.12.33 @NamedNativeQuery*.
`result-set-mapping`	xsd:string	Optional	See the `resultSetMapping` attribute in *8.12.33 @NamedNativeQuery*.

13.3.6 Elements below sql-result-set-mapping

(1) <sql-result-set-mapping>

The SQL result set mapping defined by the `sql-result-set-mapping` element applies to the persistence unit.

The defined SQL result set mapping is added to the SQL result set mapping defined using annotations. If SQL result set mapping with the same name is defined in the annotation, the SQL result set mapping defined by the `sql-result-set-mapping` element of the O/R mapping file is overwritten.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–11: Attributes of <sql-result-set-mapping>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	See the `name` attribute in *8.12.57 @SqlResultSetMapping*.

13.3.7 Elements below mapped-superclass

The following elements and attributes apply only to the mapped superclass that is the target of the elements and attributes:

(1) <mapped-superclass>

The `mapped-superclass` element defines the mapped superclass of the persistence unit.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–12: Attributes of <mapped-superclass>

Attribute name	Type	Optional/Required	Description
`class`	xsd:string	Required	Class name of the mapped superclass.
`access`	orm:access-type	Optional	The `access` attribute defines the `access` type of the mapped superclass. The `access` attribute overwrites the `access` type specified in the `persistence-unit-defaults` element (default element) and `entity-mappings` element (element valid for the entire persistence unit) provided to the mapped superclass.#1

Attribute name	Type	Optional/Required	Description
`metadata-complete`[#2]	xsd:boolean	Optional	If the `metadata-complete` attribute is specified in the `mapped-superclass` element, the annotations specified in the mapped superclass and in the fields and properties of the mapped superclass are ignored. If `metadata-complete` is specified in the `mapped-superclass` element and if the `XML` element is omitted, the default value is enabled.

#1

Specify PROPERTY or FIELD as the specified value. For details on how to specify access methods for entity class fields, see *9.12.3 Specifying the access methods for the entity class fields*.

#2

If you specify this element when an annotation is specified, the KDJE55532-W message is output.

(2) <id-class>

The `id-class` element overwrites `@IdClass` specified in the mapped superclass.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–13: Attributes of <id-class>

Attribute name	Type	Optional/Required	Description
`class`	xsd:string	Required	See the `value` attribute in *8.12.22 @IdClass*.

(3) <exclude-default-listeners>

The `exclude-default-listeners` element is applied regardless of whether `@ExcludeDefaultListeners` is specified in the mapped superclass.

The `exclude-default-listeners` element controls the default entity listeners of the mapped superclass and the sub class.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

(4) <exclude-superclass-listeners>

The `exclude-superclass-listeners` element is applied regardless of whether `@ExcludeSuperclassListeners` is specified in the mapped superclass.

The `exclude-superclass-listeners` element controls the superclass listeners of the mapped superclass and the sub class.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

(5) <entity-listeners>

The `entity-listeners` element overwrites `@EntityListeners` of the mapped superclass.

If these listeners are not controlled using other methods, these listeners will be applied in the mapped superclass and the sub class.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

(6) <pre-persist>, <post-persist>, <pre-remove>, <post-remove>, <pre-update>, <post-update>, <post-load>

These elements overwrite the definition of the lifecycle callback method according to the corresponding annotations, in the mapped superclass.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–14: Attributes of <pre-persist>, <post-persist>, <pre-remove>, <post-remove>, <pre-update>, <post-update>, and <post-load>

Attribute name	Type	Optional/Required	Description
`method-name`	xsd:string	Required	Target method name.

(7) <id>

The id element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–15: Attributes of <id>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Specifies a primary key property or field.

(8) <embedded-id>

The `embedded-id` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–16: Attributes of <embedded-id>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Specifies a compound primary key.

(9) <basic>

The `basic` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–17: Attributes of <basic>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Specifies the methods and fields that map the type.
`fetch`	orm:fetch-type	Optional	Specifies the value of the Fetch strategy. For details, see the `fetch` attribute in *8.12.5 @Basic*.
`optional`	xsd:boolean	Optional	Specifies whether to use null in the field (property). For details, see the `optional` attribute in *8.12.5 @Basic*.

(10) <version>

The `version` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–18: Attributes of <version>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Specifies a version property or field.

(11) <many-to-one>

The `many-to-one` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–19: Attributes of <many-to-one>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Specifies methods or fields with a many-to-one relationship.
`target-entity`	xsd:string	Optional	Specifies the related `entity` classes. For details, see the `targetEntity` attribute in *8.12.29 @ManyToOne*.
`fetch`	orm:fetch-type	Optional	Specifies the value of the Fetch strategy. For details, see the `fetch` attribute in *8.12.29 @ManyToOne*.
`optional`	xsd:boolean	Optional	Defines whether to specify null in all the non-primitive field and property values. For details, see the `optional` attribute in *8.12.29 @ManyToOne*.

(12) <one-to-many>

The `one-to-many` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–20: Attributes of <one-to-many>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Methods or fields with a one-to-many relationship.
`target-entity`	xsd:string	Optional	Specifies the related `entity` classes. For details, see the `targetEntity` attribute in *8.12.36 @OneToMany*.
`fetch`	orm:fetch-type	Optional	Specifies the value of the Fetch strategy. For details, see the `fetch` attribute in *8.12.36 @OneToMany*.
`mapped-by`	xsd:string	Optional	Specifies the field (property) names allocated to the elements of the owned-side `entity` class and the relations are maintained in the owner-side `entity` class. For details, see the `mappedBy` attribute in *8.12.36 @OneToMany*.

(13) <one-to-one>

The `one-to-one` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–21: Attributes of <one-to-one>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Methods or fields with a one-to-one relationship.
`target-entity`	xsd:string	Optional	See the `targetEntity` attribute in *8.12.37 @OneToOne*.
`fetch`	orm:fetch-type	Optional	See the `fetch` attribute in *8.12.37 @OneToOne*.
`optional`	xsd:boolean	Optional	See the `optional` attribute in *8.12.37 @OneToOne*.
`mapped-by`	xsd:string	Optional	See the `mappedBy` attribute in *8.12.37 @OneToOne*.

(14) <many-to-many>

The `many-to-many` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–22: Attributes of <many-to-many>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Methods or fields with a many-to-many relationship.
`target-entity`	xsd:string	Optional	See the `targetEntity` attribute in *8.12.28 @ManyToMany*.
`fetch`	orm:fetch-type	Optional	See the `fetch` attribute in *8.12.28 @ManyToMany*.
`mapped-by`	xsd:string	Optional	See the `mappedBy` attribute in *8.12.28 @ManyToMany*.

(15) <embedded>

The `embedded` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–23: Attributes of <embedded>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Property or field that is an embedded object.

(16) <transient>

The `transient` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–24: Attributes of <transient>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Property or field that is non-persistent.

13.3.8 Elements below entity

The following elements and attributes apply only to the `entity` class that is the target of the lower elements and attributes:

(1) <entity>

The `entity` element defines the entities of the persistence unit.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–25: Attributes of <entity>

Attribute name	Type	Optional/Required	Description
name	xsd:string	Optional	The name attribute defines the name of the entity. The name attribute is overwritten regardless of whether the entity name defined by the name element of @Entity is explicitly specified or is the default value. The operation when the entity class name is overwritten might not execute normally.
class	xsd:string	Required	Entity class name.
access	orm:access-type	Optional	The access attribute defines the access type of the entity class. The access attribute overwrites the access type specified in the persistence-unit-defaults element (default element) and entity-mappings element (element valid for the entire persistence unit) provided to the entity class.#1
metadata-complete#2	xsd:boolean	Optional	If the metadata-complete attribute is specified in the entity element, the annotations specified in the entity class and in the fields and properties of the entity class are ignored. If metadata-complete is specified in the entity element and if the XML element is omitted, the default value is enabled.

#1

Specify PROPERTY or FIELD as the specified value. For details on how to specify access methods for entity class fields, see *9.12.3 Specifying the access methods for the entity class fields*.

#2

If you specify this element when an annotation is specified, the KDJE55532-W message is output.

(2) <table>

The table element overwrites @Table (including default values) of the entity class.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–26: Attributes of <table>

Attribute name	Type	Optional/Required	Description
name	xsd:string	Optional	See the name attribute in *8.12.59 @Table*.
catalog	xsd:string	Optional	See the catalog attribute in *8.12.59 @Table*.
schema	xsd:string	Optional	See the schema attribute in *8.12.59 @Table*.

(3) <secondary-table>

The `secondary-table` element overwrites all `@SecondaryTable` and `@SecondaryTables` (including default values) of the `entity` class.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–27: Attributes of <secondary-table>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	See the `name` attribute in *8.12.54 @SecondaryTable*.
`catalog`	xsd:string	Optional	See the `catalog` attribute in *8.12.54 @SecondaryTable*.
`schema`	xsd:string	Optional	See the `schema` attribute in *8.12.54 @SecondaryTable*.

(4) <primary-key-join-column>

The `primary-key-join-column` element overwrites all `@PrimaryKeyJoinColumn` and `@PrimaryKeyJoinColumns` (including default values) of the `entity` class.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–28: Attributes of <primary-key-join-column>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Optional	See the `name` attribute in *8.12.51 @PrimaryKeyJoinColumn*.
`referenced-column-name`	xsd:string	Optional	See the `referencedColumnName` attribute in *8.12.51 @PrimaryKeyJoinColumn*.
`column-definition`	xsd:string	Optional	See the `columnDefinition` attribute in *8.12.51 @PrimaryKeyJoinColumn*.

(5) <id-class>

The `id-class` element overwrites `@IdClass` specified in the `entity` class.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–29: Attributes of <id-class>

Attribute name	Type	Optional/Required	Description
`value`	xsd:string	Required	See the `value` attribute in *8.12.22 @IdClass*.

(6) <inheritance>

The `inheritance` element overwrites `@Inheritance` (including default values) of the `entity` class.

The `inheritance` element will be applied to the `entity` class and the sub class (if the sub class specified in the annotation and XML element is not overwritten using another method).

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–30: Attributes of <inheritance>

Attribute name	Type	Optional/Required	Description
`strategy`	orm:inheritance-type	Optional	See the `strategy` attribute in *8.12.23 @Inheritance*.

(7) <discriminator-value>

The `discriminator-value` element overwrites `@DiscriminatorValue` (including default values) of the `entity` class.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

(8) <discriminator-column>

The `discriminator-column` element overwrites `@DiscriminatorColumn` (including default values) of the `entity` class.

The `discriminator-column` element will be applied to the `entity` class and the sub class (if the sub class specified in the annotation and XML element is not overwritten using another method).

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–31: Attributes of <discriminator-column>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Optional	See the `name` attribute in *8.12.8 @DiscriminatorColumn*.
`discriminator-type`	orm:discriminator-type	Optional	See the `discriminatorType` attribute in *8.12.8 @DiscriminatorColumn*.
`column-definition`	xsd:string	Optional	See the `columnDefinition` attribute in *8.12.8 @DiscriminatorColumn*.
`length`	xsd:int	Optional	See the `length` attribute in *8.12.8 @DiscriminatorColumn*.

(9) <sequence-generator>

The generator defined by the `sequence-generator` element is added to the generators defined using annotations and the other generators defined in the O/R mapping file. If a generator with the same name is defined in the annotation, the generator defined by the `sequence-generator` element overwrites this generator.

The generator defined by the `sequence-generator` element applies to the persistence unit.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–32: Attributes of <sequence-generator>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	See the `name` attribute in *8.12.56 @SequenceGenerator*.
`sequence-name`	xsd:string	Optional	See the `sequenceName` attribute in *8.12.56 @SequenceGenerator*.
`initial-value`	xsd:int	Optional	See the `initialValue` attribute in *8.12.56 @SequenceGenerator*.
`allocation-size`	xsd:int	Optional	See the `allocationSize` attribute in *8.12.56 @SequenceGenerator*.

(10) <table-generator>

The generator defined by the `table-generator` element is added to the generators defined using annotations and the other generators defined in the O/R mapping file. If a generator with the same name is defined in the annotation, the generator defined by the `table-generator` element overwrites this generator.

The generators defined by the `table-generator` element will be applied to the persistence unit.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–33: Attributes of <table-generator>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	See the `name` attribute in *8.12.60 @TableGenerator*.
`table`	xsd:string	Optional	See the `table` attribute in *8.12.60 @TableGenerator*.
`catalog`	xsd:string	Optional	See the `catalog` attribute in *8.12.60 @TableGenerator*.
`schema`	xsd:string	Optional	See the `schema` attribute in *8.12.60 @TableGenerator*.
`pk-column-name`	xsd:string	Optional	See the `pkColumnName` attribute in *8.12.60 @TableGenerator*.
`value-column-name`	xsd:string	Optional	See the `valueColumnName` attribute in *8.12.60 @TableGenerator*.

Attribute name	Type	Optional/Required	Description
`pk-column-value`	xsd:string	Optional	See the `pkColumnValue` attribute in *8.12.60 @TableGenerator*.
`initial-value`	xsd:int	Optional	See the `initialValue` attribute in *8.12.60 @TableGenerator*.
`allocation-size`	xsd:int	Optional	See the `allocationSize` attribute in *8.12.60 @TableGenerator*.

(11) <named-query>

The named query defined by the `named-query` element is added to the named query defined using annotations and the other named queries defined in the O/R mapping file. If a named query with the same name is defined in the annotation, the named query defined by the `named-query` element overwrites this named query.

The named query defined by the `named-query` element applies to the persistence unit.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–34: Attributes of <named-query>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	See the `name` attribute in *8.12.35 @NamedQuery*.

(12) <named-native-query>

The named native query defined by the `named-native-query` element is added to the named native query defined using annotations and the other named native queries defined in the O/R mapping file. If a named native query with the same name is defined in the annotation, the named native query defined by the `named-native-query` element overwrites this named native query.

The named native query defined by the `named-native-query` element applies to the persistence unit.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–35: Attributes of <named-native-query>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	See the `name` attribute in *8.12.33 @NamedNativeQuery*.
`result-class`	xsd:string	Optional	See the `resultClass` attribute in *8.12.33 @NamedNativeQuery*.
`result-set-mapping`	xsd:string	Optional	See the `resultSetMapping` attribute in *8.12.33 @NamedNativeQuery*.

(13) <sql-result-set-mapping>

The SQL result mapping defined by the `sql-result-set-mapping` element is added to the SQL result mapping defined using annotations and the other SQL result mapping defined in the O/R mapping file. If SQL result mapping with the same name is defined in the annotation, the SQL result mapping defined in the `sql-result-set-mapping` element overwrites this SQL result mapping.

The SQL result mapping defined by the `sql-result-set-mapping` element applies to the persistence unit.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–36: Attributes of <sql-result-set-mapping>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	See the `name` attribute in *8.12.57 @SqlResultSetMapping*.

(14) <exclude-default-listeners>

The `exclude-default-listeners` element is applied regardless of whether `@ExcludeDefaultListeners` is specified in the `entity` class.

The `exclude-default-listeners` element controls the default entity listeners of the `entity` class and the sub class.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

(15) <exclude-superclass-listeners>

The `exclude-superclass-listeners` element is applied regardless of whether `@ExcludeSuperclassListeners` is specified in the `entity` class.

The `exclude-superclass-listeners` element controls the superclass listeners of the `entity` class and the sub class.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

(16) <entity-listeners>

The `entity-listeners` element overwrites `@EntityListeners` of the `entity` class.

If these listeners are not controlled using other methods, these listeners will be applied in the `entity` class and the sub class.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

(17) <pre-persist>, <post-persist>, <pre-remove>, <post-remove>, <pre-update>, <post-update>, <post-load>

These elements overwrite the definition of the lifecycle callback method according to the corresponding annotations, in the `entity` class.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–37: Attributes of <pre-persist>, <post-persist>, <pre-remove>, <post-remove>, <pre-update>, <post-update>, and <post-load>

Attribute name	Type	Optional/Required	Description
`method-name`	xsd:string	Required	Target method name.

(18) <attribute-override>

The `attribute-override` element is added to the value defined in `@AttributeOverride` or `@AttributeOverrides` of the `entity` class. The `attribute-override` element overwrites the `AttributeOverride` element with the same attribute name.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–38: Attributes of <attribute-override>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	See the `name` attribute in *8.12.3 @AttributeOverride*.

(19) <association-override>

The `association-override` element is added to the value defined in `@AssociationOverride` or `@AssociationOverrides` of the `entity` class. The `association-override` element overwrites the `AssociationOverride` element with the same attribute name.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–39: Attributes of <association-override>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	See the `name` attribute in *8.12.1 @AssociationOverride*.

(20) <id>

The `id` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–40: Attributes of <id>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Primary key property or field.

(21) <embedded-id>

The `embedded-id` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–41: Attributes of <embedded-id>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Compound primary key.

(22) <basic>

The `basic` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–42: Attributes of <basic>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Methods or fields that map the type.
`fetch`	orm:fetch-type	Optional	See the `fetch` attribute in *8.12.5 @Basic*.
`optional`	xsd:boolean	Optional	See the `optional` attribute in *8.12.5 @Basic*.

(23) <version>

The `version` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–43: Attributes of <version>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Version property or field.

(24) <many-to-one>

The `many-to-one` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–44: Attributes of <many-to-one>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Methods or fields with a many-to-one relationship.
`target-entity`	xsd:string	Optional	See the `targetEntity` attribute in *8.12.29 @ManyToOne*.
`fetch`	orm:fetch-type	Optional	See the `fetch` attribute in *8.12.29 @ManyToOne*.
`optional`	xsd:boolean	Optional	See the `optional` attribute in *8.12.29 @ManyToOne*.

(25) <one-to-many>

The `one-to-many` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–45: Attributes of <one-to-many>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Methods or fields with a one-to-many relationship.
`target-entity`	xsd:string	Optional	See the `targetEntity` attribute in *8.12.36 @OneToMany*.
`fetch`	orm:fetch-type	Optional	See the `fetch` attribute in *8.12.36 @OneToMany*.
`mapped-by`	xsd:string	Optional	See the `mappedBy` attribute in *8.12.36 @OneToMany*.

(26) <one-to-one>

The `one-to-one` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–46: Attributes of <one-to-one>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Methods or fields with a one-to-one relationship.
`target-entity`	xsd:string	Optional	See the `targetEntity` attribute in *8.12.37 @OneToOne*.

Attribute name	Type	Optional/Required	Description
fetch	orm:fetch-type	Optional	See the fetch attribute in *8.12.37 @OneToOne*.
optional	xsd:boolean	Optional	See the optional attribute in *8.12.37 @OneToOne*.
mapped-by	xsd:string	Optional	See the mappedBy attribute in *8.12.37 @OneToOne*.

(27) <many-to-many>

The many-to-many element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–47: Attributes of <many-to-many>

Attribute name	Type	Optional/Required	Description
name	xsd:string	Required	Methods or fields with a many-to-many relationship.
target-entity	xsd:string	Optional	See the targetEntity attribute in *8.12.28 @ManyToMany*.
fetch	orm:fetch-type	Optional	See the fetch attribute in *8.12.28 @ManyToMany*.
mapped-by	xsd:string	Optional	See the mappedBy attribute in *8.12.28 @ManyToMany*.

(28) <embedded>

The embedded element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–48: Attributes of <embedded>

Attribute name	Type	Optional/Required	Description
name	xsd:string	Required	Property or field that is an embedded object.

(29) <transient>

The transient element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–49: Attributes of <transient>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Property or field that is non-persistent.

13.3.9 Elements under embeddable

The following elements and attributes apply only to the `embeddable` class that is the target of the elements and attributes:

(1) <embeddable>

The `embeddable` element defines the `embeddable` class of the persistence unit.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–50: Attributes of <embeddable>

Attribute name	Type	Optional/Required	Description
`class`	xsd:string	Required	Class name of the `embeddable` class.
`access`	orm:access-type	Optional	The `access` attribute defines the `access` type of the `embeddable` class. The `access` attribute overwrites the `access` type specified in the persistence-unit-defaults element (default element) and `entity-mappings` element (element valid for the entire persistence unit) provided to the `embeddable` class.#1
`metadata-complete`#2	xsd:boolean	Optional	If the `metadata-complete` attribute is specified in the `embeddable` element, the annotations specified in the `embeddable` class and in the fields and properties of the `embeddable` class are ignored. If metadata-complete is specified in the `embeddable` element and if the XML element is omitted, the default value is enabled.

#1

Specify PROPERTY or FIELD as the specified value. For details on how to specify access methods for entity class fields, see *9.12.3 Specifying the access methods for the entity class fields*.

#2

If you specify this element when an annotation is specified, the KDJE55532-W message is output.

(2) <basic>

The `basic` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–51: Attributes of <basic>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Methods or fields that map the type.
`fetch`	orm:fetch-type	Optional	See the `fetch` attribute in *8.12.5 @Basic*.
`optional`	xsd:boolean	Optional	See the `optional` attribute in *8.12.5 @Basic*.

(3) <transient>

The `transient` element overwrites the mapping specified in fields and properties.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–52: Attributes of <transient>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	Property or field that is non-persistent.

13.3.10 Other elements

(1) <description>

Adds the description.

(2) <entity-listener>

Specifies the entity listener.

The following table lists the specifiable attributes:

Table 13–53: Attributes of <entity-listener>

Attribute name	Type	Optional/Required	Description
`class`	xsd:string	Required	Class name of the entity listener.

(3) <pre-persist>, <post-persist>, <pre-remove>, <post-remove>, <pre-update>, <post-update>, <post-load>

Specifies the lifecycle callback method.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–54: Attributes of <pre-persist>, <post-persist>, <pre-remove>, <post-remove>, <pre-update>, <post-update>, and <post-load>

Attribute name	Type	Optional/Required	Description
`method-name`	xsd:string	Required	Target method name.

(4) <unique-constraint>

Adds a unique constraint to DDL.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

(5) <column-name>

Specifies the name of the column in which the unique constraint will be added.

The `column-name` element corresponds to the `columnNames` attribute of `@UniqueConstraint`.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

(6) <query>

Specifies the query string.

The `query` element corresponds to the query attribute of `@NamedQuery` and the `query` attribute of `@NamedNativeQuery`.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

(7) <hint>

Adds a hint to the query.

The `hint` element corresponds to the `hints` attribute of `@NamedQuery` and the `hints` attribute of `@NamedNativeQuery`.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–55: Attributes of <hint>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	See the `name` attribute in *8.12.53 @QueryHint*.

Attribute name	Type	Optional/Required	Description
`value`	xsd:string	Required	See the `value` attribute in *8.12.53 @QueryHint*.

(8) <entity-result>

Specifies the `entity` class used for mapping the native SQL query result.

The `entity-result` element corresponds to the `entities` attribute of `@SqlResultSetMapping`.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–56: Attributes of <entity-result>

Attribute name	Type	Optional/Required	Description
`entity-class`	xsd:string	Required	See the `entityClass` attribute in *8.12.15 @EntityResult*.
`discriminator-column`	xsd:string	Optional	See the `discriminatorColumn` attribute in *8.12.15 @EntityResult*.

(9) <field-result>

Specifies the field used for mapping the native SQL query result.

The `field-result` element corresponds to the `fields` attribute of `@EntityResult`.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–57: Attributes of <field-result>

Type	Attribute name	Optional/Required	Description
xsd:string	`name`	Required	See the `name` attribute in *8.12.19 @FieldResult*.
xsd:string	`column`	Required	See the `column` attribute in *8.12.19 @FieldResult*.

(10) <column-result>

Specifies the column used for mapping the native SQL query result.

The `column-result` element corresponds to the `columns` attribute of `@SqlResultSetMapping`.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–58: Attributes of <column-result>

Type	Attribute name	Optional/Required	Description
xsd:string	`name`	Required	See the `name` attribute in *8.12.7 @ColumnResult*.

(11) <attributes>

The functionality of the `attributes` element does not exist.

(12) <column>

The `column` element specifies the column mapping for the Persistent field or property.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–59: Attributes of <column>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Optional	See the `name` attribute in *8.12.6 @Column*.
`unique`	xsd:boolean	Optional	See the `unique` attribute in *8.12.6 @Column*.
`nullable`	xsd:boolean	Optional	See the `nullable` attribute in *8.12.6 @Column*.
`insertable`	xsd:boolean	Optional	See the `insertable` attribute in *8.12.6 @Column*.
`updatable`	xsd:boolean	Optional	See the `updatable` attribute in *8.12.6 @Column*.
`column-definition`	xsd:string	Optional	See the `columnDefinition` attribute in *8.12.6 @Column*.
`table`	xsd:string	Optional	See the `table` attribute in *8.12.6 @Column*.
`length`	xsd:int	Optional	See the `length` attribute in *8.12.6 @Column*.
`precision`	xsd:int	Optional	See the `precision` attribute in *8.12.6 @Column*.
`scale`	xsd:int	Optional	See the `scale` attribute in *8.12.6 @Column*.

(13) <generated-value>

Specifies the strategy for generating the primary key value.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–60: Attributes of <generated-value>

Attribute name	Type	Optional/Required	Description
`strategy`	orm:generation-type	Optional	See the `strategy` attribute in *8.12.20 @GeneratedValue*.
`generator`	xsd:string	Optional	See the `generator` attribute in *8.12.20 @GeneratedValue*.

(14) <temporal>

Specified when mapping to the DATE, TIME, and TIMESTAMP type.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

(15) <table-generator>

Adds the table generator.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–61: Attributes of <table-generator>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	See the `name` attribute in *8.12.60 @TableGenerator*.
`table`	xsd:string	Optional	See the `table` attribute in *8.12.60 @TableGenerator*.
`catalog`	xsd:string	Optional	See the `catalog` attribute in *8.12.60 @TableGenerator*.
`schema`	xsd:string	Optional	See the `schema` attribute in *8.12.60 @TableGenerator*.
`pk-column-name`	xsd:string	Optional	See the `pkColumnName` attribute in *8.12.60 @TableGenerator*.
`value-column-name`	xsd:string	Optional	See the `valueColumnName` attribute in *8.12.60 @TableGenerator*.
`pk-column-value`	xsd:string	Optional	See the `pkColumnValue` attribute in *8.12.60 @TableGenerator*.
`initial-value`	xsd:int	Optional	See the `initialValue` attribute in *8.12.60 @TableGenerator*.
`allocation-size`	xsd:int	Optional	See the `allocationSize` attribute in *8.12.60 @TableGenerator*.

(16) <attribute-override>

Overwrites the mapping of properties and fields.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–62: Attributes of <attribute-override>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Required	See the `name` attribute in *8.12.3 @AttributeOverride*.

(17) <lob>

The `lob` element is specified when mapping to the Lob type.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

(18) <enumerated>

Specified when mapping to the enumeration type.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

(19) <join-column>

In order to join tables, the `join-column` element specifies the external key column of the join table corresponding to the owner-side entity.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–63: Attributes of <join-column>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Optional	See the `name` attribute in *8.12.24 @JoinColumn*.
`referenced-column-name`	xsd:string	Optional	See the `referencedColumnName` attribute in *8.12.24 @JoinColumn*.
`unique`	xsd:boolean	Optional	See the `unique` attribute in *8.12.24 @JoinColumn*.
`nullable`	xsd:boolean	Optional	See the `nullable` attribute in *8.12.24 @JoinColumn*.
`insertable`	xsd:boolean	Optional	See the `insertable` attribute in *8.12.24 @JoinColumn*.
`updatable`	xsd:boolean	Optional	See the `updatable` attribute in *8.12.24 @JoinColumn*.
`column-definition`	xsd:string	Optional	See the `columnDefinition` attribute in *8.12.24 @JoinColumn*.
`table`	xsd:string	Optional	See the `table` attribute in *8.12.24 @JoinColumn*.

(20) <join-table>

The `join-table` element specifies the join table to be used in many-to-many and the unilateral one-to-many relationships.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–64: Attributes of <join-table>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Optional	See the `name` attribute in *8.12.26 @JoinTable*.
`catalog`	xsd:string	Optional	See the `catalog` attribute in *8.12.26 @JoinTable*.
`schema`	xsd:string	Optional	See the `schema` attribute in *8.12.26 @JoinTable*.

(21) <inverse-join-column>

In order to join tables, the `inverse-join-column` element specifies the external key column of the join table corresponding to the owned-side entity.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–65: Attributes of <inverse-join-column>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Optional	See the `name` attribute in *8.12.24 @JoinColumn*.
`referenced-column-name`	xsd:string	Optional	See the `referencedColumnName` attribute in *8.12.24 @JoinColumn*.
`unique`	xsd:boolean	Optional	See the `unique` attribute in *8.12.24 @JoinColumn*.
`nullable`	xsd:boolean	Optional	See the `nullable` attribute in *8.12.24 @JoinColumn*.
`insertable`	xsd:boolean	Optional	See the `insertable` attribute in *8.12.24 @JoinColumn*.
`updatable`	xsd:boolean	Optional	See the `updatable` attribute in *8.12.24 @JoinColumn*.
`column-definition`	xsd:string	Optional	See the `columnDefinition` attribute in *8.12.24 @JoinColumn*.
`table`	xsd:string	Optional	See the `table` attribute in *8.12.24 @JoinColumn*.

(22) <cascade>

The `cascade` element specifies the operations that you can cascade.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

(23) <cascade-all>

Cascades all the operations.

(24) <cascade-persist>

Cascades the persist operation.

(25) <cascade-merge>

Cascades the merge operation.

(26) <cascade-remove>

Cascades the remove operation.

(27) <cascade-refresh>

Cascades the refresh operation.

(28) <order-by>

Specifies the order to be applied when maintaining a relation in the collection.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

(29) <map-key>

Specifies the map key as the Map type relation.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–66: Attributes of <map-key>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Optional	See the `name` attribute in *8.12.30 @MapKey*.

(30) <primary-key-join-column>

The `primary-key-join-column` element specifies the primary key column used as the external key that JOINS with other tables.

For details on the functionality and attributes, see *8.12 javax.persistence package*.

The following table lists the specifiable attributes:

Table 13–67: Attributes of <primary-key-join-column>

Attribute name	Type	Optional/Required	Description
`name`	xsd:string	Optional	See the `name` attribute in *8.12.51 @PrimaryKeyJoinColumn*.
`referenced-column-name`	xsd:string	Optional	See the `referencedColumnName` attribute in *8.12.51 @PrimaryKeyJoinColumn*.
`column-definition`	xsd:string	Optional	See the `columnDefinition` attribute in *8.12.51 @PrimaryKeyJoinColumn*.

13.4 Query hint

In Cosminexus JPA Provider, you can specify a query hint in the hint element that is the lower element of the `named-query` element in the O/R mapping file. Note that you can also specify the query hint in the `@Hint` annotation of the argument for the `@NamedQuery` annotation. For details on the query hint to be specified for annotations, see *8.12.53 @QueryHint*.

The following table describes the query hints that can be specified in the hint element that is the lower element of the `named-query` element in the O/R mapping file:

Table 13–68: Query hints that can be used in Cosminexus JPA Provider

Key name	Description	Specifiable value	Default

`cosminexus.jpa.pessimistic-lock`	Specifies whether the pessimistic lock will be used.	`NoLock`
		The pessimistic lock will not be used.
		`Lock`
		The pessimistic lock will be used.
		If the target table is already locked, the release of the lock will be awaited.
		• In Oracle
		`SELECT ... FOR UPDATE` will be issued.
		• In HiRDB
		`SELECT ... WITH EXCLUSIVE LOCK` will be issued.
		`LockNoWait`
		The pessimistic lock will be used.
		If the target table is already locked, an exception will occur.
		• In Oracle
		`SELECT ... FOR UPDATE NO WAIT` will be issued.
		• In HiRDB
		`SELECT ... WITH EXCLUSIVE LOCK NO WAIT` will be issued.
			`NoLock`

Note:
The data type that can be specified is String.

Important note

If a value outside the specifiable range is set for a query hint specified in the O/R mapping file, an exception will occur when the application starts. Note that the value is not case sensitive.

14 Files Used in Web Server Integration

This chapter describes the storage locations, functionality, and format of the files used in Web server integration and the keys that you can specify in the files.

14.1 List of files used in Web server integration

The following table lists the files used in Web server integration:

Table 14–1: List of files used for Web server integration

File name	Classification	Overview	Reference
`isapi_redirect.conf`	Redirector action definition file for Microsoft IIS	Define the action of the redirector for Microsoft IIS.	*14.2.1*
`mod_jk.conf`	Redirector action definition file for Cosminexus HTTP Server	Define the action of the redirector for Cosminexus HTTP Server.	*14.2.2*
`uriworkermap.properties`	Mapping definition file for Microsoft IIS	Define the URL pattern to be transferred to the Web container server through the request to Microsoft IIS.	*14.2.3*
`workers.properties`	Worker definition file	Define the action of the redirector.	*14.2.4*

14.2 Details on files used in Web server integration

14.2.1 isapi_redirect.conf (Redirector action definition file for Microsoft IIS)

(1) Format

Specify the key as follows:

```
key-name = value
```

Specification method

- The string up to the linefeed is a value.
- The line beginning with a hash mark (#) is a comment.
- If you define a line without a value, the line is ignored. The parameters not defined as valid key names are also ignored even if defined in the action definition file.
- Up to 1023 characters are valid as *key-name=value*. The part exceeding this number is truncated.

(2) File storage location

Cosminexus-installation-directory`\CC\web\redirector\isapi_redirect.conf`

(3) Functionality

This action definition file defines the action parameters of the redirector for Microsoft IIS.

(4) Specifiable keys

The specifiable keys and default values are described below. If you specify an invalid value in these keys, the operations may not produce the desired results. If you specify two or more keys, the last specified key becomes valid.

Key name	Contents	Default value
`connection_sharing`	Specify whether to share a connection between workers. If you specify `true`: The workers connected to the same host and the same port share a connection, even if the worker definitions are different. If you specify `false`: The workers having different worker definitions do not share a connection.	`true`
`connect_timeout`	Specify an integer (units: seconds) from 0 to 3600 for the timeout value to connect to the Web container when a request is sent. If you specify a non-numeric value, or a numeric value outside the range, a message will be output and the default value will be used. If you set the timeout as 0 or a time period longer than the resend timer of data transmission in TCP, the timeout value becomes the timeout value of TCP. In such a case, a message indicating that an invalid timeout value has been specified will not be output.	`30`
`filter_priority`	Specify the priority order used when the redirector is registered as an ISAPI filter. The following strings can be specified:	`high`

Key name	Contents	Default value
	• `high` (priority order is set to "`high`") • `medium` (priority order is set to "`medium`") • `low` (priority order is set to "`low`")	
`gateway_host`	Specify the host name or the IP address of the gateway. When requests without a Host header are redirected to files, such as the `welcome` file, the host name of the Location header URL becomes the specified value.	None
`gateway_https_scheme`	When a client request uses https as a scheme, and the scheme for a Web server will become http by using an SSL accelerator, specify `true`. If you specify `true`, https is assumed to be used as the scheme for requests sent to the Web server. If you specify `false`, no action occurs.	`false`
`gateway_port`	Specify the port number of the gateway. If a request has no `Host` header and the request is to be redirected to a location such as a welcome file, the port number portion of a URL that is specified in the `Location` header becomes the specified value. Always specify this parameter together with `gateway_host`. If you specify `gateway_host` and omit this parameter, an access via http uses 80, and an access via hppts uses 443.	None
`log_file_dir`	Specify the output location directory of the log file. When specified as a relative path: Specify a directory name present in *Cosminexus-installation-directory*`\CC\web\redirector`. When specified as an absolute path: Specify the coded directory name. Note that you must specify write permission to the Users group, as the access permission to the directory specified as the output destination. If access permission is not set, the log file is not output. If the same value has been specified in `log_file_prefix` and `trace_log_file_prefix`, you need to specify a value different from `trace_log_file_dir` for this key. If you specify the same value, the redirector will not run.	Logs
`log_file_num`	Specify the maximum number of redirector log files. If this number is exceeded, the old log files are overwritten. Specify an integer value from 1 to 64.	`5`
`log_file_prefix`	This is a prefix for the log file name. The actual log file name will be the value specified by this key with serial-number.log added. If the same value has been specified in `log_file_dir` and `trace_log_file_dir`, you need to specify a value different from `trace_log_file_prefix` for this key. If you specify the same value, the redirector will not run.	`isapi_redirect`
`log_file_size`	Specify the size for each redirector log file in bytes. Specify an integer value from 4096 to 16777216.	`4194304`
`log_level`	Specify the output level of redirector log files. You specify only one log level. You can specify `debug`, `info`, or `error`.	`error`
`prf_id`	Specify the character string that was specified in the PRF identifier when invoking the PRF daemon.	`PRF_ID`
`receive_client_timeout`	Specify in seconds the timeout period used when the POST data is received from the client. Specify an integer value from 60 to 3600 in multiples of 60 (seconds). If the specified value is not a multiple of 60, a value rounded off to a multiple of 60 becomes the timeout period.	`300`

Key name	Contents	Default value

`request_retry_count`
`}`
	Specify an integer (units: number of times) from 1 to 16 for the retry count to connect to the Web container when a request is sent and the retry count to send requests.
	The retry count also includes the first establishment of connection and the request sending process.
	When a timeout occurs, retry occurs in the following cases:
	• When a timeout occurs while connection is being established
	• When a timeout occurs while the request header is being sent
	After the above processes, if a timeout occurs while the request body is being sent, retry does not occur.
	If you set the retry count to an abnormal value, such as a value outside the range or a value that is not an integer, the default value is set.
		`3`

`send_timeout`	Specify an integer (units: seconds) from 0 to 3600 for the timeout period for sending requests. If you specify a non-numeric value, or a numeric value outside the range, a message will be output and the default value will be used. If you set the timeout as 0 or a time period longer than the resend timer of data transmission in TCP, the timeout value becomes the timeout value of TCP. In such a case, a message indicating that an invalid timeout value has been specified will not be output.	`100`
`trace_log`	Specify whether to output the trace log for the maintenance of redirector. Specify `true` if trace log is to be output and `false` if it is not to be output.	`true`
`trace_log_file_dir`	Specify the output destination directory for the maintenance trace log files. When coded as a relative path: Specify a directory name present in *Cosminexus-installation-directory*`\CC\web\redirector`. When coded as an absolute path: Specify the coded directory name. Note that you must specify write permission for the `Users` group in the access permission to the directory specified as the output destination#. If access permission is not set, the log file is not output. If the same value has been specified in `log_file_prefix` and `trace_log_file_prefix`, you need to specify a value different from log_file_dir for this key. If you specify the same value, the redirector will not run.	logs
`trace_log_file_num`	Specify the maximum number of maintenance trace log files. If this number is exceeded, the old log files are overwritten. Specify an integer value from 1 to 64.	`4`
`trace_log_file_pref` `ix`	Specify the prefix for the maintenance trace log file name. The actual log file name will be the value specified by this key with *serial-number*.log added. If the same value has been specified in log_file_dir and `trace_log_file_dir`, you need to specify a value different from `log_file_prefix` for this key. If you specify the same value, the redirector will not run.	`iis_rd_trac` `e`
`trace_log_file_size`	Specify the size for each maintenance trace log file in bytes. Specify an integer value from 4096 to 16777216.	`16777216`
`worker_file`	Specify the location and file name of the worker definition file. When specified as a relative path: Specify a file name present in *Cosminexus-installation-directory*`\CC\web\redirector`. When specified as an absolute path: Specify the coded file name.	`workers.pro` `perties`

Key name	Contents	Default value
`worker_mount_file`	Specify the location and file name of the mapping definition file. When specified as a relative path: Specify a file name present in *Cosminexus-installation-directory*`\CC\web\redirector`. When specified as an absolute path: Specify the coded file name.	`uriworkermap.properties`

#

For integration with Microsoft IIS:

For a new installation, the default log output destination directory does not exist. You either create a directory and grant access permission or grant access permission to the directory redirector that exists one level above.

Moreover, if the log output destination directory of the redirector is changed and only half of that path exists, you either grant the access permission to the lowest level of the existing directory or create the entire specified path, and then grant the access permission.

(5) Examples of coding

```
gateway_host=hostA
gateway_https_scheme=true
gateway_port=443

log_level=error
log_file_size=4194304
log_file_num=5
log_file_dir=logs
log_file_prefix=isapi_redirect
prf_id=prfid
trace_log=true
trace_log_file_size=16777216
trace_log_file_num=4
trace_log_file_dir=logs
trace_log_file_prefix=iis_rd_trace
receive_client_timeout=300
worker_file=workers.properties
worker_mount_file=uriworkermap.properties
```

(6) Precautions

If you change the user definition of the redirector by editing this file, you will need to restart the Web server. The changed definition will be applied after the Web server is restarted.

14.2.2 mod_jk.conf (Redirector action definition file for Cosminexus HTTP Server)

(1) Format

Specify the key as follows:

```
key-name value
```

Specification method

- The string up to the linefeed is a value.
- The line beginning with a hash mark (#) is a comment.
- Separate the key name and the value with a single-byte space. If you specify multiple values, separate them with single-byte spaces.
- If the specified value contains a space in the file path, you need to enclose the entire path within double quotation marks ("").
- The command coding methods and the formats for describing the types of characters that you can code must be compliant with Cosminexus HTTP Server specifications.
- The key name is not case sensitive.

(2) File storage location

- In Windows
  Cosminexus-installation-directory`\CC\web\redirector\mod_jk.conf`
- In UNIX
  `/opt/Cosminexus/CC/web/redirector/mod_jk.conf`

(3) Functionality

Define the action of the redirector.

(4) Specifiable keys

- Module definition

  You define the library for processing the communication between the Web container server and the Cosminexus HTTP Server.

  Format

  `LoadModule jk_module` *library-name*

  You specify the library name as an absolute path. You cannot specify multiple library names.

  Examples of specification

  In Windows

  `LoadModule jk_module "`*Cosminexus-installation-directory*`\CC\web\redirector\mod_jk.dll"`

  In UNIX

  `LoadModule jk_module /opt/Cosminexus/CC/web/redirector/mod_jk.so`

  Precautions

  You must code the key specifying the module definition before other keys.

- Mapping definition

  You define the URL pattern to be transferred to the Web container server through the request to Cosminexus HTTP Server.

  Format

  `JkMount` *URL-pattern worker-name*

  Code any one worker out of those specified in the `worker.list` of the `workers.properties`. You can code multiple combinations of the URL pattern and worker name. If you set an invalid value in this file, the operations may not produce the desired results. If you specify a URL pattern which does not start with a forward slash (/), an error message is output in the log when the redirector starts and the specified content becomes ignored.

If you specify a URL pattern containing an extension with a length of 0 characters (when the URL pattern ends with `/*.`), the message KDJE41041-W is output to the log and the specified content is ignored.

For details on the `JkMount` key, see *5.4 Distributing requests by the round-robin format* and *5.5 Distributing requests by the POST data size*.

- Redirector definition

  You can specify the following keys. If you specify an invalid value in these keys, the operations may not produce the desired results.

  The *Related information* column lists the reference locations for information related to the specified key. Note that *Application Server* is omitted from the manual names.

Key name	Contents	Default value	Related information
`JkConnectTimeout`	Specify the timeout value used for connecting to the Web container when a request is sent. Specify the timeout value as an integer (units: seconds) from 0 to 3600. If you specify a non-numeric value, or a numeric value outside the range, a message will be output and the default value will be used. If you set the timeout as 0 or a time period longer than the resend timer of data transmission in TCP, the timeout value becomes the timeout value of TCP. In such a case, a message indicating that an invalid timeout value has been specified will not be output.	`30`	
`JkGatewayHost`	Specify the host name or the IP address of the gateway. When requests without a Host header are redirected to files, such as the `welcome` file, the host name of the Location header URL becomes the specified value.	None	*5.10 Notification of gateway information to a Web container*
`JkGatewayHttpsScheme`	When a client request uses https as a scheme, and the scheme for a Web server will become http by using an SSL accelerator, specify `On`. If `On` is specified, https is assumed to be used as the scheme for requests sent to the Web server. If you specify `Off`, no action occurs.	`Off`	*5.10 Notification of gateway information to a Web container*
`JkGatewayPort`	Specify the port number of the gateway. If a request has no `Host` header and the request is to be redirected to a location such as a welcome file, the port number portion of the URL that is specified in the `Location` header becomes the specified value. Always specify this parameter together with `JkGatewayHost`. If you specify `JkGatewayHost` and omit this parameter, an access via http uses 80, and an access via hppts uses 443.	None	*5.10 Notification of gateway information to a Web container*
`JkLogFileDir`	Specify the output location directory of the log file. In Windows, specify either as a relative	• In Windows logs	

Key name	Contents	Default value	Related information
	path or as an absolute path. In UNIX, specify as an absolute path. When specified as a relative path: Specify a directory name present in *Cosminexus-installation-directory*`\CC\web\redirector`. When specified as an absolute path: Specify the coded directory name. Note that you must specify write permission to the execution account of Cosminexus HTTP Server in the access permission to the directory specified as the output destination#1. If access permission is not set, the log file is not output. If the same value has been specified in `JkLogFilePrefix` and `JkTraceLogFilePrefix`, you need to specify a value different from `JkTraceLogFileDir` in this key. If you specify the same value, the redirector will not run.	• In UNIX `/opt/Cosminexus/CC/web/redirector/logs`	
`JkLogFileNum`	Specify the maximum number of redirector log files. If this number is exceeded, the old log files are overwritten. Specify an integer value in the following range: • In Windows 1≤ `JkLogFileNum`≤ 16 • In UNIX 1≤ `JkLogFileNum`≤ 64	`5`	
`JkLogFilePrefix`	This is a prefix for the log file name. The actual log file name will be the value specified by this key with *serial-number*.log added. If the same value has been specified in `JkLogFileDir` and `JkTraceLogFileDir`, you need to specify a value different from `JkTraceLogFilePrefix` in this key. If you specify the same value, the redirector will not run.	`hws_redirect`	
`JkLogFileSize`	Specify the size for each redirector log file in bytes. Specify an integer value in the following range: • In Windows 4096≤ `JkLogFileSize`≤ 2147483647 • In UNIX 4096≤ `JkLogFileSize`≤ 16777216	`4194304`	
`JkLogLevel`	Specify the log output level of message logs for redirectors and of trace logs for maintenance. You specify only one log level. You can specify `debug`, `info`, `error` (default value), and `emerg`. You can specify `emerg` only in Windows.	`error`	

Key name	Contents	Default value	Related information
	If you specify a value other than those specified above, the operation will be carried out assuming that `error` has been specified.		
`JkModulePriority`	When registering an external module other than the redirector in the Cosminexus HTTP Server, specify the execution order of the redirector for the other external modules. The values that can be specified are as follows: • Integers from -10 to 30 • `REALLY_FIRST` (corresponds to integer value -10) • `FIRST` (corresponds to integer value 0) • `MIDDLE` (corresponds to integer value 10) • `LAST` (corresponds to integer value 20) • `REALLY_LAST` (corresponds to integer value 30) Smaller the value specified, higher the position in the execution order.	`FIRST`	
`JkOptions`	Specify whether to decode the requested URL. Note that this key is used for UNIX. `ForwardURICompatUnparsed` (default): Do not decode the requested URL. `ForwardURICompat`: Decode the requested URL. URL decoding was performed in version 02-00, therefore, when using URLs containing character strings that are converted due to URL decoding, specify `ForwardURICompat` only when URL decoding needs to be performed as in the case of version 02-00.	`ForwardURICompatUnparsed`	
`JkPrfId`	Specify the character string that was specified in the PRF identifier when invoking the PRF daemon.	`PRF_ID`	
`JkRequestRetryCount`	Specify an integer (units: number of times) from 1 to 16 for the retry count to connect to the Web container when a request is sent and the retry count to send requests. The retry count also includes the first establishment of connection and the request sending process. When a timeout occurs, retry occurs in the following cases: • When a timeout occurs while connection is being established • When a timeout occurs while the request header is being sent After the above processes, if a timeout occurs while the request body is being sent, retry does not occur. If you set the retry count to an abnormal value, such as a value outside the range or a value that is not an integer, the default value is set.	3	

Key name	Contents	Default value	Related information
`JkSendTimeout`	Specify an integer (units: seconds) from 0 to 3600 for the timeout period for sending requests. If you specify a non-numeric value, or a numeric value outside the range, a message will be output and the default value will be used. If you set the timeout as 0 or a time period longer than the resend timer of data transmission in TCP, the timeout value becomes the timeout value of TCP. In such a case, a message indicating that an invalid timeout value has been specified will not be output.	`100`	
`JkTraceLog`	Specify whether to output the trace log for the maintenance of redirector. Specify `On` (default value) if trace log is to be output and `Off` if it is not to be output.	`On`	
`JkTraceLogFileDir`	Specify the output destination directory for the maintenance trace log files. In Windows, specify either as a relative path or as an absolute path. In UNIX, specify as an absolute path. When coded as a relative path: Specify a directory name present in *Cosminexus-installation-directory*`\CC\web\redirector`. When coded as an absolute path: Specify the coded directory name. If the same value has been specified in `JkLogFilePrefix` and `JkTraceLogFilePrefix`, you need to specify a value different from `JkLogFileDir` in this key. If you specify the same value, the redirector will not run.	• In Windows logs • In UNIX `/opt/Cosminexus/CC/web/redirector/logs`	
`JkTraceLogFileNum`	Specify the maximum number of maintenance trace log files. • In Windows 1 to 16 • In UNIX 1 to 64 If this number is exceeded, the old log files are overwritten.	`4`	
`JkTraceLogFilePrefix`	Specify the prefix for the maintenance trace log file name. The actual log file name will be the value specified by this key with serial-number.log added. If the same value has been specified in `JkLogFileDir` and `JkTraceLogFileDir`, you need to specify a value different from `JkLogFilePrefix` in this key. If you specify the same value, the redirector will not run.	`hws_rd_trace`	

Key name	Contents	Default value	Related information
`JkTraceLogFileSize`	Specify the size for each maintenance trace log file as an integer value (units: bytes). • In Windows 4096 to 2147483647 • In UNIX 4096 to 16777216	`16777216`	
`JkTranslateBackcompat`	The key for compatibility with versions older than 05-05. Specify whether the `translate_handler` function of the module executed after the redirector is to be invoked from the Cosminexus HTTP Server, if a request sent to the Cosminexus HTTP Server is a URL pattern transferred to the Web container. If you specify `On`, the `translate_handler` function of the module executed after the redirector is invoked. If you specify `Off`, the `translate_handler` function of the module executed after the redirector is not invoked.	`Off`	
`JkWorkersFile`	Specify the file name of the worker definition file. In Windows, specify either as a relative path or as an absolute path. In UNIX, specify as an absolute path. When specified as a relative path: Specify a file name present in *Cosminexus-installation-directory*`\CC\web\redirector`. When specified as an absolute path: Specify the coded file name.	• In Windows `workers.properties` • In UNIX `/opt/Cosminexus/CC/web/redirector/workers.properties`#2	

Note:

When you specify multiple keys, the last specified value becomes valid.

#1

For integration with Cosminexus HTTP Server:

For a new installation, the default log output destination directory does not exist. You either create a directory and grant the access permission or grant the access permission to the directory redirector that exists one level above.

Moreover, if the log output destination directory of the redirector is changed and only half of that path exists, you either grant the access permission to the lowest level of the existing directory or create the entire specified path, and then grant the access permission.

#2

If `/opt/Cosminexus/CC/web/redirector/workers.properties` does not exist, Cosminexus HTTP Server does not start.

(5) Examples of coding

- In Windows

```
LoadModule jk_module "C:\Program Files\Hitachi\Cosminexus\CC\web\redirecto
r\mod_jk.dll"#

JkGatewayHost hostA
```

```
JkGatewayHttpsScheme On
JkGatewayPort 443

JkLogLevel error
JkLogFileSize 4194304
JkLogFileNum 5
JkLogFileDir logs
JkLogFilePrefix hws_redirect
JkPrfId prfid
JkTraceLog On
JkTraceLogFileSize 16777216
JkTraceLogFileNum 4
JkTraceLogFileDir logs
JkTraceLogFilePrefix hws_rd_trace
JkTranslateBackcompat Off
JkWorkersFile workers.properties

JkMount /examples/* worker1
```

#

Specify `LoadModule` in one line in the file.

- In UNIX

```
LoadModule jk_module /opt/Cosminexus/CC/web/redirector/mod_jk.so
JkWorkersFile /opt/Cosminexus/CC/web/redirector/workers.properties

JkLogLevel error
JkLogFileSize 4194304
JkLogFileNum 5
JkLogFileDir /opt/Cosminexus/CC/web/redirector/logs
JkLogFilePrefix hws_redirect

JkTraceLog On
JkTraceLogFileSize 16777216
JkTraceLogFileNum 4
JkTraceLogFileDir /opt/Cosminexus/CC/web/redirector/logs
JkTraceLogFilePrefix hws_rd_trace

JkMount /examples/* worker1
```

(6) Precautions

- In Windows, you need to restart the Web server to change the user definition of the redirector. The changed definition will be applied after the Web server is restarted.
- In UNIX, you need to perform the following operations to apply the changed contents once the user definition of the redirector is changed:

When you change the file size or the number of files:

1. Stop the Web server.
2. Either move or delete the log files and the management file used by HNTRLib.
   <Management file used by HNTRLib>
   In the case of message log file: *JkLogFilePrefix-settings*`.mm`
   In the case of maintenance trace log file: *JkTraceLogFilePrefix-settings*`.mm`

3. Start the Web server.

Note that the default storage location of management files used in HNTRLib is as follows:
Cosminexus-installation-directory`/CC/web/redirector/logs/mmap`

When you do not change the file size or the number of files:
Restart the Web server.

14.2.3 uriworkermap.properties (Mapping definition file for Microsoft IIS)

(1) Format

Specify the key as follows:

```
key-name=value
```

Specification method

- The string up to the linefeed is a value.
- The line beginning with a hash mark (#) is a comment.
- If you define a line without a value, the line is ignored. The parameters not defined as valid key names are also ignored even if defined in the action definition file.
- Up to 1023 characters are valid as *key-name*=*value*. The part exceeding this number is truncated.

(2) File storage location

Cosminexus-installation-directory`\CC\web\redirector\uriworkermap.properties`

(3) Functionality

This mapping definition file defines the URL pattern to be transferred to the Web container server through the request to Microsoft IIS.

(4) Specifiable keys

Code any one worker out of those specified in the `worker.list` of the `workers.properties`. You can code multiple combinations of the URL pattern and worker name. If you set an invalid value in this file, the operations may not produce the desired results. If you specify a URL pattern which does not start with a forward slash (`/`), an error message is output in the log when the redirector starts and the specified content becomes ignored.

If you specify a URL pattern containing an extension with a length of 0 characters (when the URL pattern ends with `/*.`), the message KDJE41041-W is output to the log and the specified content is ignored.

```
URL-pattern=worker-name
```

(5) Examples of coding

```
/examples/*=worker1
```

(6) Precautions

If you change the user definition of the redirector, you will need to restart the Web server. The changed definition will be applied after the Web server is restarted.

14.2.4 workers.properties (Worker definition file)

(1) Format

Specify the key as follows:

```
key-name=value
```

Specification method

- The string up to the linefeed is a value.
- The line beginning with a hash mark (#) is a comment.
- If you define a line without a value, the line is ignored. The parameters not defined as valid key names are also ignored even if defined in the action definition file.
- Up to 1023 characters are valid as *key-name=value*. The part exceeding this number is truncated.
- Single-byte alphanumeric characters, underscores (_), and hyphens (`-`) can be used in the worker name. The operations cannot be guaranteed if you specify other characters.

(2) File storage location

- In Windows
  Cosminexus-installation-directory`\CC\web\redirector\workers.properties`
- In UNIX
  `/opt/Cosminexus/CC/web/redirector/workers.properties`

(3) Functionality

The worker definition file defines the workers, sets the parameters for each worker, and defines the actions of the redirector.

(4) Specifiable keys

The keys that you can specify in the worker definition file and the parameters defined for each worker are explained below:

(a) Keys specifiable in the worker definition file

These keys define workers, and the parameters for each worker. If an invalid value is specified for this key, the operations might not execute properly.

Key name	Contents	Default value
`worker.list`	Specify the list of worker names. In the case of multiple names, demarcate with a comma (`,`). You need to specify at least one worker name.	None

Key name	Contents	Default value
	Specify the worker name specified in `mod_jk.conf` (redirector action definition file for HTTP Server), or `uriworkermap.properties` (mapping definition file for Microsoft IIS).	
`worker.`*worker-name*`.`*parameter*	Specify the parameters defined for each worker. Set the parameters for each worker coded in `worker.list`. For details on the defined parameters, see *(b) Parameters defined for each worker*.	None

(b) Parameters defined for each worker

Parameters that you can define	Contents	Default value	Related information
`worker.`*worker-name*`.balanced_workers`	Specify the list of workers for which load balancing is to be performed. In the case of multiple names, demarcate with a comma (`,`).	None	
`worker.`*worker-name*`.cachesize`	Specify an integer value from 1 to 2147483647 for the number of connections to the workers re-used in the redirector. Note that this parameter is used in Windows. When the number of connections to the workers is within the scope of the specified value, the connections remain in the redirector and are re-used for communication with the applicable workers without being released until the J2EE server at the connection destination terminates. If the multiplicity of requests exceeds the set value, for only those requests that exceed the set value, establish and release connections to workers for each request. This value uses up memory as per the following formula: (Expression) Memory consumption = `(worker.`*worker-name*`.cachesize value)`× `10KB`	`64`	
`worker.`*worker-name*`.default_worker`	Specify worker name of the default worker. If you specify a worker same as the worker specified in the `POST` request transfer destination worker, the requests satisfying the distribution conditions as per the `POST` data size and requests satisfying the conditions for the default worker are transferred to the specified worker. Note that if you specify a worker that is not specified in the `POST` request transfer destination worker in this parameter, and if `worker.`*worker-name*`.post_data` is specified in that worker, the definition of `worker.`*worker-name*`.post_data` is ignored. The null characters (space, tab, or form feed) before and after the worker name are ignored. If you omit this parameter or specify a null string, an error is returned for the request for which a worker satisfying the transfer conditions does not exist.	None	
`worker.`*worker-name*`.delegate_error_code`	Specify the error status code that uses the delegation functionality of the error page. # If you want to specify multiple worker names, demarcate with a comma (`,`).	None	
`worker.`*worker-name*`.host`#2	Specify the host name or the IP address of the worker.	None	

Parameters that you can define	Contents	Default value	Related information
`worker.`*worker-name*`.lbfactor`	Specifies the load balancing value of a request. Set a value greater than 0. You can also specify a decimal value.	`1`	
`worker.`*worker-name*`.port`#2	Specify an integer value from 1 to 65535 for the worker port number. You cannot specify a port number that is already being used or secured for another application.	None	
`worker.`*worker-name*`.post_data`	Specify a value adding 1 to the maximum value of the Content-Length header value for the requests to be transferred to the worker specified in *worker-name* as follows: • An integer value from 1 to 2147483648 (unit: bytes) • Value adding `k` or `K` to an integer value from 1 to 2097152 (units: kilobytes) • Value adding `m` or `M` to an integer value from 1 to 2048 (units: megabytes) The requests in which the value of the Content-Length header is lesser than the specified value is transferred to a worker specified in *worker-name*. If multiple workers are specified in the `worker.`*worker-name*`.post_size_workers` parameter, the request is transferred to a worker in which the request Content-Length header value is lesser than the specified value, and in which the value is the smallest specified value. If you specify the value in the `worker.`*POST-request-distribution-worker's-worker-name*`.post_size_workers` parameter, do not specify the same value as the other workers. The null characters (space, tab, or form feed) before and after the value are ignored.	None	
`worker.`*worker-name*`.post_size_workers`	Specify the list of worker names of the `POST` request transfer destination workers. If you want to specify multiple worker names, demarcate with a comma (`,`). You cannot specify the same worker name. The null characters (space, tab, or form feed) before and after the worker name are ignored.	None	
`worker.`*worker-name*`.receive_timeout`	Specify the communication timeout value. Specify an integer value (units: seconds) from 0 to 3600 for the time period to await the response data. If you specify 0, the system continues to wait until a response is received and a communication timeout does not occur.	`3600`	*5.6 Communication timeout (Web server integration)*
`worker.`*worker-name*`.type`	Specify the worker type from the types shown below. For details on the parameters that you can set in each type, see *(c) Parameters defined for each worker.worker-name.type*. You need to specify these parameters for each worker. `ajp13`: This worker transfers requests to the Web container server running by an external process.	None	

Parameters that you can define	Contents	Default value	Related information
	`ajp12`: This worker maintains compatibility with older versions. This worker runs assuming that ajp13 is specified. `lb`: This worker has the load balancing functionality based on the round robin process. `post_size_lb`: This is a `POST` request distribution worker. You can use this worker only when you use Cosminexus HTTP Server.		

#1

The specifiable codes are mentioned in comments. Remove the comments if required.

#2

You can also specify the host name and port number of the same Web container for multiple workers.

(c) Parameters defined for each worker.worker-name.type

Parameters that you can define	Worker types		
	ajp13	lb	post_size_lb
`worker.`*worker-name*`.balanced_workers`	N	Y	N
`worker.`*worker-name*`.cachesize`[#1]	O	N	N
`worker.`*worker-name*`.default_worker`	N	N	O
`worker.`*worker-name*`.delegate_error_code`	O	N	N
`worker.`*worker-name*`.host`	Y	N	N
`worker.`*worker-name*`.lbfactor`	O	N	N
`worker.`*worker-name*`.port`	Y	N	N
`worker.`*worker-name*`.post_data`	N/Y[#2]	N	N
`worker.`*worker-name*`.post_size_workers`	N	N	Y
`worker.`*worker-name*`.receive_timeout`	O	N	N

Legend:

Y: Always specify.

O: Specification is optional.

N: Cannot be specified.

Notes

The parameters that you can specify in ajp12 are same as in ajp13.

In UNIX, if you do not specify a value for the mandatory items or if the specified value is invalid, the Cosminexus HTTP Server does not start.

#1

Valid only in Windows. In UNIX, the parameter is ignored.

#2

Mandatory for the POST request transfer destination worker.

(5) Examples of coding

```
worker.list=worker1
worker.worker1.port=8007
worker.worker1.host=localhost
worker.worker1.type=ajp13
#worker.worker1.cachesize=64
#worker.worker1.receive_timeout=3600
#worker.worker1.delegate_error_code=400,401,402,403,404,405,406,407,408,409,
410,411,412,413,414,415,416,417,422,423,424,500,501,502,503,504,505,507,510

#-----------------------------------------------------------------
# Example setting for Loadbalancer.
#-----------------------------------------------------------------
#worker.list=loadbalancer1
#
#worker.loadbalancer1.type=lb
#worker.loadbalancer1.balanced_workers=worker1,worker2
#
#worker.worker1.port=8007
#worker.worker1.host=host1
#worker.worker1.type=ajp13
#worker.worker1.cachesize=64
#worker.worker1.lbfactor=1
#worker.worker1.receive_timeout=3600
#worker.worker1.delegate_error_code=400,401,402,403,404,405,406,407,408,409,
410,411,412,413,414,415,416,417,422,423,424,500,501,502,503,504,505,507,510

#
#worker.worker2.port=8007
#worker.worker2.host=host2
#worker.worker2.type=ajp13
#worker.worker2.cachesize=64
#worker.worker2.lbfactor=1
#worker.worker2.receive_timeout=3600
#worker.worker2.delegate_error_code=400,401,402,403,404,405,406,407,408,409,
410,411,412,413,414,415,416,417,422,423,424,500,501,502,503,504,505,507,510
#-----------------------------------------------------------------
# Example setting for post data size based worker.
#-----------------------------------------------------------------
#worker.list=postsizelb1#worker.postsizelb1.type=post_size_lb
#worker.postsizelb1.post_size_workers=worker1,worker2
#worker.postsizelb1.default_worker=worker1
#
#worker.worker1.port=8007
#worker.worker1.host=host1
#worker.worker1.type=ajp13
#worker.worker1.post_data=100m
#worker.worker1.receive_timeout=3600
#worker.worker1.delegate_error_code=400,401,402,403,404,405,406,407,408,409,
410,411,412,413,414,415,416,417,422,423,424,500,501,502,503,504,505,507,510
#
#worker.worker2.port=8007
#worker.worker2.host=host2
#worker.worker2.type=ajp13
#worker.worker2.post_data=2048m
#worker.worker2.receive_timeout=3600
```

```
#worker.worker2.delegate_error_code=400,401,402,403,404,405,406,407,408,409,
410,411,412,413,414,415,416,417,422,423,424,500,501,502,503,504,505,507,510
```

(6) Precautions

If you change the user definition of the redirector by editing this file, you will need to restart the Web server. The changed definition will be applied after the Web server is restarted.

15 Performance Analysis Trace

The performance analysis trace functionality collects the performance analysis information (trace information) output by each functionality of the application server when the server processes a request from a client, and collects performance analysis information (trace information) output by the application processing.

Based on this information, you can analyze the processing performance of the system and applications. This chapter describes how to analyze performance of the system and applications by using performance analysis traces. The following describes the items that differ between V9 compatibility mode and recommended mode. For information that is not provided in this chapter, see *7. Performance Analysis by Using Trace Based Performance Analysis* in the manual *uCosminexus Application Server Maintenance and Migration Guide*. For details on the points when performance analysis information is collected and the scope of information collection (PRF trace collection levels) by using performance analysis trace, see the sections from *15.5 Overview on trace collection points and PRF trace collection levels of performance analysis trace* through *15.13 Trace collection points of CDI*.

15.1 Overview of performance analysis traces

The location for storing the trace based performance analysis is as follows:

- In Windows
  Environment-variable-PRFSPOOL-settings-directory`\utt\prf\`*PRF-identifier*
- In UNIX
  `$PRFSPOOL/utt/prf/`*PRF-identifier*

For details on how to collect a performance analysis trace, see *7. Performance Analysis by Using Trace Based Performance Analysis* in the manual *uCosminexus Application Server Maintenance and Migration Guide*. The session trace may also be output to the trace based performance analysis.

15.2 Overview of the trace based performance analysis of Application Server

The redirector and the Web container that output the trace are referred to as the *function layer*. In the trace based performance analysis, trace information is output at the entrance and the exit of the following function layer. Moreover, the trace information is output as and when necessary for each process that affects the performance among the processes of the function layers. The following table shows the application execution environments and the applicable function layers that differ in V9 compatibility mode and in recommended mode.

Table 15–1: Execution environment of the application and the applicable function layer

Function layer	Execution environment of the application	
	Execution environment of the J2EE application	Execution environment of batch applications
Redirector	Y	--
Cosminexus JPA Provider	Y	--

Legend:

Y: Applicable

--: Not applicable

Reference note

- Key information (route application information) is added to the PRF trace. For a J2EE application, information acquired by the redirector or EJB client is added.
- When a timeout occurs during the transaction of the J2EE application or in the redirector in receiving the response, you can use the root application information that is output in the trace based performance analysis to identify the transaction and the request that timed out.

15.3 Output information of the trace based performance analysis file (for the trace based performance analysis)

The trace based performance analysis collects the trace information for the functionality layer.

The following table shows the information, output by each function layer, that differs in V9 compatibility mode and in recommended mode.

Table 15–2: Information output to the trace based performance analysis file (for the trace based performance analysis)

Trace information header	Description	Range of values
Thread	Thread ID and hash value of the thread in the process that acquired the trace information[#1].	Thread ID: A decimal number up to twenty digits is output. Hash value: A decimal number up to ten digits is output.
ProcessName	Process name	A string[#2] displaying the process up to 32 characters is output.
OPT[#3]	Additional information for each collection point.	A hexadecimal number string up to 514 characters is output.

#1

There are cases when the hash value of a thread is not output in the trace information acquired by the redirector and CTM.

#2

For a redirector, the process name is determined as follows:

- **In the case of redirector**

  If Hitachi Web Server is used as the Web server, "RD-*Web-server-waits-to-receive-request-on-this-port-number*" will be the process name. If Microsoft IIS is used, the process name will be `Redirector`.

#3

Some function layers have trace collection points that output the entry time to `OPT`. The entry time is the time at which the entry trace corresponding to the trace of the trace collection point is output. For example, in the case of trace collection point of a redirector, the entry time output at the trace collection point when the process of sending HTTP response body information to a Web server finishes (`0x8104`), will be the time to start sending the HTTP response body information from a Web container (`0x8004`).

Note that the entry time is output in 16 bytes that is the time elapsed from 01/01/1970 00:00:00. The first 8 bytes of the value are seconds and the last 8 bytes of the value are microseconds. However, in the case of trace based performance analysis of redirector, the last 8 bytes are milli seconds.

15.4 Analyzing the Response Time of a Web Server

You can analyze the processing performance of Application Server based on the trace information output from the function layer such as the redirector and Web container, in the series of processes from the client to EIS such as databases, until the processing result is returned to the client.

This section describes how to analyze the time from when a Web server received a request from a client to when the response was returned to the client, with an example.

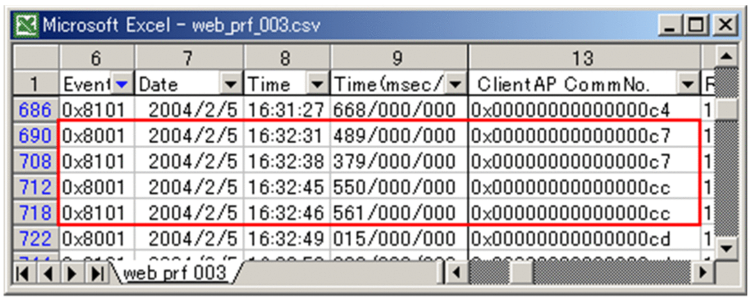
You use the event ID `0x8001` and `0x8101` assigned to the redirector as a key to perform filtering of trace based performance analysis files collected in the Web server. The following table describes the trace collection points which are indicated by the event IDs used in filtering.

Table 15–3: Trace collection points indicated by the event IDs used in filtering

Event ID	Trace acquisition points
0x8001	Sending the request header information to a Web container
0x8101	Receiving the response completion notification from the Web container

The following is an example of filtering the trace based performance analysis, by using the event IDs `0x8001` and `0x8101` as a key.

Figure 15–1: Example of filtered trace based performance analysis file



	6	7	8	9	13	
1	Event	Date	Time	Time(msec/	ClientAP CommNo.	F
686	0x8101	2004/2/5	16:31:27	668/000/000	0x00000000000000c4	1
690	0x8001	2004/2/5	16:32:31	489/000/000	0x00000000000000c7	1
708	0x8101	2004/2/5	16:32:38	379/000/000	0x00000000000000c7	1
712	0x8001	2004/2/5	16:32:45	550/000/000	0x00000000000000cc	1
718	0x8101	2004/2/5	16:32:46	561/000/000	0x00000000000000cc	1
722	0x8001	2004/2/5	16:32:49	015/000/000	0x00000000000000cd	1

You can analyze the response time of each request from the trace collection date and time of `0x8001` and `0x8101`. In the example shown in Figure 15-1, if you compare the response time of requests in which communication number of the client application information are `0x00000000000000c7` and `0x00000000000000cc`, you will observe that the time taken for processing the request `0x00000000000000c7` is more.

15.4.1 Identifying the Request for Which Timeout Occurred

This section describes how to identify a request in which timeout occurred by using the trace based performance analysis, if a timeout has occurred while receiving the response in the redirector.

If a timeout occurs while receiving a response in the redirector, the KDJE41019-E message is output. The following information is output in this message:

- URI of the request
- IP address of the host in which the Web container used for communication with the Web server is running
- Port number of the Web container used for the communication with the Web server
- File descriptor of the Web container used for communication with the Web server

- Root application information of the trace based performance analysis

You can check where the timeout has occurred in a request in the trace based performance analysis, by comparing the root application information of the trace based performance analysis output in the message with the root application information output in the trace based performance analysis file.

You can check the request that is timed out with the trace information of the event IDs, described in the table below, of the request processing in which the corresponding root application information is output. Identify the corresponding request with the URI output to the operation name.

Table 15–4: Trace based performance analysis that you can use to check the request that is timed out

Event ID	Description
0x8000	This information is output after getting a request for processing the request. The URI of the request is output as an operation name.
0x8100	This information is output after the processing of the redirector finishes. The URI of the request is output as an operation name.

15.4.2 Investigating the Log Using the Root Application Information

If the application API is used in a J2EE application or batch application, the character string expression of root application information of the performance analysis information can be output to the log file at any time.

The character string expression of the root application information is output to the following format (maximum 48 characters):

```
IP address / process ID / communication number
(Example: 10.209.15.130/1234/0x0000000000000001)
```

If the root application information is output to a log file using an API, you compare the log file with the trace based performance analysis file to investigate.

When a new request is received in the Web container, a new root application information is assigned to the single request. The following table describes the trace points to which the new root application information is assigned.

Table 15–5: Trace point to which the new root application information is assigned

Conditions	Event ID (process contents)	Process contents
When using the in-process HTTP server	0x8211	When acquiring a request, when completing the request header analysis
When connecting to a Web server#	0x8200	When acquiring a request, when completing the request header analysis

#

When connecting to a Web server, the root application information acquired by the redirector is assigned. If route application information is not issued because the redirector failed or a failure occurred in loading the PRF trace output library used by the redirector, new route application information is assigned in the Web container.

Moreover, in the following trace collection point, IPaddress/process ID/communication number might be output as `0.0.0. 0/0/0x0000000000000000`.

- `0x8212`

  When data is read from Web client where the in-process HTTP server is used.
- `0x8312`

  When data is written to the Web client where the in-process HTTP server is used.

In the following cases the IP-address/process-ID/communication-number is output as `0.0.0. 0/0/0x0000000000000000`:

- If an HTTP request header is received
- If an incorrect data that is not an HTTP request is received
- If an exception occurs during processing of the request

Note that the APIs used to output the character string expression of the root application information is the `getRootApInfo` method of the `CprfTrace` class, provided in the `com.hitachi.software.ejb.application.prf` package.

For details on implementing acquisition of route application information when developing a J2EE application or batch application, see *7.7.9 Investigation about the location of the problem associated to the trace based performance analysis file and thread dump* in the manual *uCosminexus Application Server Maintenance and Migration Guide*. For details on the APIs, see the *uCosminexus Application Server API Reference Guide*.

15.5 Overview on trace collection points and PRF trace collection levels of performance analysis trace

This section describes the trace collection points and PRF trace collection levels.

15.5.1 Trace collection point

The trace collection points are broadly classified into trace collection during the startup and termination of a J2EE server, trace collection during processing in each function layer, and trace collection during the startup and termination of application methods.

(1) Trace collection during the startup and termination of the J2EE server

The trace information can be collected when the startup processing of the J2EE server ends, and when the termination processing of the J2EE server starts. The event IDs that can be acquired, and the references are as follows:

- Event ID
  `0x8FFE` to `0x8FFF`
- Reference
  For details on trace collection when a J2EE server starts or terminates, see *8.22 Trace collection points when a J2EE server is started or terminated* in the manual *uCosminexus Application Server Maintenance and Migration Guide*.

(2) Trace collection in each function layer

The following table describes the correspondence between the event IDs that can be acquired and the function layers.

Table 15–6: Event IDs that can be acquired and function layers

Event ID	Function layer	No. in the figures#	Reference location
0x1101 to 0x1102 0x1301 to 0x1302 0x1401 to 0x1406 0x2002 to 0x2003 0x2101 to 0x2104 0x3000 to 0x3008	CTM	5	*8.3* in the *uCosminexus Application Server Maintenance and Migration Guide*
0x8000 to 0x8004 0x8100 to 0x8104	Redirector	1	*15.6*
0x8200 to 0x8203 0x8206 to 0x8225 0x8300 to 0x8303 0x8306 to 0x8307 0x8311 to 0x8325 0x8E01 to 0x8E06	Web container	2	*15.7*, *15.8*, *15.9*, *15.10*
0x8401 to 0x840A 0x8425 to 0x8428 0x842D to 0x8434 0x8453 to 0x8454	EJB container	6	*8.8* in the *uCosminexus Application Server*

Event ID	Function layer	No. in the figures#	Reference location
0x8460 to 0x846D 0x8470 to 0x8477 0x8490 to 0x8491 0x84A0 to 0x84D9 0x8C41			*Maintenance and Migration Guide*
0x8603 to 0x861C	JNDI	4	*8.9* in the *uCosminexus Application Server Maintenance and Migration Guide*
0x8435 to 0x843F 0x8811 to 0x8820 0x8C41	JTA	7	*8.10* in the *uCosminexus Application Server Maintenance and Migration Guide*
0x8B00 to 0x8B01 0x8B80 to 0x8B89 0x8B8A to 0x8C03 0x8C10 to 0x8C13 0x8C20 to 0x8C41 0x8C60 to 0x8C65 0x8C80 to 0x8C93 0x8CC0 to 0x8CD9 0x8D00 to 0x8D19 0x8D60 to 0x8D63 0x8D80 to 0x8D89 0x8D8A to 0x8D8F 0x8D90 to 0x8D99	DB Connector and JCA container	8	*8.11* in the *uCosminexus Application Server Maintenance and Migration Guide*
0x8E01 to 0x8E06	RMI	3	*8.12* in the *uCosminexus Application Server Maintenance and Migration Guide*
0x9400 to 0x9413	OTS	9	*8.13* in the *uCosminexus Application Server Maintenance and Migration Guide*
0x9C00 to 0x9C03	Standard output, standard error output, and user log	--	*8.14* in the *uCosminexus Application Server Maintenance and Migration Guide*
0x9D00, 0x9D01	DI	10	*8.15* in the *uCosminexus Application Server*

Event ID	Function layer	No. in the figures#	Reference location
			Maintenance and Migration Guide
0xA100, 0xA101	Batch application execution functionality	11	*8.16* in the *uCosminexus Application Server Maintenance and Migration Guide*
0xA500 to 0xA5AF	JPA	12	*15.11*
0xA300 to 0xA305 0xA310 to 0xA315 0xA320 to 0xA323 0xA330 to 0xA331 0xA340 to 0xA363 0xA370 to 0xA381 0xA390 to 0xA3C3	Cosminexus JPA provider	13	*15.12*
0x842F 0x8430 to 0x8432 0x8825 0x8826 0x8B86, 0x8B87 0x8B8A to 0x8B93 0xAA00 to 0xAA06 0xAA08 to 0xAA0D 0xAA10 to 0xAA19	TP1 inbound integrated function	14	*8.17* in the *uCosminexus Application Server Maintenance and Migration Guide*
0xA600 to 0xA60F 0xA610 to 0xA619 0xA61E, 0xA61F 0xA620 to 0xA62F 0xA630 to 0xA63F 0xA640 to 0xA64F 0xA650 to 0xA65F 0xA660 to 0xA667 0xA61A, 0xA61B 0xA668 to 0xA66F 0xA670 to 0xA67F 0xA686 to 0xA68D 0xA692 to 0xA69B 0xA69E, 0xA69F 0xA6A0, 0xA6A1 0xA6A6 to 0xA6AB	Cosminexus JMS Provider	15	*8.18* in the *uCosminexus Application Server Maintenance and Migration Guide*
0xAD00 to 0xAD15 0xAD80 to 0xAD93	JavaMail	16	*8.19* in the *uCosminexus Application Server Maintenance and Migration Guide*
0xAF00 to 0xAF13	JSF	17	*8.20* in the *uCosminexus Application*

Event ID	Function layer	No. in the figures#	Reference location
			Server Maintenance and Migration Guide
0xb002 to 0xb009	CDI	18	*15.13*

Legend:

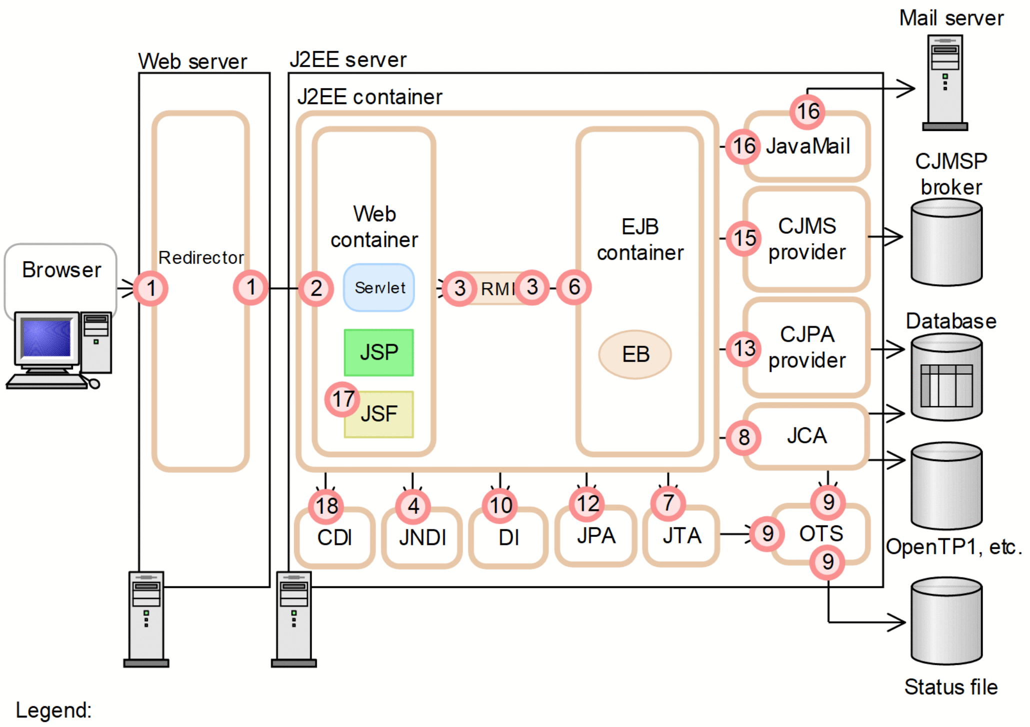
--: Not applicable

#

Corresponds to the numbers in *Figure 15-2* through *Figure 15-5*.

The following figures show the function layers for which the PRF trace is output, and the trace collection points for each system configuration.

Figure 15–2: Function layers and trace collection points (in the case of Web client configuration)



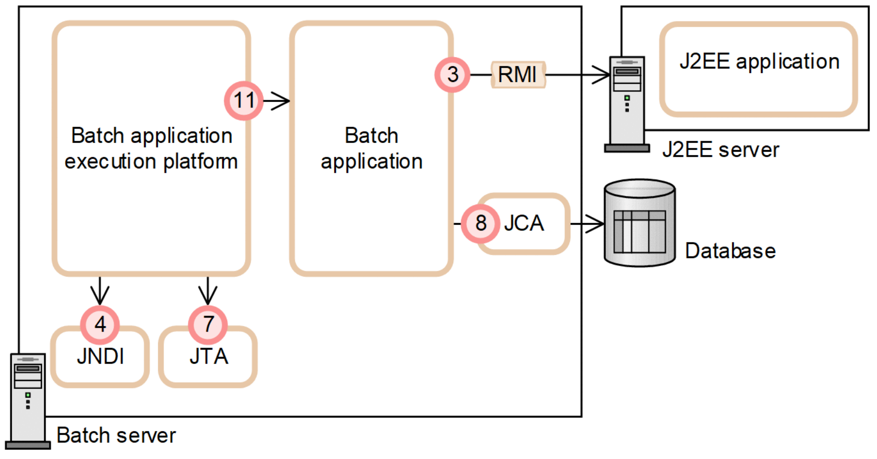
Legend:

: Functionality layer for which the PRF trace is output

● : Trace collection point

EB : Enterprise Bean

Figure 15–3: Function layers and trace collection points (in the case of a system for executing batch applications)

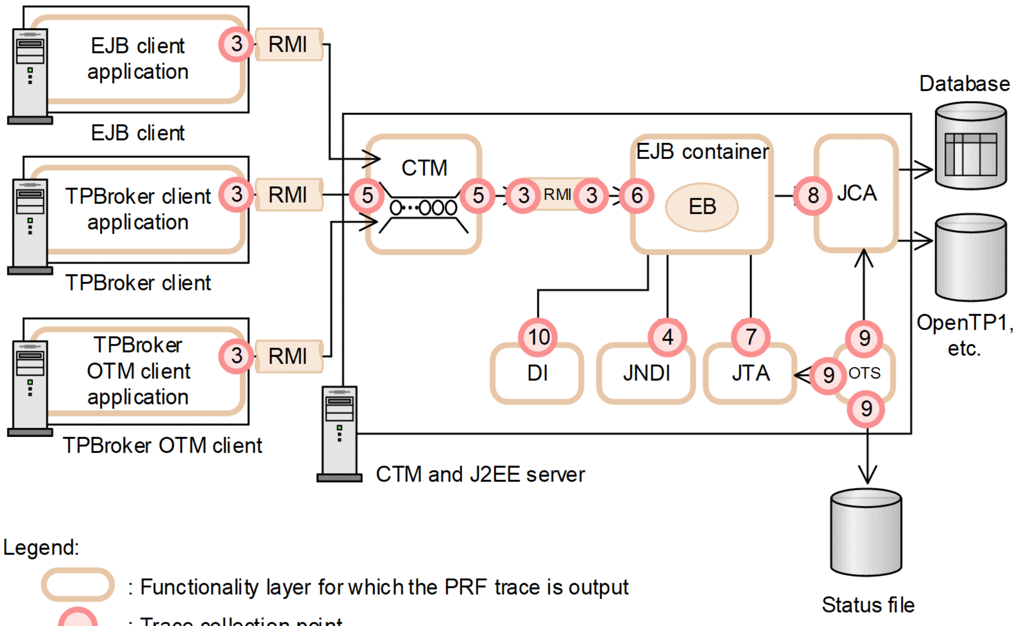


Legend:

▭ : Functionality layer for which the PRF trace is output

● : Trace collection point

Figure 15–4: Function layers and trace collection points (in the case of EJB client, TPBroker client, or TPBroker OTM client configuration (CTM usage))



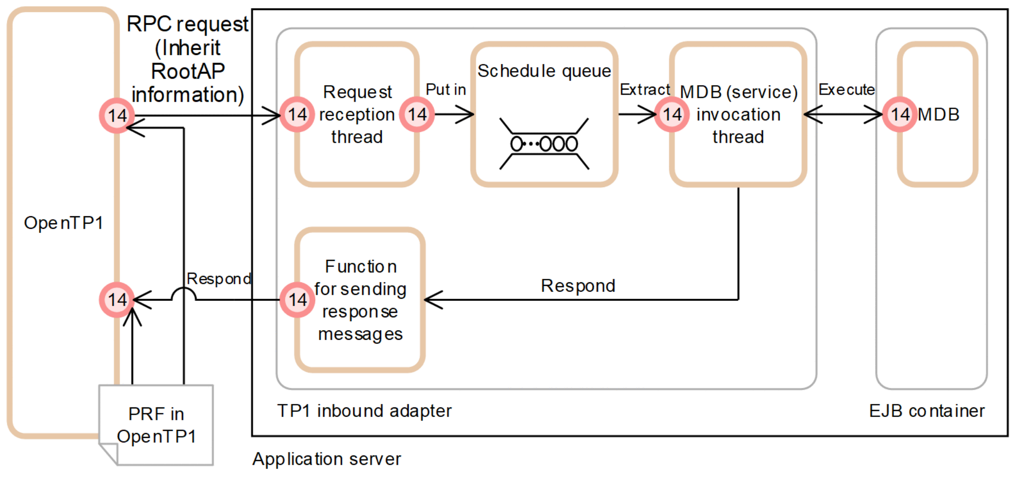
Legend:

▭ : Functionality layer for which the PRF trace is output

● : Trace collection point

EB : Enterprise Bean

Figure 15–5: Functionality layers and trace collection points (In the TP1 inbound integrated function)



Legend:

: Functionality layer for which the PRF trace is output

: Trace collection point

The trace collection points are divided in detail in each function layer, and the PRF trace collection level differs depending on the trace collection point. For details about trace collection points of each function layer, and the PRF trace collection level, see the references described in *Table 15-6*.

Reference note

Apart from the function layers described in *Table 15-6*, the PRF trace can be collected for some Application Server processes, component software, and related programs as well.

The following table describes the correspondence between the function layers other than those described in *Table 15-6* for which the PRF trace can be collected, and their event IDs.

Table 15–7: Correspondence between event IDs that can collect PRF trace and function layers other than those shown in the preceding table (Event IDs that can be acquired and function layers)

Event ID	Function layer
0x9000 to 0x90FF 0xA400 to 0xA4FF	Cosminexus Web Services - Base
0x9100 to 0x91FF	• uCosminexus TP1 Connector • TP1/Client/J
0x9200 to 0x92FF	TP1/MQ Access
0x9300 to 0x93FF	Cosminexus RM
0x9800 to 0x9B6E	HCSC server
0x9E00 to 0x9EFF	Service Coordinator Interactive Workflow
0x9F00 to 0x9FFF	HCSC server (Object Access adapter)
0xA000 to 0xA0FF	HCSC server (File adapter)
0xA200 to 0xA2FF	HCSC server (Message Queue adapter)

Event ID	Function layer
0xAB00 to 0xABFF 0xAC00 to 0xACFF	HCSC server (FTP adapter)
0xA400 to 0xA4FF	JAX-WS Engine
0xE000 to 0xE0FF	Elastic Application Data store

(3) Trace collection during the startup and termination of application methods

You can collect the trace information when an application method starts and terminates. The trace information that you can collect is as follows:

- Method start and termination time
- Identity ID
- Package name, class name, method name
- Line number of the last line executed by the method
- Class name of the exception or error that occurs

For details on trace information, see *8.23 Trace collection points of an application* in the *uCosminexus Application Server Maintenance and Migration Guide*.

(4) Return codes for each trace

For an entrance trace, the return code of each trace is always output as `0`.

For an exit trace and the trace after invocation, the return codes are output as follows:

Normal termination: `0`

Abnormal termination: Other than 0

15.5.2 PRF trace collection level

In the trace based performance analysis, you can specify the following four types of PRF trace collection levels to output the trace. The number of trace collection points differs depending upon the PRF trace level used. For details about the trace collection points, and PRF trace collection levels, see the references described in *Table 15-6*.

- Standard level
  Output the trace information that can identify the boundaries (entrance and exit) of each function layer.
- Advanced level
  Output the trace information of processes in every function layer, in addition to the output contents of the standard level.
- Maintenance level
  This is the level for acquiring the maintenance information required when a failure occurs.
- Prevention level

This is the level for preventing the output of trace information. This level can be set up only in the RMI function layer.

When the operation is performed by using the Management Server, the trace information is output by setting up a common level for all function layers, in the Easy Setup definition file.

In the next sections, the trace collection points for the trace collection levels *Standard* and *Advanced* are described. Because the Maintenance level is the level for collecting maintenance information, such as when a failure occurs, the information for this level need not usually be collected.

15.6 Trace collection points of a redirector

This section describes the trace collection points of a redirector, and the trace information that can be collected.

Note that when the PRF trace collection level is set to *Advanced*, the trace of request processing and the session trace is output.

15.6.1 Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels. The session trace is not output at the point `0x8003`.

Table 15–8: Details of trace collection points in a redirector

Event ID	No. in the figure#	Trace acquisition points	Level
0x8000	1	Immediately after acquiring the request processing request	A/B
0x8001	2	Immediately after sending the request header information to a Web container	A/B
0x8002	4	Immediately after receiving a request for the HTTP request body information from the Web container	B
0x8003	6	Immediately after receiving the HTTP response header information from the Web container	B
0x8004	8	Immediately after starting the transmission of the HTTP response body information from the Web container	B
0x8100	11	Immediately after the completion of redirector processing	A/B
0x8101	10	Immediately after receiving the response completion notification from the Web container	A/B
0x8102	3, 5	Immediately before sending the HTTP request body information to the Web container	B
0x8103	7	Immediately after the setup completion of the HTTP response header information in the Web server	B
0x8104	9	Immediately after the completion of transmission of the HTTP response body information to the Web server	B

Legend:

B: Advanced

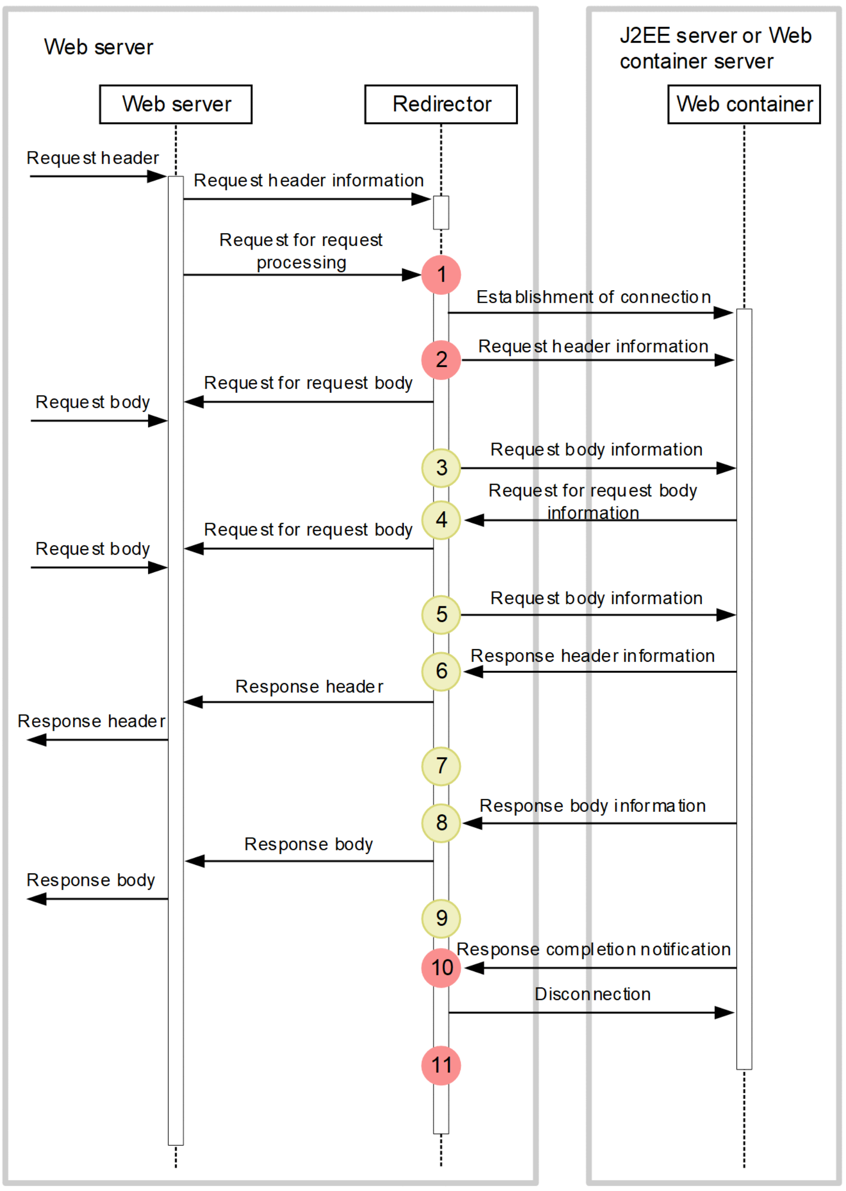
A/B: Different information is collected for the *Standard* and *Advanced* levels

#

Corresponds to the numbers in *Figure 15-6*.

The following figure shows the trace collection points in a redirector.

Figure 15–6: Trace collection points of a redirector


Web server
J2EE server or Web container server
Web server
Redirector
Web container
Request header
Request header information
Request for request processing
1
Establishment of connection
2
Request header information
Request for request body
Request body
3
Request body information
Request for request body information
4
Request for request body
Request body
5
Request body information
6
Response header information
Response header
Response header
7
8
Response body information
Response body
Response body
9
Response completion notification
10
Disconnection
11

Legend: 🔴 : Indicates a trace collection point. The PRF trace collection level is "Standard".

🟡 : Indicates a trace collection point. The PRF trace collection level is "Advanced".

Because a connection is usually used repeatedly to establish a permanent connection, it is not disconnected during a request. A connection is disconnected only when either an exception occurs during communication, or the upper limit value of the permanent connection is reached.

The output of the trace information is limited at the following points:

Point 3

The trace information is output when the body data is not in the chunk format. When the body data is in the chunk format, the request body request is not sent to the Web server between points 2 and 4, and the request body information is not sent to the Web container.

Points 4 and 5

The trace information is output only in the case of a request for the request body information from the Web container. Furthermore, because the request body information and response body information is sent to the client more than once at the points 4, 5, 8, and 9, the trace information is also output more than once.

Either an invalid session ID might be acquired, or the session ID might not be acquired at the following points:

Points 1 to 5

The session ID can be acquired. However, an invalid session ID (the ID of an HttpSession discarded in a J2EE application, or the ID of an HttpSession discarded due to the expiry of the valid period) might be acquired because the session ID is acquired from the Cookie or URL of the request header.

Even when a valid session ID is acquired, the session might be discarded in the J2EE application.

Points 7 to 11

The session ID can be acquired only when a session is generated.

15.6.2 Trace information that can be collected

The following table describes the trace information that can be collected in a redirector.

Table 15–9: Trace information that can be collected in a redirector

No. in the figure#1	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional#2
1	0x8000	A/B	Client address: HTTP method	URI	#3
2	0x8001	A/B	Container address: Port number	--	#3
3, 5	0x8102	B	Transmission size	--	*Entrance-time number-of-session-ID-characters*: *collection-method*
4	0x8002	B	Request size	--	*Number-of-session-ID-characters*: *session ID*: *collection-method*
6	0x8003	B	--	--	--
7	0x8103	B	--	--	#4
8	0x8004	B	Transmission size	--	*Number-of-session-ID-characters*: *session ID*: *collection-method*
9	0x8104	B	Transmitted size	--	*Entrance-time number-of-session-ID-characters*: *session-ID*
10	0x8101	A/B	Container address: Port number	--	#5
11	0x8100	A/B	Client address: HTTP method	URI	#6

Legend:

B: Advanced

A/B: Different information is collected for the *Standard* and *Advanced* levels

--: Not applicable

#1

Corresponds to the numbers in *Figure 15-6*.

#2

If the session Cookie name is changed to a string other than `JSESSIONID`, the number of session ID characters, session ID, and collection method are not output.

#3

When the level is *Standard*, nothing is output.

When the level is *Advanced*, the number of session ID characters: session ID: collection method is output.

#4

When the processing is performed normally, the entrance time and session ID are displayed.

When an exception occurs, in addition to the entrance time, four-byte maintenance information, and the number of session ID characters: session ID is displayed.

#5

When the level is *Standard*, the entrance time is displayed.

When the level is *Advanced*, in addition to the entrance time, the number of session ID characters: session ID is displayed.

#6

When the level is *Standard*, the entrance time and status code are displayed at all times.

However, if an exception occurs during the use of Microsoft IIS, in addition to the entrance time and status code, four-byte maintenance information will be displayed.

When the level is *Detailed*, in addition to the information that is output when the level is *Standard*, the number of session ID characters: session ID is displayed.

15.7 Trace collection points of a Web container (trace of request processing)

This section describes the trace collection points of a Web container, and the trace information that can be collected. In a Web container, the trace of request processing and the session trace are output. The trace of request processing is described below. The trace points when an in-process HTTP server is used, and the trace information that can be collected is also described.

15.7.1 Trace Get Point and the PRF Trace Get Level

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–10: Details of trace collection points in a Web container (trace of request processing)

Event ID	No. in the figure[#1]	Trace acquisition points		Level
0x8200[#2]	1	Immediately after a request is acquired, or when the request header analysis is complete	Via the Web server	A
0x8201			Via the simple Web server	A
0x8202[#3]	3	Immediately before a servlet or JSP is invoked		A
0x8203	2	Immediately before the filter that is executed before the execution of the servlet or JSP that receives the request is invoked (when the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is omitted, or when a filter for which `REQUEST` is specified in the `<dispatcher>` tag is invoked)		B
0x8206	4	Immediately before a servlet or JSP is invoked via RequestDispatcher		B
0x8207	3	When the static contents are invoked		B
0x8300	8	When request processing is complete	Via the Web server	A
0x8301			Via the simple Web server	A
0x8302	6	Immediately after the processing of the servlet or JSP is complete		A
0x8303	7	Immediately after the completion of processing of the filter that is executed before the execution of the servlet or JSP that receives the request (when the processing of the filter for which `REQUEST` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is complete)		B
0x8306	5	Immediately after the processing of the servlet or JSP via RequestDispatcher is complete		B
0x8307	6	Immediately after the completion of processing of static contents (DefaultServlet)		B

Legend:

A: Standard

B: Advanced

#1

Corresponds to the numbers in *Figure 15-7*.

#2

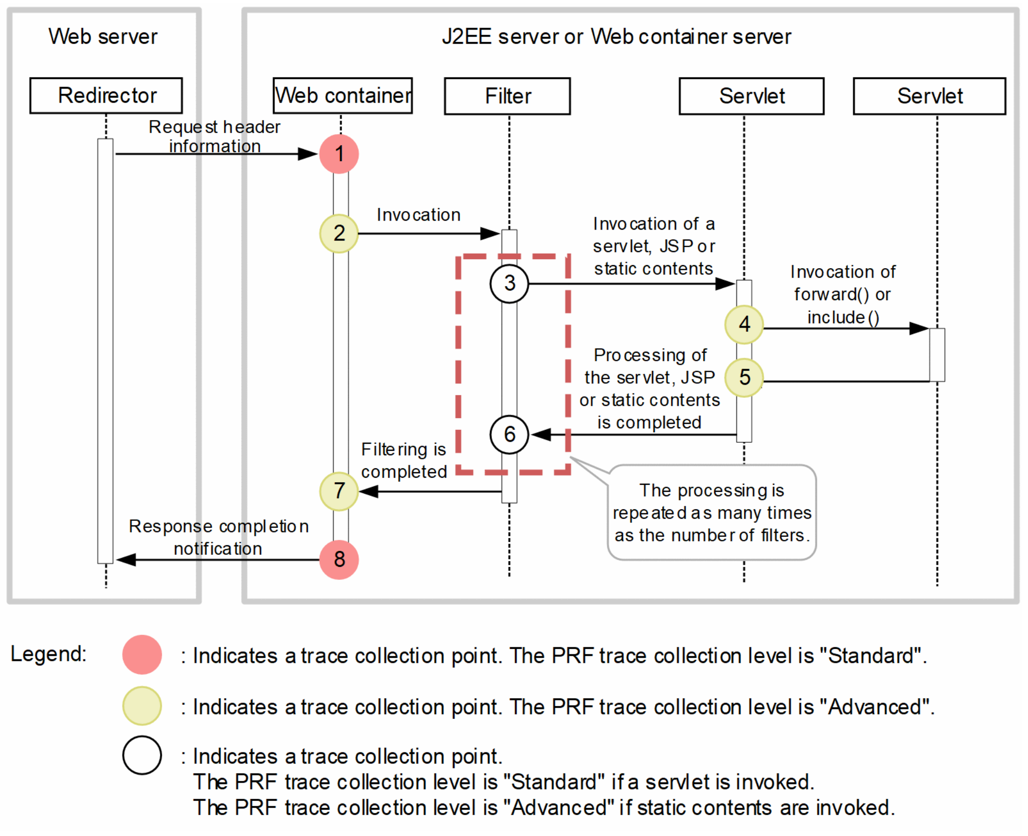
For a POST request, if the POST data does not arrive from the client, the request processing might be delayed in this interval.

#3

If a JSP compilation is required, the trace is collected after the JSP compilation is executed.

The following figure shows the trace collection points in a Web container.

Figure 15–7: Trace collection points of a Web container (trace of request processing)


Web server
J2EE server or Web container server
Redirector
Web container
Filter
Servlet
Servlet
Request header information
1
Invocation
2
Invocation of a servlet, JSP or static contents
3
Invocation of forward() or include()
4
Processing of the servlet, JSP or static contents is completed
5
6
Filtering is completed
7
The processing is repeated as many times as the number of filters.
Response completion notification
8
Legend:
: Indicates a trace collection point. The PRF trace collection level is "Standard".
: Indicates a trace collection point. The PRF trace collection level is "Advanced".
: Indicates a trace collection point.
The PRF trace collection level is "Standard" if a servlet is invoked.
The PRF trace collection level is "Advanced" if static contents are invoked.

15.7.2 Trace information that can be collected

The following table describes the trace information that can be collected in a Web container.

Table 15–11: Trace information that can be collected in a Web container (trace of request processing)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8200	A	HTTP method	URI	--
		B			*Number-of-session-ID-characters*: *session ID*: *collection-method*
	0x8201	A	HTTP method	URI	--
2	0x8203	B	Class name	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
3	0x8202	A	Class name or JSP file name	--	--

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
		B		Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8207	B	--	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
4	0x8206	B	Class name	Dispatch type Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
5	0x8306	B	Class name	Dispatch type Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
6	0x8302	A	Class name or JSP file name	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
				Context root name	• When normal: *Entrance-time number-of-session-ID-characters: session-ID* • For an exception: *Entrance-time exception-name: number-of-session-ID-characters: session-ID*
	0x8307	B	--	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
7	0x8303	B	Class name	--	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name: number-of-session-ID-characters: session-ID*
8	0x8300	A	HTTP method	URI	• When normal: *Entrance-time status-code* • For an exception: *Entrance-time status-code exception-name*

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
		B			• When normal: *Entrance-time status-code number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID* • For an exception: *Entrance-time status-code exception-name: number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8301	A	HTTP method	URI	• When normal: *Entrance-time status-code* • For an exception: *Entrance-time status-code number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in *Figure 15-7*.

Reference note

When a request is received from other than the SOAP client, `0` is displayed always in the *client application information* that is the key information of the trace information. The *client application information* is output only when a request is received from the SOAP client.

15.7.3 Trace collection points and PRF trace collection levels (when an in-process HTTP server is used)

The following table describes the events IDs, trace collection points, and PRF trace collection levels when an in-process HTTP server is used.

Table 15–12: Details of trace collection points in an in-process HTTP server

Event ID	No. in the figure[#1]	Trace acquisition points	Level
0x8202[#2]	5	Immediately before a servlet or JSP is invoked	A/B
0x8203	4	Immediately before the filter that is executed before the execution of the servlet or JSP that receives the request is invoked (when the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is omitted or when a filter for which `REQUEST` is specified in the `<dispatcher>` tag is invoked)	B
0x8206	6	Immediately before a servlet or JSP is invoked via RequestDispatcher	B
0x8207	5	Immediately before the static contents are invoked	B
0x8211	3	Immediately after acquiring a request/completing the request header analysis	A/B
0x8212	1	Immediately before data reading from the Web client starts	B
0x8213	8	Immediately before data writing into the Web client starts	B
0x8302	10	Immediately after the completion of processing of the servlet or JSP	A
0x8303	11	Immediately after the completion of processing of the filter that is executed before the execution of the servlet or JSP that receives the request (when the processing of the filter for which `REQUEST` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is complete)	B
0x8306	7	Immediately after the completion of processing of the servlet or JSP via RequestDispatcher	B
0x8307	10	Immediately after completion of processing of static contents (DefaultServlet)	B
0x8311	12	Immediately after the completion of request processing	A/B
0x8312	2	Immediately after the completion of data reading from the Web client	B
0x8313	9	Immediately after the completion of data writing into the Web client	B

Legend:

A: Standard

B: Advanced

A/B: Different information is collected for the *Standard* and *Advanced* levels.

#1

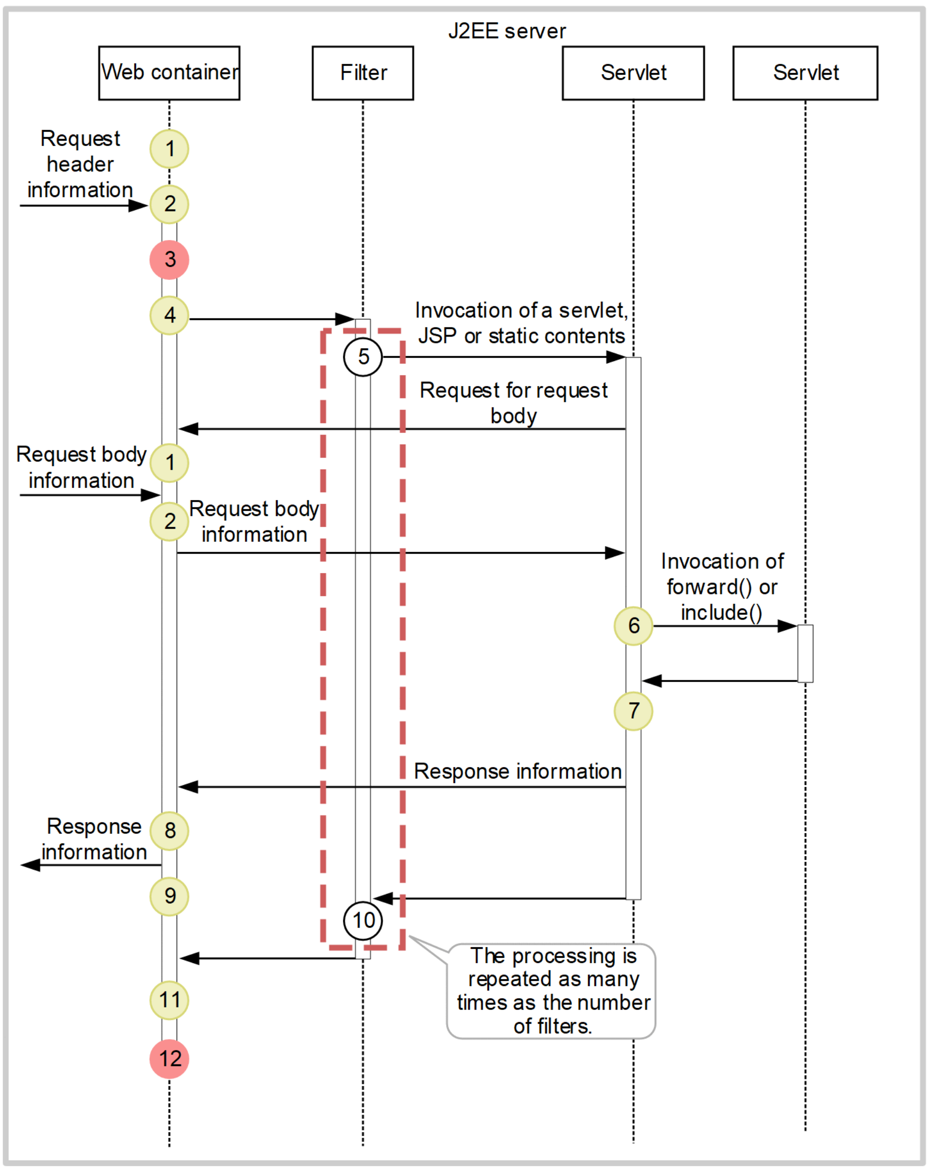
Corresponds to the numbers in *Figure 15-8*.

#2

If JSP compilation is required, the trace is collected after the JSP compilation is executed.

The following figure shows the trace collection points when an in-process HTTP server is used.

Figure 15–8: Trace collection points of an in-process HTTP server


J2EE server
Web container
Filter
Servlet
Servlet
Request header information
1
2
3
4
Invocation of a servlet, JSP or static contents
5
Request for request body
Request body information
1
2
Request body information
Invocation of forward() or include()
6
7
Response information
Response information
8
9
10
The processing is repeated as many times as the number of filters.
11
12

Legend:

● : Indicates a trace collection point. The PRF trace collection level is "Standard".

● : Indicates a trace collection point. The PRF trace collection level is "Advanced".

○ : Indicates a trace collection point.
The PRF trace collection level is "Standard" if a servlet is invoked.
The PRF trace collection level is "Advanced" if static contents are invoked.

Because the request information is received from the client more than once at the points 1 and 2, the trace information is also output more than once.

Because the response information is sent to the client more than once at the points 8 and 9, the trace information is also output more than once.

Even when a request for the request body information, and a response are sent from the filter, the trace for points 2 or 8, and points 8 and 9 is output.

15.7.4 Trace information that can be collected

The following table describes the trace information that can be collected in an in-process HTTP server.

Table 15–13: Trace information that can be collected in an in-process HTTP server (trace of request processing)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8212	B	Request size	--	--
2	0x8312	B	Read size	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
3	0x8211	A	HTTP method	URI	--
		B			*Number-of-session-ID-characters*: *session ID*: *collection-method*
4	0x8203	B	Class name	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
5	0x8202	A	Class name or JSP file name	--	--
		B		Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8207	B	-	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
6	0x8206	B	Class name	Dispatch type Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
7	0x8306	B	Class name	Dispatch type Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
8	0x8213	B	Write size	--	--
9	0x8313	B	Written size	--	• When normal: *Entrance-time* • For an exception:

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
					Entrance-time exception-name
10	0x8302	A	Class name or JSP file name	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
		B		Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
	0x8307	B	--	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
11	0x8303	B	Class name	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
12	0x8311	A	HTTP method	URI	*Entrance-time status-code*
		B			*Entrance-time status-code number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in *Figure 15-8*.

15.8 Trace collection points of a Web container (session trace)

This section describes the trace collection points of the trace of a Web container, and the trace information that can be collected. In a Web container, the trace of request processing and the session trace are output. The trace collection points of the session trace and global session, and the trace information that can be collected is described below.

15.8.1 Trace Get Point and the PRF Trace Get Level (Session Trace)

The following table describes the event IDs, trace collection points, and PRF trace collection levels of the trace concerning the session trace. Note that the information about the global session is also output at points `0x8203`, `0x8202`, `0x8207`, `0x8206`, and `0x8300`.

Table 15–14: Details of trace collection points in a Web container (session trace)

Event ID	No. in the figure#1	Trace acquisition points	Level#2
0x8200	1	Immediately after a request is acquired, or when the request header analysis is complete (via the Web server)	A/B
0x8202	4, 9	Immediately before a servlet or JSP is invoked	A/B
0x8203	2, 3	Immediately before the filter that is executed before the execution of the servlet or JSP that receives the request is invoked (when the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is omitted, or when a filter for which `REQUEST` is specified in the `<dispatcher>` tag is invoked)	B
0x8206	7	Immediately before a servlet or JSP is invoked via RequestDispatcher	B
0x8207	4, 9	Immediately before the static contents are invoked (DefaultServlet)	B
0x8208	5	After a session is generated	B
0x8209	6	After a session is discarded	B
0x8210	17	After the session times out	B
0x8211	1	Immediately after a request is acquired, or when the request header analysis is complete (via the in-process HTTP server)	A/B
0x8214	8	Immediately before the filter executed during Forward is invoked (when the filter for which `FORWARD` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is invoked)	B
0x8215	8	Immediately before the filter executed during Include is invoked (when the filter for which `INCLUDE` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is invoked)	B
0x8216	2	Immediately before the filter that is executed during transfer to the error page is invoked (when the filter for which `ERROR` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is invoked)	B
0x8300	16	Immediately after the completion of request processing (via the Web server)	A/B
0x8302	10, 13	Immediately after the completion of processing of the servlet or JSP	A/B
0x8303	14, 15	Immediately after the completion of processing of the filter that is executed before the execution of the servlet or JSP that receives the request	B

Event ID	No. in the figure#1	Trace acquisition points	Level#2
		(when the processing of the filter for which `REQUEST` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is complete)	
0x8306	12	Immediately after the completion of the processing of the servlet or JSP via RequestDispatcher	B
0x8307	10, 13	Immediately after the completion of the processing of the static contents (DefaultServlet)	B
0x8311	16	Immediately after the completion of the request processing (via the in-process HTTP server)	A/B
0x8314	11	Immediately after the completion of the processing of the filter executed during Forward (when the processing of the filter for which `FORWARD` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is complete)	B
0x8315	11	Immediately after the completion of the processing of the filter executed during Include (when the processing of the filter for which `INCLUDE` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is complete)	B
0x8316	15	Immediately after the completion of the processing of the filter executed during transfer to the error page (when the processing of the filter for which `ERROR` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is complete)	B

Legend:

A: Standard

B: Advanced

A/B: Different information is collected for the *Standard* and *Advanced* levels.

#1

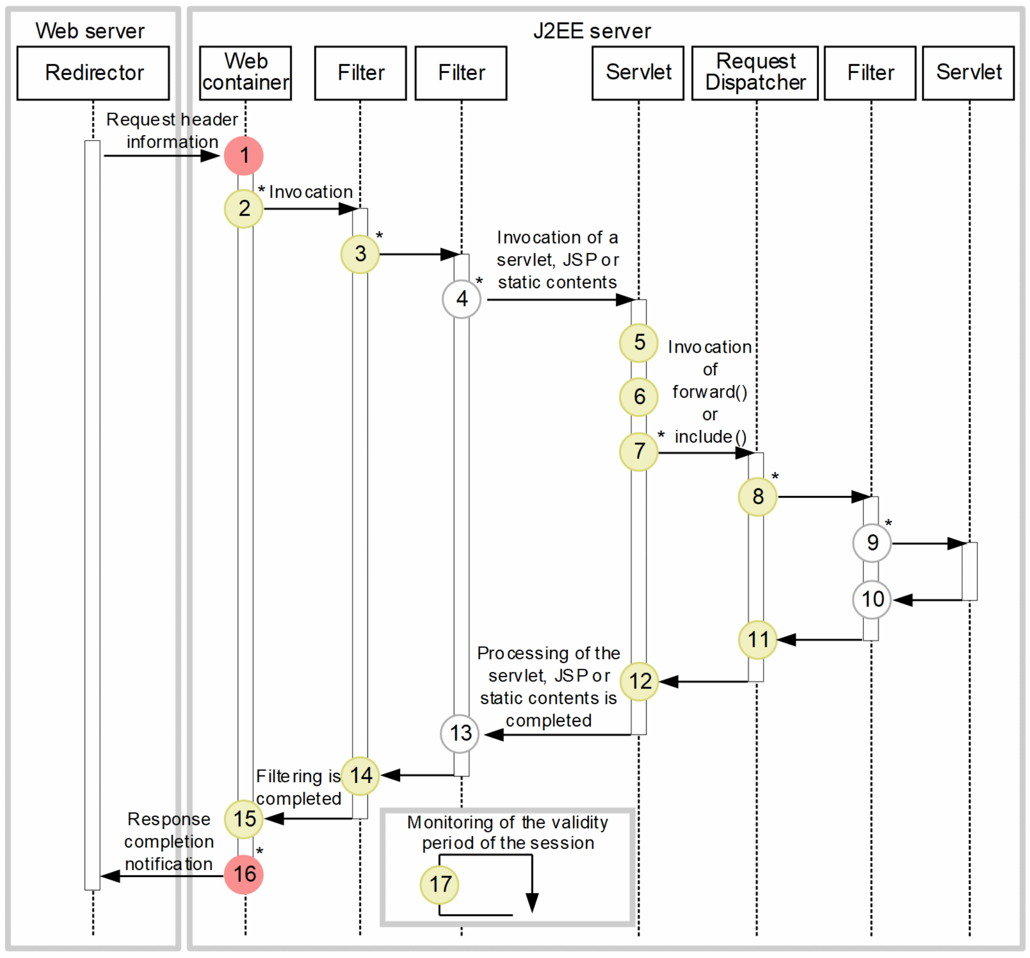
Corresponds to the numbers in *Figure 15-9*.

#2

The information about the session trace is output only when the level is *Advanced*.

The following figure shows the trace collection points of the session trace in a Web container.

Figure 15–9: Trace collection points of a Web container (session trace)


Web server
Redirector
J2EE server
Web container
Filter
Filter
Servlet
Request Dispatcher
Filter
Servlet
Request header information
1
2
* Invocation
3
*
4
*
Invocation of a servlet, JSP or static contents
5
6
7
Invocation of forward() or * include()
8
*
9
*
10
11
12
13
Processing of the servlet, JSP or static contents is completed
14
Filtering is completed
15
Response completion notification
16
*
Monitoring of the validity period of the session
17

Legend:

🔴 : Indicates a trace collection point. The PRF trace collection level is "Standard".

🟡 : Indicates a trace collection point. The PRF trace collection level is "Advanced".

○ : Indicates a trace collection point.
The PRF trace collection level differs depending on the event ID to be output.

* This indicates a trace collection point where a global session ID can also be collected.

The session ID that can be acquired at each point is as follows:

Point 1

The session ID can be acquired. However, an invalid session ID (the ID of an HttpSession discarded in a J2EE application, or the ID of an HttpSession discarded due to the expiry of the valid period) might be acquired because the session ID is acquired from the Cookie or URL of the request header.

Even when a valid session ID is acquired, the session might be discarded in the J2EE application.

Points 2, 3, 4, 7, 8, and 9

A valid session ID can be acquired at the trace collection points. However, the session might be discarded in the J2EE application.

Furthermore, the global session ID can also be acquired at these points. The contents of the global session ID that can be acquired are different for each trace collection point.

- Point 2 is the trace collection point at which the event ID `0x8203` is output initially for one request. At this trace collection point, the global session ID sent as a request from the Web client can be acquired. However, at this point, a global session ID that has already become invalid might also be output.

- Valid global session IDs can be acquired from the trace with event IDs `0x8216`, `0x8202`, `0x8203`, `0x8206`, `0x8207`, `0x8214`, and `0x8215` output at points 3, 4, 7, 8, and 9.

Point 5

A valid session ID can be acquired at the trace collection point only when a session is generated in the J2EE application. However, the session might be discarded in the J2EE application.

Point 6

An invalid session ID can be acquired at the trace collection point only when a session is discarded in the J2EE application. However, the session might be discarded in the J2EE application.

Points 10, 11, 12, and 13

A valid session ID can be acquired at the trace collection points. However, the session might be discarded in the J2EE application.

Points 14 and 15

A valid session ID can be acquired at the trace collection points. When the request processing finishes at these trace collection points, the session is not discarded in the J2EE application thereafter.

Point 16

A valid session ID can be acquired at the trace collection point. When the request processing finishes at this trace collection point, the session is not discarded in the J2EE application thereafter.

Furthermore, when a global session is generated, a valid global session ID can be acquired when the request processing ends.

Point 17

An invalid session ID can be acquired only when a session that has exceeded the valid period is discarded.

15.8.2 Trace information that can be collected

The following table describes the trace information about the session trace that can be collected in a Web container. The information about the global session is also output at the trace collection points with event IDs `0x8202`, `0x8203`, `0x8206`, `0x8207`, `0x8214`, `0x8215`, `0x8300`, and `0x8311`.

Table 15–15: Trace information that can be collected in a Web container (session trace)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8200	A	HTTP method	URI	--
		B			*Number-of-session-ID-characters*: *session-ID*: *collection-method*
4, 9	0x8202	A	Class name (JSP file name when a JSP is invoked)	--	--
		B		Context root name#2	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
2, 3	0x8203	B	Class name	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
7	0x8206	B	Class name	Dispatch type	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-*

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
				Context root name	*session-ID-characters*: *global-session-ID*
4, 9	0x8207	B	--	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
5	0x8208	B	Context root name	Session valid period	*Number-of-session-ID-characters*: *Session-ID*
6	0x8209	B	Context root name	Session generation time	*Number-of-session-ID-characters*: *Session-ID*
17	0x8210	B	Context root name	Session valid period: session generation time	*Number-of-session-ID-characters*: *Session-ID*
1	0x8211	A	HTTP method	URI	--
		B			*Number-of-session-ID-characters*: *session-ID*: *collection-method*
8	0x8214	B	Class name	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
8	0x8215	B	Class name	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
2	0x8216	B	Class name	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
16	0x8300	A	HTTP method	URI	• When normal: *Entrance-time status-code* • For an exception: *Entrance-time status-code exception-name*
		B			• When normal: *Entrance-time status-code number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID* • For an exception: *Entrance-time status-code exception-name: number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
10, 13	0x8302	A	Class name (JSP file name when a JSP is invoked)	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
		B		Context root name	• When normal:

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
					Entrance-time number-of-session-ID-characters: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
14, 15	0x8303	B	Class name	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
12	0x8306	B	Class name	Dispatch type Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
10, 13	0x8307	B	--	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
16	0x8311	A	HTTP method	URI	*Entrance-time status-code*
		B			*Entrance-time status-code number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
11	0x8314	B	Class name	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
11	0x8315	B	Class name	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
15	0x8316	B	Class name	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in *Figure 15-9*.

15.9 Trace collection points of a Web container (filter trace)

This section describes the trace collection points of the trace of a Web container when a filter that is invoked during Forward or Include is specified, and also describes the trace information that can be collected.

In the case of a Web container in which a filter that is invoked during Forward or Include is specified, the trace information that can be collected is different when the processing terminates normally, and when an error occurs. Trace acquisition points for both cases are explained below.

When an error page is set up by using the `errorPage` attribute in the page directive of a JSP, and an exception occurs in the JSP, the error page will be displayed when forwarding the request. Therefore, the trace output during Forward will be output even when an error page is displayed in the JSP.

15.9.1 Trace collection points of a Web container when the processing terminates normally (filter trace)

(1) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–16: Details of trace collection points in a Web container during normal termination (filter trace)

Event ID	No. in the figure#	Trace acquisition points	Level
0x8200	1	Immediately after a request is acquired, or when the request header analysis is complete (via the Web server)	A/B
0x8201	1	Immediately after a request is acquired, or when the request header analysis is complete (via the simple Web server)	A
0x8202	3	Immediately before a servlet or JSP is invoked	A/B
0x8202	6	Immediately before a servlet or JSP is invoked	A/B
0x8203	2	Immediately before the filter that is executed before the execution of the servlet or JSP that receives the request is invoked (filter for which the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is omitted, or for which `REQUEST` is specified in the `<dispatcher>` tag)	B
0x8206	4	Immediately before a servlet or JSP is invoked via RequestDispatcher	B
0x8207	6	Immediately before the static contents are invoked (DefaultServlet)	B
0x8211	1	Immediately after a request is acquired, or when the request header analysis is complete (via the in-process HTTP server)	A/B
0x8214	5	Immediately before the filter executed during Forward is invoked (when the filter for which `FORWARD` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is invoked)	B
0x8215	5	Immediately before the filter executed during Include is invoked (filter for which `INCLUDE` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml`)	B
0x8300	12	Immediately after completion of the request processing (via the Web server)	A/B

Event ID	No. in the figure#	Trace acquisition points	Level
0x8301	12	Immediately after completion of the request processing (via the simple Web server)	A
0x8302	7	Immediately after completion of the processing of the servlet or JSP is complete	A/B
0x8302	10	Immediately after completion of the processing of the servlet or JSP is complete	A/B
0x8303	11	Immediately before the processing of the filter that is executed before the execution of the servlet or JSP that receives the request is complete (filter for which `REQUEST` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml`)	B
0x8306	9	Immediately after the processing of the servlet or JSP via RequestDispatcher is complete	B
0x8307	7	Immediately after completion of the processing of the static contents (DefaultServlet)	B
0x8311	12	Immediately after completion of the request processing is complete (via the in-process HTTP server)	A/B
0x8314	8	Immediately after completion of the processing of the filter executed during Forward (filter for which `FORWARD` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml`)	B
0x8315	8	Immediately after completion of the processing of the filter executed during Include (filter for which `INCLUDE` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml`)	B

Legend:

A: Standard

B: Advanced

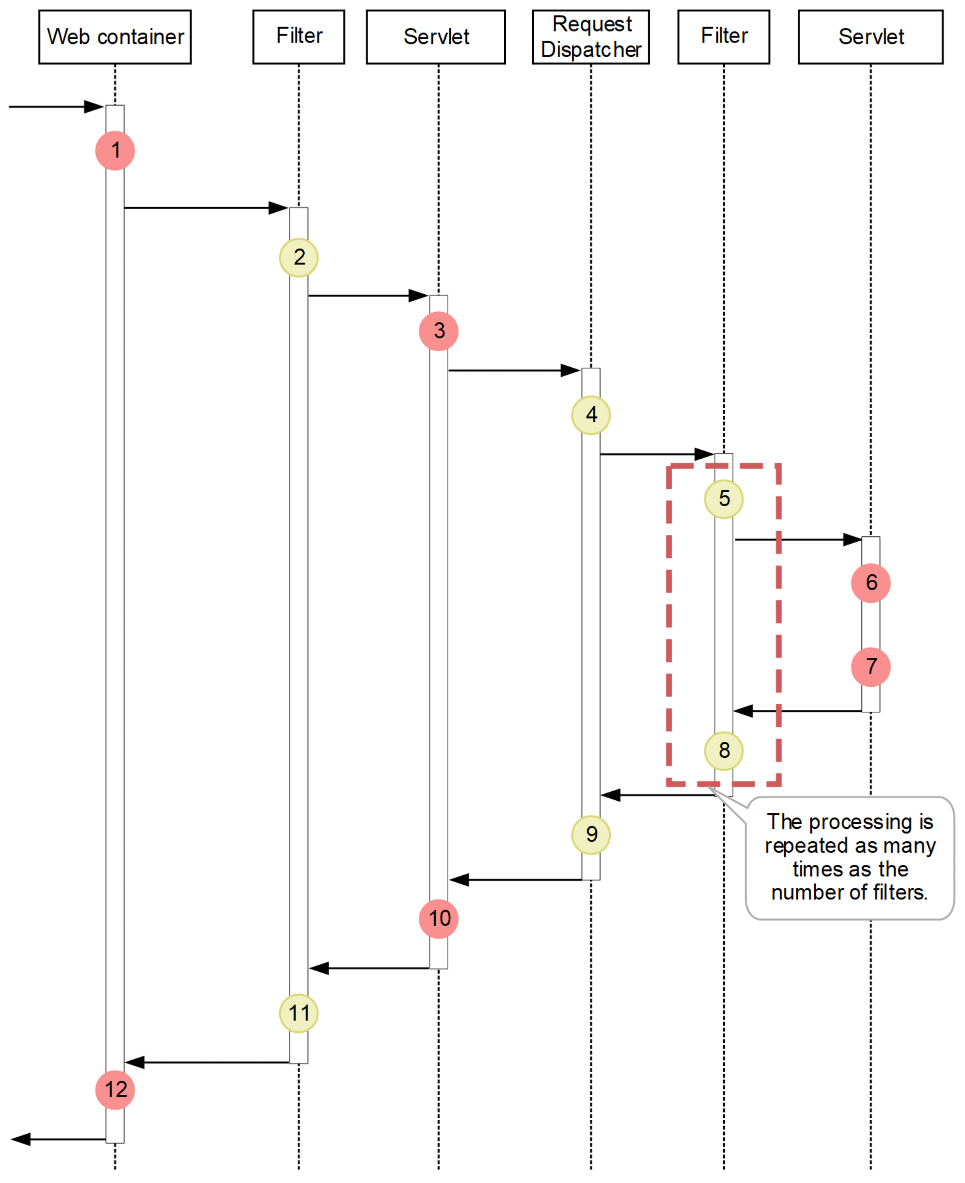
A/B: Different information is collected for the *Standard* and *Advanced* levels.

#

Corresponds to the numbers in *Figure 15-10*.

The following figure shows the trace collection points in a Web container, when the filter that is invoked during Forward or Include is specified.

Figure 15–10: Trace collection points in a Web container during normal termination (filter trace)



Legend:

🔴 : Indicates a trace collection point. The PRF trace collection level is "Standard".

🟡 : Indicates a trace collection point. The PRF trace collection level is "Advanced".

(2) Trace information that can be collected

The following table describes the trace information that can be collected in a Web container, when the filter that is invoked during Forward or Include is specified.

Table 15–17: Trace information that can be collected in a Web container during normal termination (filter trace)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8200	A	HTTP method	URI	--

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
		B			*Number-of-session-ID-characters*: *session ID*: *collection-method*
	0x8201	A	HTTP method	URI	--
	0x8211	A	HTTP method	URI	--
		B			*Number-of-session-ID-characters*: *session ID*: *collection-method*
2	0x8203	B	Class name	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
3	0x8202	A	Class name (JSP file name when a JSP is invoked)	--	--
		B		Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
4	0x8206	B	Class name	Dispatch type Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
5	0x8214	B	Class name	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8215	B	Class name	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
6	0x8202	A	Class name (JSP file name when a JSP is invoked)	--	--
		B		Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8207	B	--	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
7	0x8302	A	Class name (JSP file name when a JSP is invoked)	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
		B		Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
	0x8307	B	--	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
8	0x8314	B	Class name	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
	0x8315	B	Class name	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
9	0x8306	B	Class name	Dispatch type Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
10	0x8302	A	Class name (JSP file name when a JSP is invoked)	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
		B		Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception:

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
					Entrance-time exception-name: *number-of-session-ID-characters*: *session-ID*
11	0x8303	B	Class name	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
12	0x8300	A	HTTP method	URI	• When normal: *Entrance-time status-code* • For an exception: *Entrance-time status-code exception-name*
		B			• When normal: *Entrance-time status-code number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID* • For an exception: *Entrance-time status-code exception-name: number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8301	A	HTTP method	URI	• When normal: *Entrance-time status-code* • For an exception: *Entrance-time status-code exception-name*
	0x8311	A	HTTP method	URI	*Entrance-time status-code*
		B			*Entrance-time status-code number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in *Figure 15-10*.

15.9.2 Trace collection points of a Web container when an exception occurs (filter trace)

(1) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–18: Details of trace collection points in a Web container when an exception occurs (filter trace)

Event ID	No. in the figure#	Trace collection points	Level
0x8200	1	Immediately after a request is acquired or after the completion of request header analysis (via the Web server)	A/B
0x8201	1	When a request is acquired or immediately after the completion of request header analysis (via the simple Web server)	A
0x8202	3	Immediately before a servlet or JSP is invoked	A/B
0x8202	8	Immediately before a servlet or JSP is invoked	A/B
0x8203	2	Immediately before the invocation of the filter that is executed before the execution of the servlet or JSP which received a request (filter for which the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is omitted, or for which `REQUEST` is specified in the `<dispatcher>` tag)	B
0x8206	6	Immediately before a servlet or JSP is invoked via RequestDispatcher	B
0x8207	8	Immediately before the static contents are invoked (DefaultServlet)	B
0x8211	1	When a request is acquired or immediately after the completion of request header analysis (via the in-process HTTP server)	A/B
0x8216	7	Immediately before the invocation of the filter that is executed when being transferred to the error page (the filter for which `ERROR` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml`)	B
0x8300	12	Immediately after the completion of request processing (via the Web server)	A/B
0x8301	12	Immediately after the completion of request processing (via the simple Web server)	A
0x8302	4, 9	Immediately after the completion of processing of the servlet or JSP	A/B
0x8303	5	Immediately after the completion of the processing of the filter that is executed before the execution of the servlet or JSP which received a request (the filter for which `REQUEST` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml`)	B
0x8306	11	Immediately after the completion of the processing of the servlet or JSP via RequestDispatcher	B
0x8307	9	Immediately after the completion of the processing of static contents (DefaultServlet)	B
0x8311	12	Immediately after the completion of request processing (via the in-process HTTP server)	A/B
0x8316	10	Immediately after the completion of the processing of the filter executed when being transferred to the error page (the filter for which `ERROR` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml`)	B

Legend:
A: Standard

B: Advanced

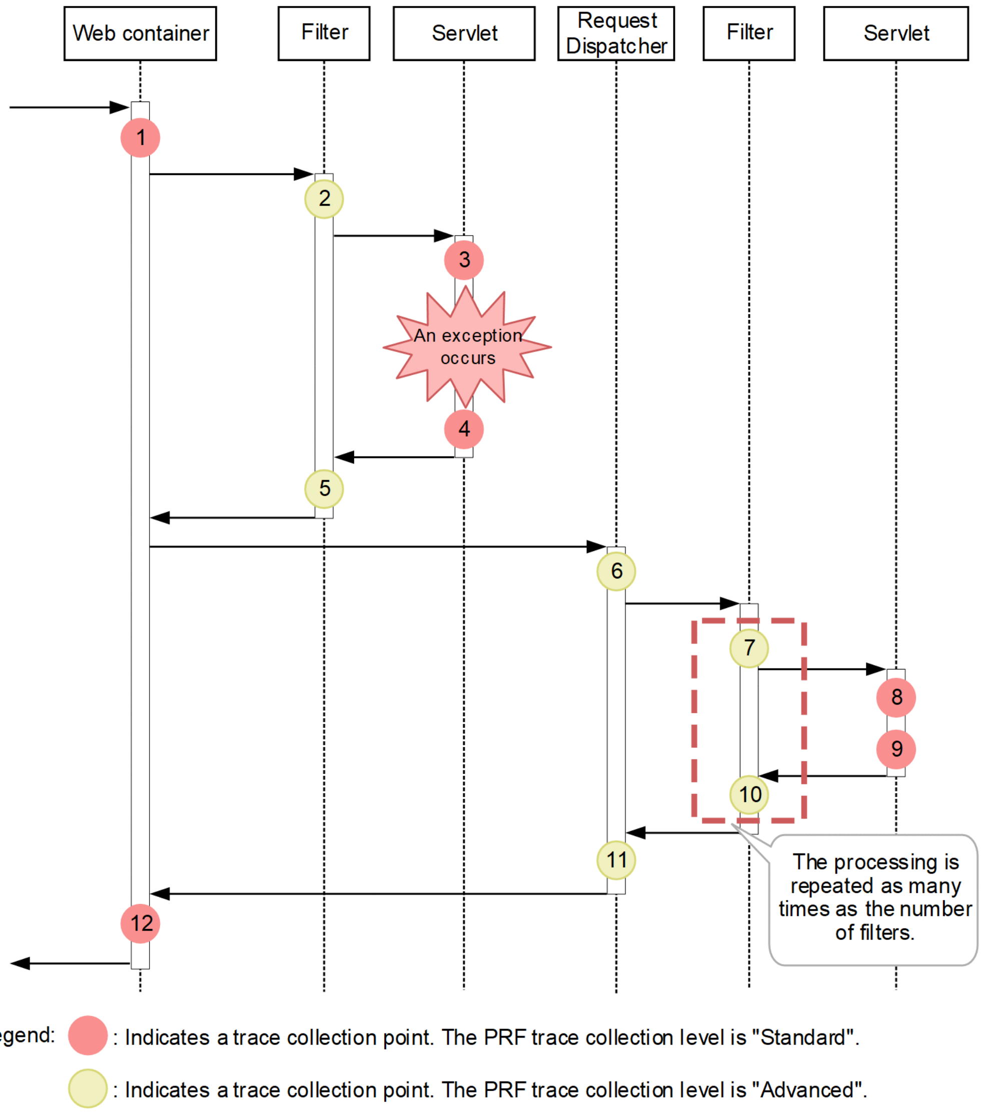
A/B: Different information is collected for the *Standard* and *Advanced* levels.

#

Corresponds to the numbers in *Figure 15-11*.

The following figure shows the trace collection points in a Web container when an exception occurs.

Figure 15–11: Trace collection points in a Web container when an exception occurs



Web container
Filter
Servlet
Request Dispatcher
Filter
Servlet
1
2
3
An exception occurs
4
5
6
7
8
9
10
11
12
The processing is repeated as many times as the number of filters.
Legend:
: Indicates a trace collection point. The PRF trace collection level is "Standard".
: Indicates a trace collection point. The PRF trace collection level is "Advanced".

(2) Trace information that can be collected

The following table describes the trace information that can be collected in a Web container, when the filter that is invoked during Forward or Include is specified.

Table 15–19: Trace information that can be collected in a Web container when an exception occurs (filter trace)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8200	A	HTTP method	URI	--

No. in the figure[#]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
		B			*Number-of-session-ID-characters*: *session ID*: *collection-method*
	0x8201	A	HTTP method	URI	--
	0x8211	A	HTTP method	URI	--
		B			*Number-of-session-ID-characters*: *session ID*: *collection-method*
2	0x8203	B	Class name	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
3	0x8202	A	Class name (JSP file name when a JSP is invoked)	--	--
		B		Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
4	0x8302	A	Class name (JSP file name when a JSP is invoked)	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
		B		Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
5	0x8303	B	Class name	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
6	0x8206	B	Class name	Dispatch type Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
7	0x8216	B	Class name	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*

No. in the figure[#]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
8	0x8202	A	Class name (JSP file name when a JSP is invoked)	--	--
		B		Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8207	B	--	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
9	0x8302	A	Class name (JSP file name when a JSP is invoked)	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
		B		Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
	0x8307	B	--	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
10	0x8316	B	Class name	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
11	0x8306	B	Class name	Dispatch type Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
12	0x8300	A	HTTP method	URI	• When normal: *Entrance-time status-code* • For an exception:

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
					Entrance-time status-code exception-name
		B			• When normal: *Entrance-time status-code number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID* • For an exception: *Entrance-time status-code exception-name: number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8301	A	HTTP method	URI	• When normal: *Entrance-time status-code* • For an exception: *Entrance-time status-code number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8311	A	HTTP method	URI	*Entrance-time status-code*
		B			*Entrance-time status-code number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in *Figure 15-11*.

15.10 Trace collection points of a Web container (trace of the database session failover functionality)

This section describes the trace collection points and the trace information that can be collected when the database session failover functionality is used.

15.10.1 Trace collection points and trace information that can be collected during request processing for creating an HTTP session (Trace of the database session failover functionality)

This section describes the trace collection points and the trace information that can be collected during request processing for creating an HTTP session.

(1) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–20: Details of trace collection points of the request processing for creating an HTTP session (database session failover functionality)

Event ID	No. in the figure#	Trace acquisition points	Level
0x8200	1	Immediately after the acquisition of a request, or after the completion of the request header analysis (via the Web server)	A/B
0x8202	2	Immediately before invoking a servlet or JSP	A/B
0x8203	2	Immediately before invoking the filter that is executed before the execution of the servlet or JSP that receives the request (filter for which the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is omitted, or for which `REQUEST` is specified in the `<dispatcher>` tag)	B
0x8207	2	Immediately before invoking the static contents (DefaultServlet)	B
0x8211	1	Immediately after the acquisition of a request, or after the completion of the request header analysis (via the in-process HTTP server)	A/B
0x8219	6	Immediately before starting the serialization of the HTTP session property information with the database session failover functionality	A
0x8222	8	Immediately before starting database access after the Web application processing with the database session failover functionality	A
0x8223	3	Immediately before starting database access during the creation of the HTTP session with the database session failover functionality	A
0x8300	10	Immediately after the completion of request processing (via the Web server)	A/B
0x8302	5	Immediately after the completion of processing of the servlet or JSP	A/B
0x8303	5	Immediately after the completion of processing of the filter that is executed before the execution of the servlet or JSP that receives the request (filter for which `REQUEST` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml`)	B
0x8307	5	Immediately after the completion of processing of the static contents (DefaultServlet)	B

Event ID	No. in the figure#	Trace acquisition points	Level
0x8311	10	Immediately after the completion of request processing (via the in-process HTTP server)	A/B
0x8316	5	Immediately after the completion of processing of the filter executed during transfer to the error page is complete (filter for which `ERROR` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml`)	B
0x8319	7	Immediately after the termination of serialization of the HTTP session property information with the database session failover functionality	A
0x8322	9	Immediately after the termination of database access after the Web application processing with the database session failover functionality	A
0x8323	4	Immediately after the termination of database access during the creation of the HTTP session with the database session failover functionality	A

Legend:

A: Standard

B: Advanced

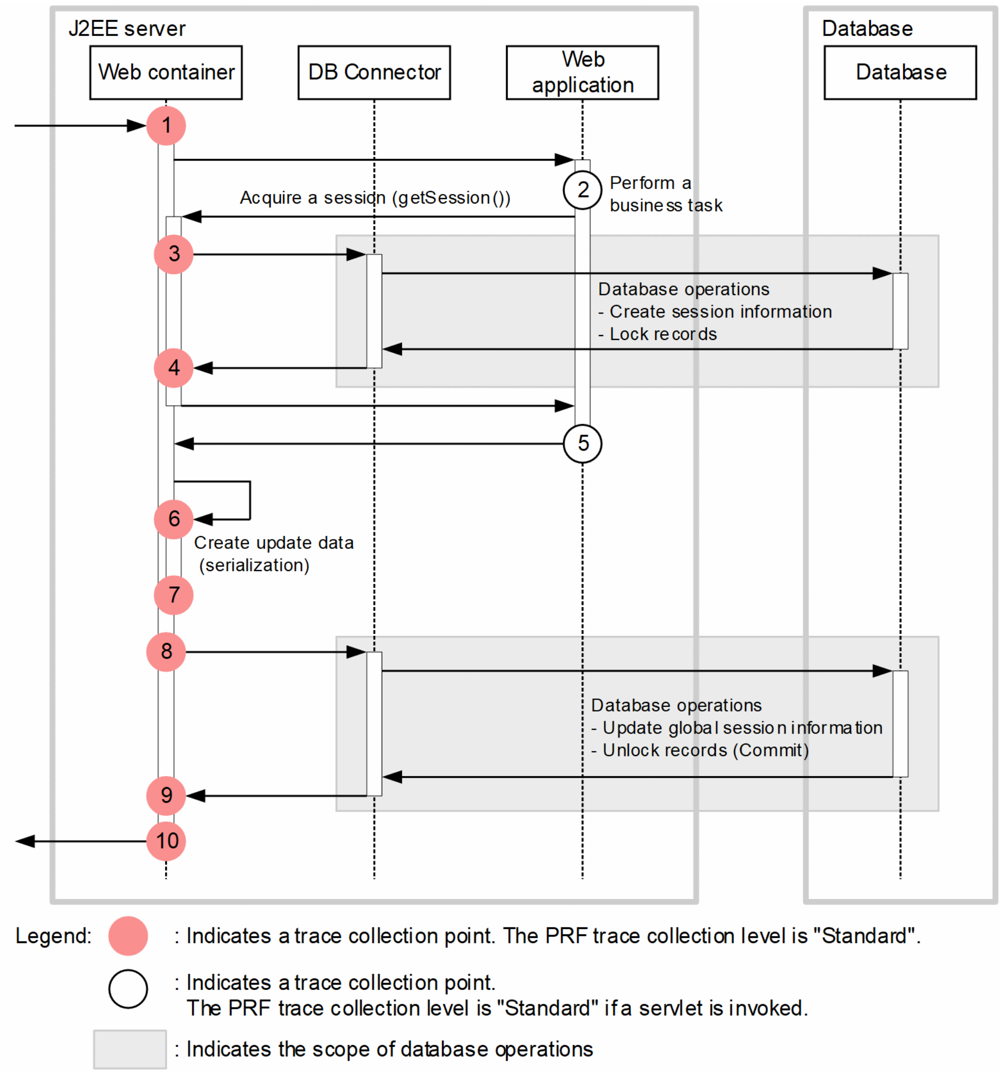
A/B: Different information is collected for the *Standard* and *Advanced* levels.

#

Corresponds to the numbers in *Figure 15-12*.

The following figure shows the trace collection points.

Figure 15–12: Trace collection points of the request processing for creating an HTTP session (database session failover functionality)


J2EE server
Web container
DB Connector
Web application
Database
Database
1
Perform a business task
2
Acquire a session (getSession())
3
Database operations
- Create session information
- Lock records
4
5
6
Create update data (serialization)
7
8
Database operations
- Update global session information
- Unlock records (Commit)
9
10
Legend:
: Indicates a trace collection point. The PRF trace collection level is "Standard".
: Indicates a trace collection point.
The PRF trace collection level is "Standard" if a servlet is invoked.
: Indicates the scope of database operations

(2) Trace information that can be collected

The following table describes the trace information that can be collected during request processing for creating an HTTP session.

Table 15–21: Trace information that can be collected during request processing for creating an HTTP session (database session failover functionality)

No. in the figure[#]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8200	A	HTTP method	URI	--
		B			*Number-of-session-ID-characters*: *session-ID*: *collection-method*
	0x8211	A	HTTP method	URI	--
		B			*Number-of-session-ID-characters*: *session-ID*: *collection-method*

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
2	0x8202	A	Class name (JSP file name when a JSP is invoked)	--	--
		B		Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8207	B	--	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8203	B	Class name	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
3	0x8223	A	--	--	--
4	0x8323	A	When integrity protection mode is set Number of the record for which exclusion is acquired When integrity protection mode is -1	--	• When normal: *Entrance-time number-of-session-ID-characters: session-ID: session-ID-of-HTTP-session-created* • For an exception: *Entrance-time exception-name*
5	0x8302	A	Class name (JSP file name when a JSP is invoked)	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
		B		Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
	0x8307	B	--	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
	0x8303	B	Class name	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID*

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
					• For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
	0x8316	B	Class name	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
6	0x8219	A	Request URL	--	*Number-of-session-ID-characters*: *session-ID*
7	0x8319	A	Request URL	Size (bytes) of the HTTP session property information after serialization	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
8	0x8222	A	--	--	*Number-of-session-ID-characters*: *session-ID-of-HTTP-session*
9	0x8322	A	When integrity protection mode is set Number of the record for which exclusion is released When integrity protection mode is disabled -1	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
10	0x8300	A	HTTP method	URI	• When normal: *Entrance-time status-code* • For an exception: *Entrance-time status-code exception-name*
		B			• When normal: *Entrance-time status-code number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID* • For an exception: *Entrance-time status-code exception-name: number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8311	A	HTTP method	URI	*Entrance-time status-code*

No. in the figure[#]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
		B			*Entrance-time status-code number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in *Figure 15-12*.

15.10.2 Trace collection points and trace information that can be collected during request processing for updating an HTTP session (Trace of database session failover functionality)

This section describes the trace collection points and the trace information that can be collected during request processing for updating an HTTP session.

(1) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–22: Details of trace collection points of the request processing for updating an HTTP session (database session failover functionality)

Event ID	No. in the figure[#1]	Trace acquisition points	Level
0x8200	1	Immediately after the acquisition of a request, or after the completion of the request header analysis (via the Web server)	A/B
0x8202	6	Immediately before invoking a servlet or JSP	A/B
0x8203	6	Immediately before invoking the filter that is executed before the execution of the servlet or JSP that receives the request (filter for which the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is omitted, or for which `REQUEST` is specified in the `<dispatcher>` tag)	B
0x8207	6	Immediately before invoking the static contents (DefaultServlet)	B
0x8211	1	Immediately after the acquisition of a request, or after the completion of the request header analysis (via the in-process HTTP server)	A/B
0x8219[#2]	8	Immediately before starting the serialization of the HTTP session property information with the database session failover functionality	A
0x8220	4	Immediately before starting the de-serialization of the HTTP session property information with the database session failover functionality	A
0x8221	2	Immediately before starting database access before the Web application processing with the database session failover functionality	A

Event ID	No. in the figure[#1]	Trace acquisition points	Level
0x8222[#2]	10	Immediately before starting database access after the Web application processing with the database session failover functionality	A
0x8300	12	Immediately after the completion of request processing (via the Web server)	A/B
0x8302	7	Immediately after the completion of processing of the servlet or JSP	A/B
0x8303	7	Immediately after the completion of processing of the filter that is executed before the execution of the servlet or JSP that receives the request (filter for which `REQUEST` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml`)	B
0x8307	7	Immediately after the completion of processing of the static contents (DefaultServlet)	B
0x8311	12	Immediately after the completion of request processing (via the in-process HTTP server)	A/B
0x8316	7	Immediately after the completion of processing of the filter executed during transfer to the error page is complete (filter for which `ERROR` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml`)	B
0x8319[#2]	9	Immediately after the termination of serialization of the HTTP session property information with the database session failover functionality	A
0x8320	5	Immediately after the termination of de-serialization of the HTTP session property information with the database session failover functionality	A
0x8321	3	Immediately after the termination of database access before the Web application processing with the database session failover functionality	A
0x8322[#2]	11	Immediately after the termination of database access after the Web application processing with the database session failover functionality	A

Legend:

A: Standard

B: Advanced

A/B: Different information is collected for the *Standard* and *Advanced* levels.

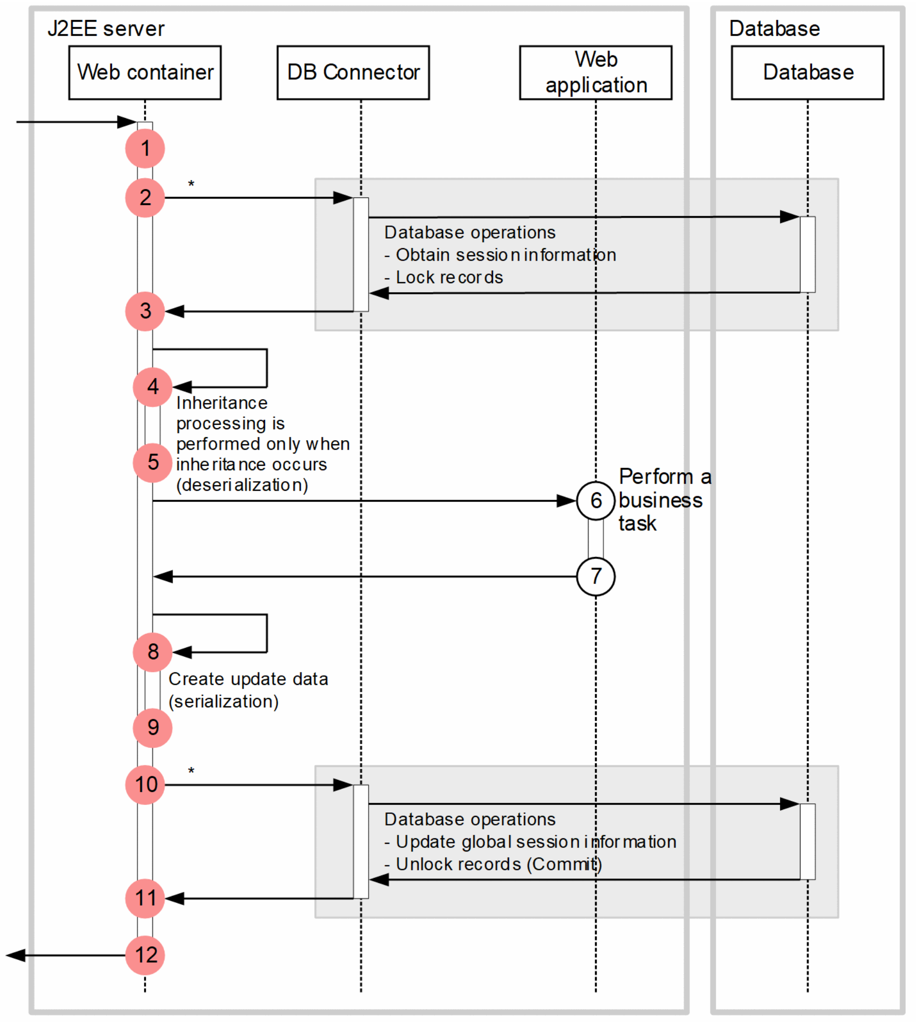
#1

Corresponds to the numbers in *Figure 15-13*.

#2

Not output for requests meant for referencing the HTTP session.

Figure 15–13: Trace collection points of the request processing for updating an HTTP session (database session failover functionality)


J2EE server
Web container
DB Connector
Web application
Database
Database
1
2
*
Database operations
- Obtain session information
- Lock records
3
4
Inheritance processing is performed only when inheritance occurs (deserialization)
5
6
Perform a business task
7
8
Create update data (serialization)
9
10
*
Database operations
- Update global session information
- Unlock records (Commit)
11
12

Legend: ● : Indicates a trace collection point. The PRF trace collection level is "Standard".

○ : Indicates a trace collection point.
The PRF trace collection level is "Standard" if a servlet is invoked.

▨ : Indicates the scope of database operations

* Actually, DB Connector will be invoked more than once.

(2) Trace information that can be collected

The following table describes the trace information that can be collected during request processing for updating an HTTP session.

Table 15–23: Trace information that can be collected during request processing for creating an HTTP session (database session failover functionality)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8200	A	HTTP method	URI	--

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
		B			*Number-of-session-ID-characters*: *session ID*: *collection-method*
	0x8211	A	HTTP method	URI	--
		B			*Number-of-session-ID-characters*: *session ID*: *collection-method*
2	0x8221	A	--	--	*Number-of-session-ID-characters*: *session-ID-received-with-the-request*
3	0x8321	A	When integrity protection mode is set Number of the record for which exclusion is acquired When integrity protection mode is disabled -1	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
4	0x8220	A	Request URL	Size (bytes) of the HTTP session property information before de-serialization	*Number-of-session-ID-characters*: *session-ID*
5	0x8320	A	Request URL	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
6	0x8202	A	Class name (JSP file name when a JSP is invoked)	--	--
		B		Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8207	B	--	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8203	B	Class name	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
7	0x8302	A	Class name (JSP file name when a JSP is invoked)	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
		B		Context root name	• When normal:

No. in the figure[#]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
					Entrance-time number-of-session-ID-characters: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
	0x8307	B	--	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
	0x8303	B	Class name	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
	0x8316	B	Class name	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
8	0x8219	A	Request URL	--	*Number-of-session-ID-characters*: *session-ID*
9	0x8319	A	Request URL	Size (bytes) of the HTTP session property information after serialization	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
10	0x8222	A	--	--	*Number-of-session-ID-characters*: *session-ID-of-HTTP-session*
11	0x8322	A	When integrity protection mode is set Number of the record for which exclusion is released When integrity protection mode is disabled -1	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
12	0x8300	A	HTTP method	URI	• When normal:

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
					Entrance-time status-code • For an exception: *Entrance-time status-code exception-name*
		B			• When normal: *Entrance-time status-code number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID* • For an exception: *Entrance-time status-code exception-name: number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8311	A	HTTP method	URI	*Entrance-time-status-code*
		B			*Entrance-time status-code number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in *Figure 15-13*.

15.10.3 Trace collection points and trace information that can be collected during request processing for disabling an HTTP session (Trace of database session failover functionality)

This section describes the trace collection points and the trace information that can be collected during request processing for disabling an HTTP session.

(1) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–24: Details of trace collection points of the request processing for disabling an HTTP session (database session failover functionality)

Event ID	No. in the figure#	Trace acquisition points	Level
0x8200	1	Immediately after the acquisition of a request, or after the completion of the request header analysis (via the Web server)	A/B
0x8202	6	Immediately before invoking a servlet or JSP	A/B
0x8203	6	Immediately before invoking the filter that is executed before the execution of the servlet or JSP that receives the request (filter for which the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml` is omitted, or for which `REQUEST` is specified in the `<dispatcher>` tag)	B
0x8207	6	Immediately before invoking the static contents (DefaultServlet)	B
0x8211	1	Immediately after the acquisition of a request, or after the completion of the request header analysis (via the in-process HTTP server)	A/B
0x8220	4	Immediately before starting the de-serialization of the HTTP session property information with the database session failover functionality	A
0x8221	2	Immediately before starting database access before the Web application processing with the database session failover functionality	A
0x8224	7	Immediately before starting database access when disabling the HTTP session with the database session failover functionality	A
0x8300	10	Immediately after the completion of request processing (via the Web server)	A/B
0x8302	9	Immediately after the completion of processing of the servlet or JSP	A/B
0x8303	9	Immediately after the completion of processing of the filter that is executed before the execution of the servlet or JSP that receives the request (filter for which `REQUEST` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml`)	B
0x8307	9	Immediately after the completion of processing of the static contents (DefaultServlet)	B
0x8311	10	Immediately after the completion of request processing (via the in-process HTTP server)	A/B
0x8316	9	Immediately after the completion of processing of the filter executed during transfer to the error page is complete (filter for which `ERROR` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml`)	B
0x8320	5	Immediately after the termination of de-serialization of the HTTP session property information with the database session failover functionality	A
0x8321	3	Immediately after the termination of database access before the Web application processing with the database session failover functionality	A
0x8324	8	Immediately after the termination of database access when the HTTP session is disabled with the database session failover functionality	A

Legend:

A: Standard

B: Advanced

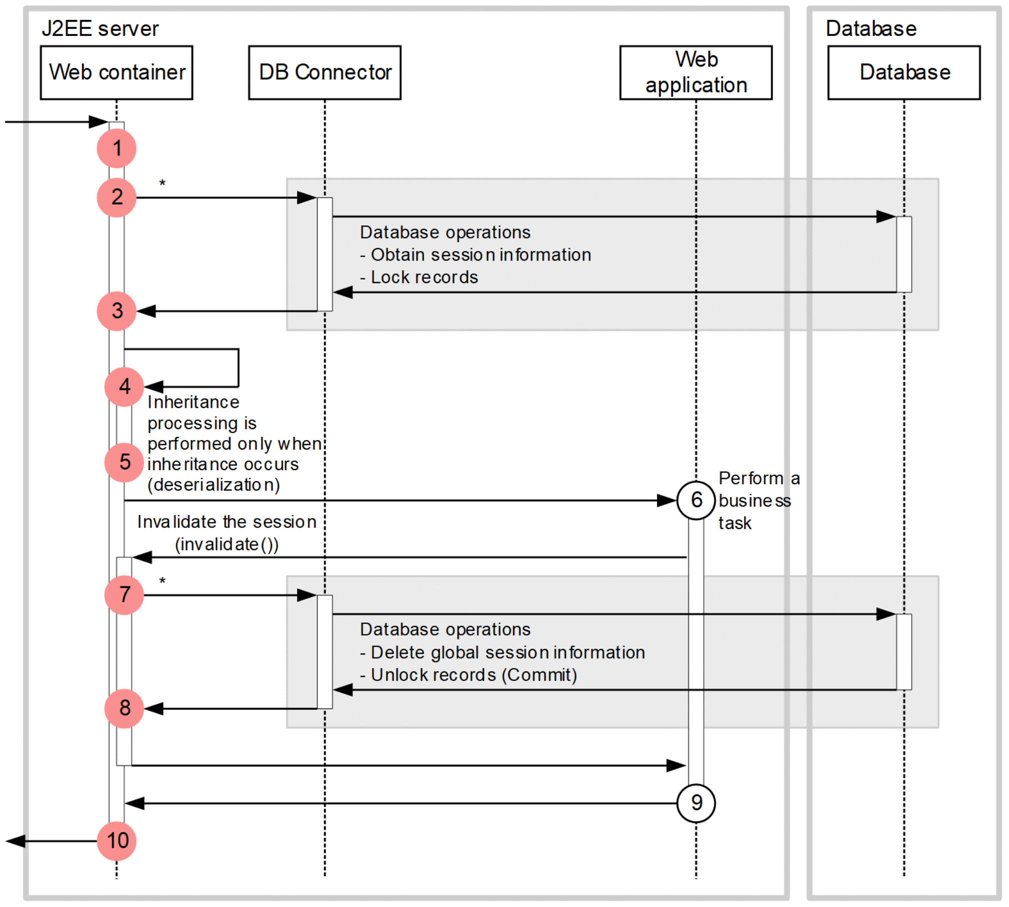
A/B: Different information is collected for the *Standard* and *Advanced* levels.

#

Corresponds to the numbers in *Figure 15-14*.

The following figure shows the trace collection points.

Figure 15–14: Trace collection points of the request processing for disabling an HTTP session (database session failover functionality)


J2EE server
Web container
DB Connector
Web application
Database
Database
1
2
*
Database operations
- Obtain session information
- Lock records
3
4
Inheritance processing is performed only when inheritance occurs (deserialization)
5
Perform a business task
6
Invalidate the session (invalidate())
7
*
Database operations
- Delete global session information
- Unlock records (Commit)
8
9
10

Legend: ● : Indicates a trace collection point. The PRF trace collection level is "Standard".

○ : Indicates a trace collection point.
The PRF trace collection level is "Standard" if a servlet is invoked.

▨ : Indicates the scope of database operations

* Actually, DB Connector will be invoked more than once.

(2) Trace information that can be collected

The following table describes the trace information that can be collected during request processing for disabling an HTTP session.

Table 15–25: Trace information that can be collected during request processing for disabling an HTTP session (database session failover functionality)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8200	A	HTTP method	URI	--
		B			*Number-of-session-ID-characters*: *session-ID*: *collection-method*
	0x8211	A	HTTP method	URI	--

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
		B			*Number-of-session-ID-characters*: *session-ID*: *collection-method*
2	0x8221	A	--	--	*Number-of-session-ID-characters*: *session-ID-received-with-the-request*
3	0x8321	A	When integrity protection mode is set Number of the record for which exclusion is acquired When integrity protection mode is disabled -1	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
4	0x8220	A	Request URL	Size (bytes) of the HTTP session property information before de-serialization	*Number-of-session-ID-characters*: *session-ID*
5	0x8320	A	Request URL	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
6	0x8202	A	Class name (JSP file name when a JSP is invoked)	--	--
		B		Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8207	B	--	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8203	B	Class name	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
7	0x8224	A	--	--	*Number-of-session-ID-characters*: *session-ID-of-the-disabled-HTTP-session*
8	0x8324	A	When integrity protection mode is set Number of the record for which exclusion is released When integrity protection mode is disabled -1	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
9	0x8302	A	Class name	--	• When normal:

No. in the figure[#]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
			(JSP file name when a JSP is invoked)		*Entrance-time* • For an exception: *Entrance-time exception-name*
		B		Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
	0x8307	B	--	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
	0x8303	B	Class name	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
	0x8316	B	Class name	Context root name	• When normal: *Entrance-time number-of-session-ID-characters*: *session-ID* • For an exception: *Entrance-time exception-name*: *number-of-session-ID-characters*: *session-ID*
10	0x8300	A	HTTP method	URI	• When normal: *Entrance-time status-code* • For an exception: *Entrance-time status-code exception-name*
		B			• When normal: *Entrance-time status-code number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID* • For an exception: *Entrance-time status-code exception-name: number-of-session-*

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
					ID-characters: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
	0x8311	A	HTTP method	URI	*Entrance-time-status-code*
		B			*Entrance-time status-code number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in *Figure 15-14*.

15.10.4 Trace collection points and trace information that can be collected during request processing for disabling an HTTP session through valid period monitoring (Trace of database session failover functionality)

This section describes the trace collection points and the trace information that can be collected during request processing for disabling an HTTP session by monitoring the valid period.

(1) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–26: Details of trace collection points of the request processing for disabling an HTTP session through valid period monitoring (database session failover functionality)

Event ID	No. in the figure#	Trace acquisition points	Level
0x8210	3	After session timeout	B
0x8225	1	Immediately before starting the valid period monitoring process of the global session information on the database	A
0x8325	2	Immediately after the termination of the valid period monitoring process of the global session information on the database	A

Legend:

A: Standard

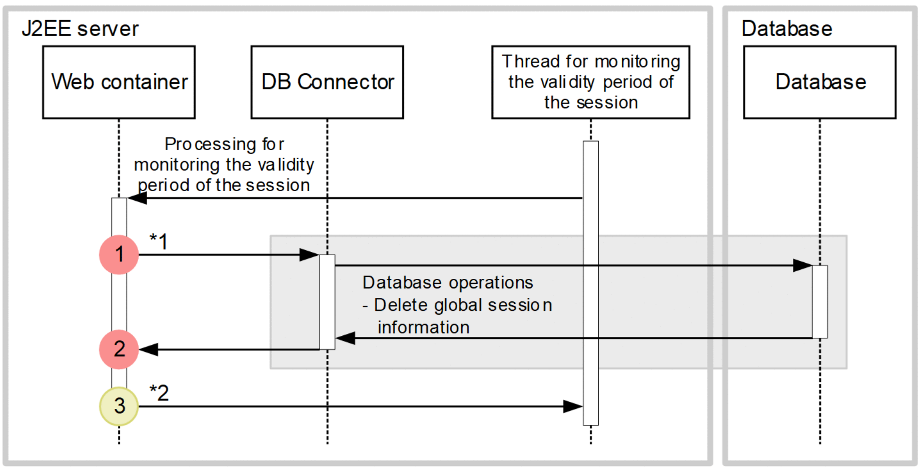
B: Advanced

#

Corresponds to the numbers in *Figure 15-15*.

The following figure shows the trace collection points.

Figure 15–15: Trace collection points of the request processing for disabling an HTTP session through valid period monitoring (database session failover functionality)


J2EE server
Web container
DB Connector
Thread for monitoring the validity period of the session
Database
Database
Processing for monitoring the validity period of the session
*1
1
Database operations
- Delete global session information
2
*2
3

Legend: 🟥 : Indicates a trace collection point. The PRF trace collection level is "Standard".

🟨 : Indicates a trace collection point. The PRF trace collection level is "Advanced".

▨ : Indicates the scope of database operations

*1 Actually, DB Connector will be invoked more than once.

*2 The output will be made as many times as the number of the deleted HTTP sessions.

(2) Trace information that can be collected

The following table describes the trace information that can be collected during request processing for disabling an HTTP session through valid period monitoring.

Table 15–27: Trace information that can be collected during request processing for disabling an HTTP session through valid period monitoring (database session failover functionality)

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8225[#2]	A	--	--	--
2	0x8325[#2]	A	When the valid period checking of the global session information is executed Number of disabled global sessions When the valid period checking of the global session information is not executed Name (IP address) of the J2EE server that currently manages the valid period checking	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
3	0x8210	B	Context root name	Session valid period: session generation time	*Number-of-session-ID-characters*: *session-ID*

Legend:

A: Standard

B: Advanced

--: Not applicable

#1

Corresponds to the numbers in *Figure 15-15*.

#2

Not output when the integrity protection mode is disabled.

15.11 Trace collection points in a JPA

This section describes the trace collection points of a JPA, and the trace information that can be collected.

15.11.1 Trace collection points and trace information that can be collected when the persistent context of application management is used

This section describes the trace collection points and the trace information that can be collected when the persistent context of application management is used.

(1) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–28: Details of trace collection points when the persistent context of application management is used

Event ID	No. in the figure#	Trace acquisition points	Level
0xA500	1	When the processing of EntityManagerFactory#createEntityManagerFactory() starts	A
0xA501	2	When the processing of EntityManagerFactory#createEntityManagerFactory() ends	A
0xA502	1	When the processing of EntityManagerFactory#createEntityManagerFactory(Map map) starts	A
0xA503	2	When the processing of EntityManagerFactory#createEntityManagerFactory(Map map) ends	A
0xA504	1	When the processing of EntityManagerFactory#isOpen() starts	A
0xA505	2	When the processing of EntityManagerFactory#isOpen() ends	A
0xA506	1	When the processing of EntityManagerFactory#close() starts	A
0xA507	2	When the processing of EntityManagerFactory#close() ends	A
0xA508	1	When the processing of EntityManager#find(Class<T> entityClass, Object primaryKey) starts	A
0xA509	2	When the processing of EntityManager#find(Class<T> entityClass, Object primaryKey) ends	A
0xA50A	1	When the processing of EntityManager#getReference(Class<T> entityClass, Object primaryKey) starts	A
0xA50B	2	When the processing of EntityManager#getReference(Class<T> entityClass, Object primaryKey) ends	A
0xA50C	1	When the processing of EntityManager#contains(Object entity) starts	A
0xA50D	2	When the processing of EntityManager#contains(Object entity) ends	A
0xA50E	1	When the processing of EntityManager#lock(Object entity, LockModeType lockMode) starts	A
0xA50F	2	When the processing of EntityManager#lock(Object entity, LockModeType lockMode) ends	A

Event ID	No. in the figure[#]	Trace acquisition points	Level
0xA510	1	When the processing of EntityManager#merge(T entity) starts	A
0xA511	2	When the processing of EntityManager#merge(T entity) ends	A
0xA512	1	When the processing of EntityManager#persist(Object entity) starts	A
0xA513	2	When the processing of EntityManager#persist(Object entity) ends	A
0xA514	1	When the processing of EntityManager#refresh(Object entity) starts	A
0xA515	2	When the processing of EntityManager#refresh(Object entity) ends	A
0xA516	1	When the processing of EntityManager#remove(Object entity) starts	A
0xA517	2	When the processing of EntityManager#remove(Object entity) ends	A
0xA518	1	When the processing of EntityManager#clear() starts	A
0xA519	2	When the processing of EntityManager#clear() ends	A
0xA51A	1	When the processing of EntityManager#flush() starts	A
0xA51B	2	When the processing of EntityManager#flush() ends	A
0xA51C	1	When the processing of EntityManager#createQuery(String qlString) starts	A
0xA51D	2	When the processing of EntityManager#createQuery(String qlString) ends	A
0xA51E	1	When the processing of EntityManager#createNamedQuery(String name) starts	A
0xA51F	2	When the processing of EntityManager#createNamedQuery(String name) ends	A
0xA520	1	When the processing of EntityManager#createNativeQuery(String sqlString) starts	A
0xA521	2	When the processing of EntityManager#createNativeQuery(String sqlString) ends	A
0xA522	1	When the processing of EntityManager#createNativeQuery(String sqlString, Class resultClass) starts	A
0xA523	2	When the processing of EntityManager#createNativeQuery(String sqlString, Class resultClass) ends	A
0xA524	1	When the processing of EntityManager#createNativeQuery(String sqlString, String resultSetMapping) starts	A
0xA525	2	When the processing of EntityManager#createNativeQuery(String sqlString, String resultSetMapping) ends	A
0xA526	1	When the processing of EntityManager#setFlushMode(FlushModeType flushMode) starts	A
0xA527	2	When the processing of EntityManager#setFlushMode(FlushModeType flushMode) ends	A
0xA528	1	When the processing of EntityManager#getFlushMode() starts	A
0xA529	2	When the processing of EntityManager#getFlushMode() ends	A
0xA52A	1	When the processing of EntityManager#joinTransaction() starts	A
0xA52B	2	When the processing of EntityManager#joinTransaction() ends	A
0xA52C	1	When the processing of EntityManager#getTransaction() starts	A

Event ID	No. in the figure#	Trace acquisition points	Level
0xA52D	2	When the processing of EntityManager#getTransaction() ends	A
0xA52E	1	When the processing of EntityManager#getDelegate() starts	A
0xA52F	2	When the processing of EntityManager#getDelegate() ends	A
0xA530	1	When the processing of EntityManager#isOpen() starts	A
0xA531	2	When the processing of EntityManager#isOpen() ends	A
0xA532	1	When the processing of EntityManager#close() starts	A
0xA533	2	When the processing of EntityManager#close() ends	A
0xA534	1	When the processing of EntityTransaction#begin() starts	A
0xA535	2	When the processing of EntityTransaction#begin() ends	A
0xA536	1	When the processing of EntityTransaction#commit() starts	A
0xA537	2	When the processing of EntityTransaction#commit() ends	A
0xA538	1	When the processing of EntityTransaction#rollback() starts	A
0xA539	2	When the processing of EntityTransaction#rollback() ends	A
0xA53A	1	When the processing of EntityTransaction#getRollbackOnly() starts	A
0xA53B	2	When the processing of EntityTransaction#getRollbackOnly() ends	A
0xA53C	1	When the processing of EntityTransaction#setRollbackOnly() starts	A
0xA53D	2	When the processing of EntityTransaction#setRollbackOnly() ends	A
0xA53E	1	When the processing of EntityTransaction#isActive() starts	A
0xA53F	2	When the processing of EntityTransaction#isActive() ends	A
0xA540	1	When the processing of Query#executeUpdate() starts	A
0xA541	2	When the processing of Query#executeUpdate() ends	A
0xA542	1	When the processing of Query#getResultList() starts	A
0xA543	2	When the processing of Query#getResultList() ends	A
0xA544	1	When the processing of Query#getSingleResult() starts	A
0xA545	2	When the processing of Query#getSingleResult() ends	A
0xA546	1	When the processing of Query#setFlushMode(FlushModeType flushMode) starts	A
0xA547	2	When the processing of Query#setFlushMode(FlushModeType flushMode) ends	A
0xA548	1	When the processing of Query#setFirstResult(int startPosition) starts	B
0xA549	2	When the processing of Query#setFirstResult(int startPosition) ends	B
0xA54A	1	When the processing of Query#setMaxResults(int maxResult) starts	B
0xA54B	2	When the processing of Query#setMaxResults(int maxResult) ends	B
0xA54C	1	When the processing of Query#setHint(String hintName, Object value) starts	B
0xA54D	2	When the processing of Query#setHint(String hintName, Object value) ends	B

Event ID	No. in the figure#	Trace acquisition points	Level
0xA54E	1	When the processing of Query#setParameter(int position, Calendar value, TemporalType temporalType) starts	B
0xA54F	2	When the processing of Query#setParameter(int position, Calendar value, TemporalType temporalType) ends	B
0xA550	1	When the processing of Query#setParameter(int position, Date value, TemporalType temporalType) starts	B
0xA551	2	When the processing of Query#setParameter(int position, Date value, TemporalType temporalType) ends	B
0xA552	1	When the processing of Query#setParameter(int position, Object value) starts	B
0xA553	2	When the processing of Query#setParameter(int position, Object value) ends	B
0xA554	1	When the processing of Query#setParameter(String name, Calendar value, TemporalType temporalType) starts	B
0xA555	2	When the processing of Query#setParameter(String name, Calendar value, TemporalType temporalType) ends	B
0xA556	1	When the processing of Query#setParameter(String name, Date value, TemporalType temporalType) starts	B
0xA557	2	When the processing of Query#setParameter(String name, Date value, TemporalType temporalType) ends	B
0xA558	1	When the processing of Query#setParameter(String name, Object value) starts	B
0xA559	2	When the processing of Query#setParameter(String name, Object value) ends	B

Legend:

A: Standard

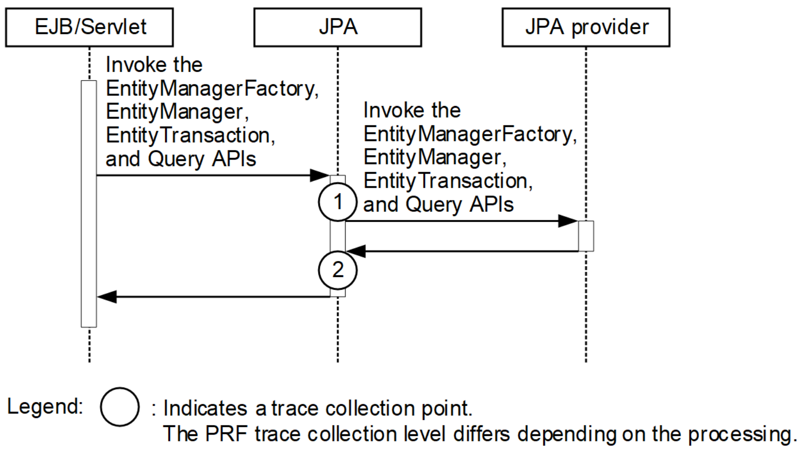
B: Advanced

#

Corresponds to the numbers in *Figure 15-16*.

The following figure shows the trace collection points.

Figure 15–16: Trace collection points when the persistent context of application management is used


EJB/Servlet
JPA
JPA provider
Invoke the EntityManagerFactory, EntityManager, EntityTransaction, and Query APIs
Invoke the EntityManagerFactory, EntityManager, EntityTransaction, and Query APIs
1
2
Legend:
: Indicates a trace collection point.
The PRF trace collection level differs depending on the processing.

(2) Trace information that can be collected

The following table describes the trace information that can be collected when the persistent context of application management is used.

Table 15–29: Trace information that can be collected when the persistent context of application management is used

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xA500	A	--	--	--
	0xA502	A	--	--	--
	0xA504	A	--	--	--
	0xA506	A	--	--	--
	0xA508	A	entity class name	--	--
	0xA50A	A	entity class name	--	--
	0xA50C	A	entity class name	--	--
	0xA50E	A	entity class name	lockMode value	--
	0xA510	A	entity class name	--	--
	0xA512	A	entity class name	--	--
	0xA514	A	entity class name	--	--
	0xA516	A	entity class name	--	--
	0xA518	A	--	--	--
	0xA51A	A	--	--	--
	0xA51C	A	--	--	--
	0xA51E	A	name	--	--
	0xA520	A	--	--	--
	0xA522	A	resultClass class name	--	--
	0xA524	A	resultSetMapping	--	--
	0xA526	A	flushMode value	--	--
	0xA528	A	--	--	--
	0xA52A	A	--	--	--
	0xA52C	A	--	--	--
	0xA52E	A	--	--	--
	0xA530	A	--	--	--
	0xA532	A	--	--	--
	0xA534	A	--	--	--
	0xA536	A	--	--	--
	0xA538	A	--	--	--

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0xA53A	A	--	--	--
	0xA53C	A	--	--	--
	0xA53E	A	--	--	--
	0xA540	A	--	--	--
	0xA542	A	--	--	--
	0xA544	A	--	--	--
	0xA546	A	flushMode value	--	--
	0xA548	B	startPosition value	--	--
	0xA54A	B	maxResult value	--	--
	0xA54C	B	hintName	value class name	--
	0xA54E	B	position value	--	--
	0xA550	B	position value	--	--
	0xA552	B	position value	--	--
	0xA554	B	name value	--	--
	0xA556	B	name value	--	--
	0xA558	B	name value	--	--
2	0xA501	A	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
	0xA503	A	--	--	
	0xA505	A	--	--	
	0xA507	A	--	--	
	0xA509	A	--	--	
	0xA50B	A	--	--	
	0xA50D	A	--	--	
	0xA50F	A	--	--	
	0xA511	A	--	--	
	0xA513	A	--	--	
	0xA515	A	--	--	
	0xA517	A	--	--	
	0xA519	A	--	--	
	0xA51B	A	--	--	
	0xA51D	A	--	--	
	0xA51F	A	--	--	
	0xA521	A	--	--	
	0xA523	A	--	--	

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0xA525	A	--	--	
	0xA527	A	--	--	
	0xA529	A	--	--	
	0xA52B	A	--	--	
	0xA52D	A	--	--	
	0xA52F	A	--	--	
	0xA531	A	--	--	
	0xA533	A	--	--	
	0xA535	A	--	--	
	0xA537	A	--	--	
	0xA539	A	--	--	
	0xA53B	A	--	--	
	0xA53D	A	--	--	
	0xA53F	A	--	--	
	0xA541	A	--	--	
	0xA543	A	--	--	
	0xA545	A	--	--	
	0xA547	A	--	--	
	0xA549	B	--	--	
	0xA54B	B	--	--	
	0xA54D	B	--	--	
	0xA54F	B	--	--	
	0xA551	B	--	--	
	0xA553	B	--	--	
	0xA555	B	--	--	
	0xA557	B	--	--	
	0xA559	B	--	--	

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in *Figure 15-16*.

15.11.2 Trace collection points and trace information that can be collected when the persistent context of container management is used

This section describes the trace collection points and the trace information that can be collected when the persistent context of container management is used. The explanation is divided into the following four categories:

- When the persistent context of the transaction scope is used in the transaction
- When the entity manager related to the persistent context of the transaction scope is used outside a transaction
- When a Query generated outside a transaction is used outside the transaction
- When an extended persistent context is used

(1) Trace collection points and PRF trace collection levels

(a) When the persistent context of the transaction scope is used in the transaction

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–30: Details of trace collection points when the persistent context of the transaction scope is used in the transaction

Event ID	No. in the figure#	Trace acquisition points	Level
0xA508	1	When the processing of EntityManager#find(Class<T>entityClass, Object primaryKey) starts	A
0xA509	6	When the processing o EntityManager#find(Class<T> entityClass, Object primaryKey) ends	A
0xA50A	1	When the processing of EntityManager#getReference(Class<T> entityClass, Object primaryKey) starts	A
0xA50B	6	When the processing of EntityManager#getReference(Class<T> entityClass, Object primaryKey) ends	A
0xA50C	1	When the processing of EntityManager#contains(Object entity) starts	A
0xA50D	6	When the processing of EntityManager#contains(Object entity) ends	A
0xA50E	1	When the processing of EntityManager#lock(Object entity, LockModeType lockMode) starts	A
0xA50F	6	When the processing of EntityManager#lock(Object entity, LockModeType lockMode) ends	A
0xA510	1	When the processing of EntityManager#merge(T entity) starts	A
0xA511	6	When the processing of EntityManager#merge(T entity) ends	A
0xA512	1	When the processing of EntityManager#persist(Object entity) starts	A
0xA513	6	When the processing of EntityManager#persist(Object entity) ends	A
0xA514	1	When the processing of EntityManager#refresh(Object entity) starts	A
0xA515	6	When the processing of EntityManager#refresh(Object entity) ends	A
0xA516	1	When the processing of EntityManager#remove(Object entity) starts	A
0xA517	6	When the processing of EntityManager#remove(Object entity) ends	A
0xA518	1	When the processing of EntityManager#clear() starts	A

Event ID	No. in the figure#	Trace acquisition points	Level
0xA519	6	When the processing of EntityManager#clear() ends	A
0xA51A	1	When the processing of EntityManager#flush() starts	A
0xA51B	6	When the processing of EntityManager#flush() ends	A
0xA51C	1	When the processing of EntityManager#createQuery(String qlString) starts	A
0xA51D	6	When the processing of EntityManager#createQuery(String qlString) ends	A
0xA51E	1	When the processing of EntityManager#createNamedQuery(String name) starts	A
0xA51F	6	When the processing of EntityManager#createNamedQuery(String name) ends	A
0xA520	1	When the processing of EntityManager#createNativeQuery(String sqlString) starts	A
0xA521	6	When the processing of EntityManager#createNativeQuery(String sqlString) ends	A
0xA522	1	When the processing of EntityManager#createNativeQuery(String sqlString, Class resultClass) starts	A
0xA523	6	When the processing of EntityManager#createNativeQuery(String sqlString, Class resultClass) ends	A
0xA524	1	When the processing of EntityManager#createNativeQuery(String sqlString, String resultSetMapping) starts	A
0xA525	6	When the processing of EntityManager#createNativeQuery(String sqlString, String resultSetMapping) ends	A
0xA526	1	When the processing of EntityManager#setFlushMode(FlushModeType flushMode) starts	A
0xA527	6	When the processing of EntityManager#setFlushMode(FlushModeType flushMode) ends	A
0xA528	1	When the processing of EntityManager#getFlushMode() starts	A
0xA529	6	When the processing of EntityManager#getFlushMode() ends	A
0xA52A	1	When the processing of EntityManager#joinTransaction() starts	A
0xA52B	6	When the processing of EntityManager#joinTransaction() ends	A
0xA52C	1	When the processing of EntityManager#getTransaction() starts	A
0xA52D	6	When the processing of EntityManager#getTransaction() ends	A
0xA52E	1	When the processing of EntityManager#getDelegate() starts	A
0xA52F	6	When the processing of EntityManager#getDelegate() ends	A
0xA530	1	When the processing of EntityManager#isOpen() starts	A
0xA531	6	When the processing of EntityManager#isOpen() ends	A
0xA532	1	When the processing of EntityManager#close() starts	A
0xA533	6	When the processing of EntityManager#close() ends	A
0xA540	1	When the processing of Query#executeUpdate() starts	A
0xA541	6	When the processing of Query#executeUpdate() ends	A

Event ID	No. in the figure#	Trace acquisition points	Level
0xA542	1	When the processing of Query#getResultList() starts	A
0xA543	6	When the processing of Query#getResultList() ends	A
0xA544	1	When the processing of Query#getSingleResult() starts	A
0xA545	6	When the processing of Query#getSingleResult() ends	A
0xA546	1	When the processing of Query#setFlushMode(FlushModeType flushMode) starts	A
0xA547	6	When the processing of Query#setFlushMode(FlushModeType flushMode) ends	A
0xA548	1	When the processing of Query#setFirstResult(int startPosition) starts	B
0xA549	6	When the processing of Query#setFirstResult(int startPosition) ends	B
0xA54A	1	When the processing of Query#setMaxResults(int maxResult) starts	B
0xA54B	6	When the processing of Query#setMaxResults(int maxResult) ends	B
0xA54C	1	When the processing of Query#setHint(String hintName, Object value) starts	B
0xA54D	6	When the processing of Query#setHint(String hintName, Object value) ends	B
0xA54E	1	When the processing of Query#setParameter(int position, Calendar value, TemporalType temporalType) starts	B
0xA54F	6	When the processing of Query#setParameter(int position, Calendar value, TemporalType temporalType) ends	B
0xA550	1	When the processing of Query#setParameter(int position, Date value, TemporalType temporalType) starts	B
0xA551	6	When the processing of Query#setParameter(int position, Date value, TemporalType temporalType) ends	B
0xA552	1	When the processing of Query#setParameter(int position, Object value) starts	B
0xA553	6	When the processing of Query#setParameter(int position, Object value) ends	B
0xA554	1	When the processing of Query#setParameter(String name, Calendar value, TemporalType temporalType) starts	B
0xA555	6	When the processing of Query#setParameter(String name, Calendar value, TemporalType temporalType) ends	B
0xA556	1	When the processing of Query#setParameter(String name, Date value, TemporalType temporalType) starts	B
0xA557	6	When the processing of Query#setParameter(String name, Date value, TemporalType temporalType) ends	B
0xA558	1	When the processing of Query#setParameter(String name, Object value) starts	B
0xA559	6	When the processing of Query#setParameter(String name, Object value) ends	B
0xA560	4	When the processing of find(Class<T> entityClass, Object primaryKey) of EntityManager of the JPA provider starts	B
0xA561	5	When the processing of find(Class<T> entityClass, Object primaryKey) of EntityManager of the JPA provider ends	B
0xA562	4	When the processing of getReference(Class<T> entityClass, Object primaryKey) of EntityManager of the JPA provider starts	B

Event ID	No. in the figure#	Trace acquisition points	Level
0xA563	5	When the processing of getReference(Class<T> entityClass, Object primaryKey) of EntityManager of the JPA provider ends	B
0xA564	4	When the processing of contains(Object entity) of EntityManager of the JPA provider starts	B
0xA565	5	When the processing of contains(Object entity) of EntityManager of the JPA provider ends	B
0xA566	4	When the processing of lock(Object entity, LockModeType lockMode) of EntityManager of the JPA provider starts	B
0xA567	5	When the processing of lock(Object entity, LockModeType lockMode) of EntityManager of the JPA provider ends	B
0xA568	4	When the processing of merge(T entity) of EntityManager of the JPA provider starts	B
0xA569	5	When the processing of merge(T entity) of EntityManager of the JPA provider ends	B
0xA56A	4	When the processing of persist(Object entity) of EntityManager of the JPA provider starts	B
0xA56B	5	When the processing of persist(Object entity) of EntityManager of the JPA provider ends	B
0xA56C	4	When the processing of refresh(Object entity) of EntityManager of the JPA provider starts	B
0xA56D	5	When the processing of refresh(Object entity) of EntityManager of the JPA provider ends	B
0xA56E	4	When the processing of remove(Object entity) of EntityManager of the JPA provider starts	B
0xA56F	5	When the processing of remove(Object entity) of EntityManager of the JPA provider ends	B
0xA570	4	When the processing of clear() of EntityManager of the JPA provider starts	B
0xA571	5	When the processing of clear() of EntityManager of the JPA provider ends	B
0xA572	4	When the processing of flush() of EntityManager of the JPA provider starts	B
0xA573	5	When the processing of flush() of EntityManager of the JPA provider ends	B
0xA574	4	When the processing of createQuery(String qlString) of EntityManager of the JPA provider starts	B
0xA575	5	When the processing of createQuery(String qlString) of EntityManager of the JPA provider ends	B
0xA576	4	When the processing of createNamedQuery(String name) of EntityManager of the JPA provider starts	B
0xA577	5	When the processing of createNamedQuery(String name) of EntityManager of the JPA provider ends	B
0xA578	4	When the processing of createNativeQuery(String sqlString) of EntityManager of the JPA provider starts	B
0xA579	5	When the processing of createNativeQuery(String sqlString) of EntityManager of the JPA provider ends	B
0xA57A	4	When the processing of createNativeQuery(String sqlString, Class resultClass) of EntityManager of the JPA provider starts	B

Event ID	No. in the figure#	Trace acquisition points	Level
0xA57B	5	When the processing of createNativeQuery(String sqlString, Class resultClass) of EntityManager of the JPA provider ends	B
0xA57C	4	When the processing of createNativeQuery(String sqlString, String resultSetMapping) of EntityManager of the JPA provider starts	B
0xA57D	5	When the processing of createNativeQuery(String sqlString, String resultSetMapping) of EntityManager of the JPA provider ends	B
0xA57E	4	When the processing of setFlushMode(FlushModeType flushMode) of EntityManager of the JPA provider starts	B
0xA57F	5	When the processing of setFlushMode(FlushModeType flushMode) of EntityManager of the JPA provider ends	B
0xA580	4	When the processing of getFlushMode() of EntityManager of the JPA provider starts	B
0xA581	5	When the processing of getFlushMode() of EntityManager of the JPA provider ends	B
0xA582	4	When the processing of joinTransaction() of EntityManager of the JPA provider starts	B
0xA583	5	When the processing of joinTransaction() of EntityManager of the JPA provider ends	B
0xA584	4	When the processing of isOpen() of EntityManager of the JPA provider starts	B
0xA585	5	When the processing of isOpen() of EntityManager of the JPA provider ends	B
0xA586	4	When the processing of executeUpdate() of Query of the JPA provider starts	B
0xA587	5	When the processing of executeUpdate() of Query of the JPA provider ends	B
0xA588	4	When the processing of getResultList() of Query of the JPA provider starts	B
0xA589	5	When the processing of getResultList() of Query of the JPA provider ends	B
0xA58A	4	When the processing of getSingleResult() of Query of the JPA provider starts	B
0xA58B	5	When the processing of getSingleResult() of Query of the JPA provider ends	B
0xA58C	4	When the processing of setFlushMode(FlushModeType flushMode) of Query of the JPA provider starts	B
0xA58D	5	When the processing of setFlushMode(FlushModeType flushMode) of Query of the JPA provider ends	B
0xA58E	4	When the processing of setFirstResult(int startPosition) of Query of the JPA provider starts	B
0xA58F	5	When the processing of setFirstResult(int startPosition) of Query of the JPA provider ends	B
0xA590	4	When the processing of setMaxResults(int maxResult) of Query of the JPA provider starts	B
0xA591	5	When the processing of setMaxResults(int maxResult) of Query of the JPA provider ends	B
0xA592	4	When the processing of setHint(String hintName, Object value) of Query of the JPA provider starts	B
0xA593	5	When the processing of setHint(String hintName, Object value) of Query of the JPA provider ends	B

Event ID	No. in the figure#	Trace acquisition points	Level
0xA594	4	When the processing of setParameter(int position, Calendar value, TemporalType temporalType) of Query of the JPA provider starts	B
0xA595	5	When the processing of setParameter(int position, Calendar value, TemporalType temporalType) of Query of the JPA provider ends	B
0xA596	4	When the processing of setParameter(int position, Date value, TemporalType temporalType) of Query of the JPA provider starts	B
0xA597	5	When the processing of setParameter(int position, Date value, TemporalType temporalType) of Query of the JPA provider ends	B
0xA598	4	When the processing of setParameter(int position, Object value) of Query of the JPA provider starts	B
0xA599	5	When the processing of setParameter(int position, Object value) of Query of the JPA provider ends	B
0xA59A	4	When the processing of setParameter(String name, Calendar value, TemporalType temporalType) of Query of the JPA provider starts	B
0xA59B	5	When the processing of setParameter(String name, Calendar value, TemporalType temporalType) of Query of the JPA provider ends	B
0xA59C	4	When the processing of setParameter(String name, Date value, TemporalType temporalType) of Query of the JPA provider starts	B
0xA59D	5	When the processing of setParameter(String name, Date value, TemporalType temporalType) of Query of the JPA provider ends	B
0xA59E	4	When the processing of setParameter(String name, Object value) of Query of the JPA provider starts	B
0xA59F	5	When the processing of setParameter(String name, Object value) of Query of the JPA provider ends	B
0xA5A0	2	At the time of start of the processing for generating the persistent context, when the persistent context of the transaction scope is used	A
0xA5A1	3	At the time of termination of the processing for generating the persistent context, when the persistent context of the transaction scope is used	A
0xA5A2	7	At the time of start of the processing for discarding the persistent context, when the persistent context of the transaction scope is used	A
0xA5A3	8	At the time of termination of the processing for discarding the persistent context, when the persistent context of the transaction scope is used	A

Legend:

A: Standard

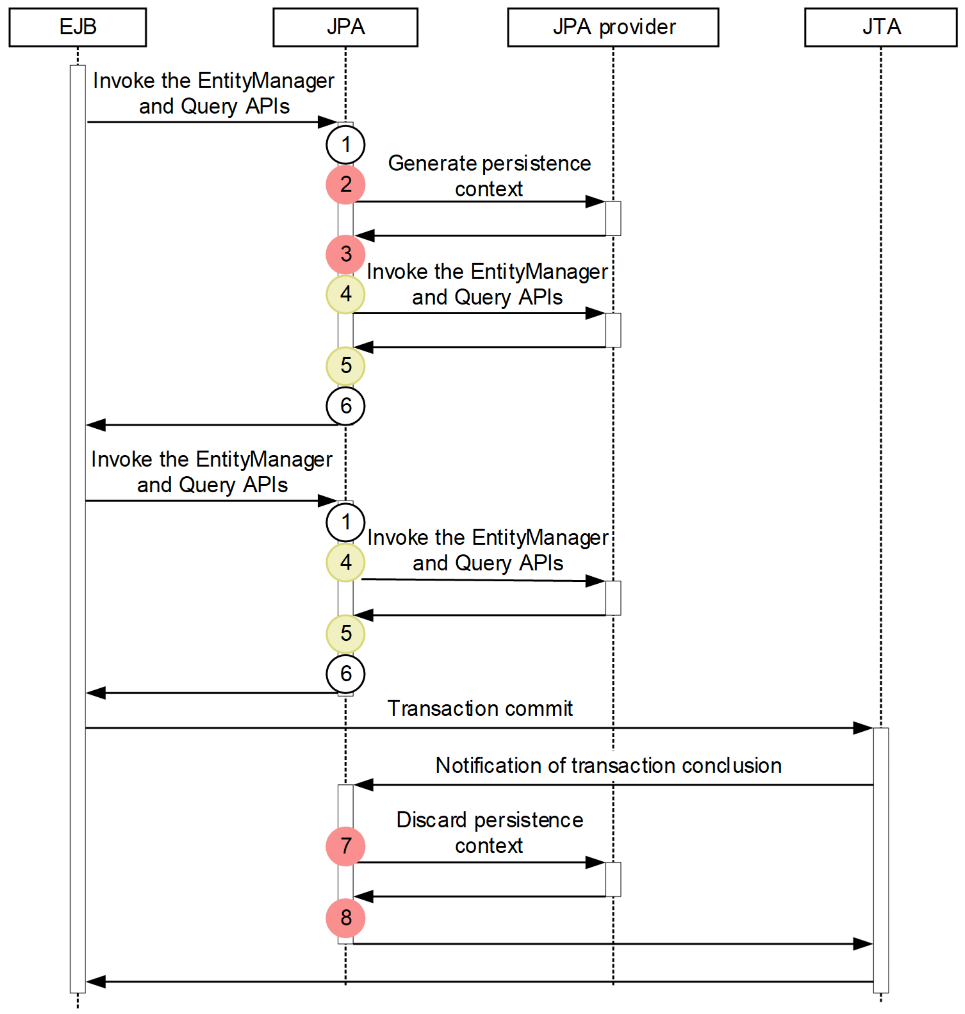
B: Advanced

#

Corresponds to the numbers in *Figure 15-17*.

The following figure shows the trace collection points.

Figure 15–17: Trace collection points when the persistent context of the transaction scope is used in the transaction


EJB
JPA
JPA provider
JTA
Invoke the EntityManager and Query APIs
1
2
Generate persistence context
3
Invoke the EntityManager and Query APIs
4
5
6
Invoke the EntityManager and Query APIs
1
Invoke the EntityManager and Query APIs
4
5
6
Transaction commit
Notification of transaction conclusion
Discard persistence context
7
8

Legend: 🔴 : Indicates a trace collection point. The PRF trace collection level is "Standard".

🟡 : Indicates a trace collection point. The PRF trace collection level is "Advanced".

◯ : Indicates a trace collection point. The PRF trace collection level differs depending on the processing.

(b) When the entity manager related to the persistent context of the transaction scope is used outside a transaction

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–31: Details of trace collection points when the entity manager related to the persistent context of the transaction scope is used outside a transaction

Event ID	No. in the figure#	Trace acquisition points	Level
0xA508	1	When the processing of EntityManager#find(Class<T> entityClass, Object primaryKey) starts	A
0xA509	8	When the processing of EntityManager#find(Class<T> entityClass, Object primaryKey) ends	A
0xA50A	1	When the processing of EntityManager#getReference(Class<T> entityClass, Object primaryKey) starts	A

Event ID	No. in the figure#	Trace acquisition points	Level
0xA50B	8	When the processing of EntityManager#getReference(Class<T> entityClass, Object primaryKey) ends	A
0xA50C	1	When the processing of EntityManager#contains(Object entity) starts	A
0xA50D	8	When the processing of EntityManager#contains(Object entity) ends	A
0xA50E	1	When the processing of EntityManager#lock(Object entity, LockModeType lockMode) starts	A
0xA50F	8	When the processing of EntityManager#lock(Object entity, LockModeType lockMode) ends	A
0xA510	1	When the processing of EntityManager#merge(T entity) starts	A
0xA511	8	When the processing of EntityManager#merge(T entity) ends	A
0xA512	1	When the processing of EntityManager#persist(Object entity) starts	A
0xA513	8	When the processing of EntityManager#persist(Object entity) ends	A
0xA514	1	When the processing of EntityManager#refresh(Object entity) starts	A
0xA515	8	When the processing of EntityManager#refresh(Object entity) ends	A
0xA516	1	When the processing of EntityManager#remove(Object entity) starts	A
0xA517	8	When the processing of EntityManager#remove(Object entity) ends	A
0xA518	1	When the processing of EntityManager#clear() starts	A
0xA519	8	When the processing of EntityManager#clear() ends	A
0xA51A	1	When the processing of EntityManager#flush() starts	A
0xA51B	8	When the processing of EntityManager#flush() ends	A
0xA526	1	When the processing of EntityManager#setFlushMode(FlushModeType flushMode) starts	A
0xA527	8	When the processing of EntityManager#setFlushMode(FlushModeType flushMode) ends	A
0xA528	1	When the processing of EntityManager#getFlushMode() starts	A
0xA529	8	When the processing of EntityManager#getFlushMode() ends	A
0xA52A	1	When the processing of EntityManager#joinTransaction() starts	A
0xA52B	8	When the processing of EntityManager#joinTransaction() ends	A
0xA52C	1	When the processing of EntityManager#getTransaction() starts	A
0xA52D	8	When the processing of EntityManager#getTransaction() ends	A
0xA52E	1	When the processing of EntityManager#getDelegate() starts	A
0xA52F	8	When the processing of EntityManager#getDelegate() ends	A
0xA530	1	When the processing of EntityManager#isOpen() starts	A
0xA531	8	When the processing of EntityManager#isOpen() ends	A
0xA532	1	When the processing of EntityManager#close() starts	A
0xA533	8	When the processing of EntityManager#close() ends	A
0xA560	4	When the processing of find(Class<T> entityClass, Object primaryKey) of EntityManager of the JPA provider starts	B

Event ID	No. in the figure[#]	Trace acquisition points	Level
0xA561	5	When the processing of find(Class<T> entityClass, Object primaryKey) of EntityManager of the JPA provider ends	B
0xA562	4	When the processing of getReference(Class<T> entityClass, Object primaryKey) of EntityManager of the JPA provider starts	B
0xA563	5	When the processing of getReference(Class<T> entityClass, Object primaryKey) of EntityManager of the JPA provider ends	B
0xA564	4	When the processing of contains(Object entity) of EntityManager of the JPA provider starts	B
0xA565	5	When the processing of contains(Object entity) of EntityManager of the JPA provider ends	B
0xA566	4	When the processing of lock(Object entity, LockModeType lockMode) of EntityManager of the JPA provider starts	B
0xA567	5	When the processing of lock(Object entity, LockModeType lockMode) of EntityManager of the JPA provider ends	B
0xA568	4	When the processing of merge(T entity) of EntityManager of the JPA provider starts	B
0xA569	5	When the processing of merge(T entity) of EntityManager of the JPA provider ends	B
0xA56A	4	When the processing of persist(Object entity) of EntityManager of the JPA provider starts	B
0xA56B	5	When the processing of persist(Object entity) of EntityManager of the JPA provider ends	B
0xA56C	4	When the processing of refresh(Object entity) of EntityManager of the JPA provider starts	B
0xA56D	5	When the processing of refresh(Object entity) of EntityManager of the JPA provider ends	B
0xA56E	4	When the processing of remove(Object entity) of EntityManager of the JPA provider starts	B
0xA56F	5	When the processing of remove(Object entity) of EntityManager of the JPA provider ends	B
0xA570	4	When the processing of clear() of EntityManager of the JPA provider starts	B
0xA571	5	When the processing of clear() of EntityManager of the JPA provider ends	B
0xA572	4	When the processing of flush() of EntityManager of the JPA provider starts	B
0xA573	5	When the processing of flush() of EntityManager of the JPA provider ends	B
0xA57E	4	When the processing of setFlushMode(FlushModeType flushMode) of EntityManager of the JPA provider starts	B
0xA57F	5	When the processing of setFlushMode(FlushModeType flushMode) of EntityManager of the JPA provider ends	B
0xA580	4	When the processing of getFlushMode() of EntityManager of the JPA provider starts	B
0xA581	5	When the processing of getFlushMode() of EntityManager of the JPA provider ends	B
0xA582	4	When the processing of joinTransaction() of EntityManager of the JPA provider starts	B

Event ID	No. in the figure#	Trace acquisition points	Level
0xA583	5	When the processing of joinTransaction() of EntityManager of the JPA provider ends	B
0xA584	4	When the processing of isOpen() of EntityManager of the JPA provider starts	B
0xA585	5	When the processing of isOpen() of EntityManager of the JPA provider ends	B
0xA5A4	2	At the time of start of the processing for generating the persistent context, when an API that is not generated in the Query of EntityManager of the transaction scope is used while the transaction does not exist	A
0xA5A5	3	At the time of termination of the processing for generating the persistent context, when an API that is not generated in the Query of EntityManager of the transaction scope is used while the transaction does not exist	A
0xA5A6	6	At the time of start of the processing for discarding the persistent context generated when an API that is not generated in the Query of EntityManager of the transaction scope is used while the transaction does not exist	A
0xA5A7	7	At the time of termination of the processing for discarding the persistent context generated when an API that is not generated in the Query of EntityManager of the transaction scope is used while the transaction does not exist	A

Legend:

A: Standard

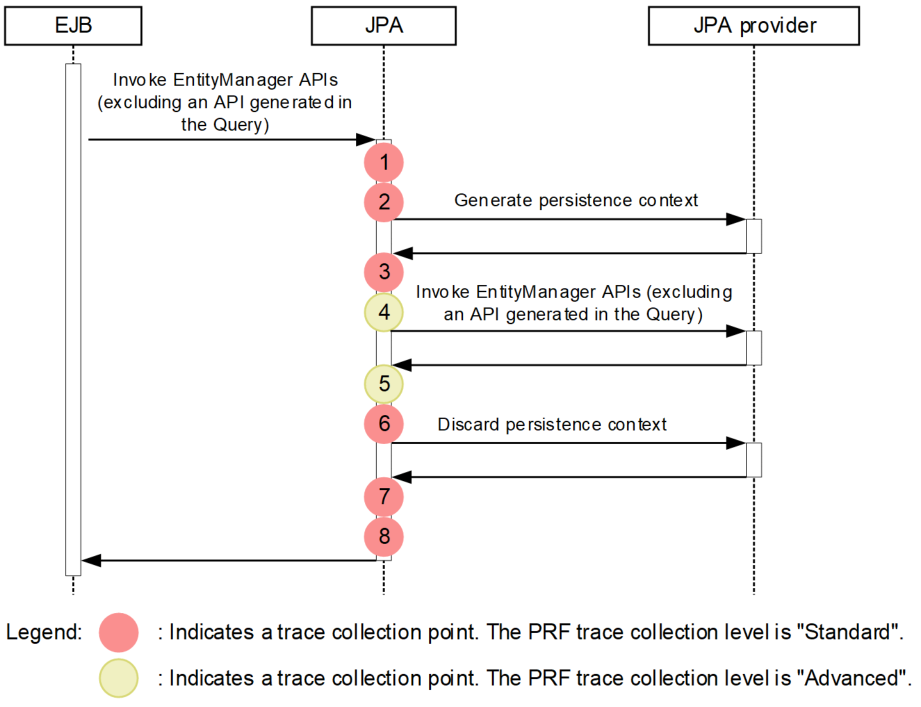
B: Advanced

#

Corresponds to the numbers in *Figure 15-18*.

The following figure shows the trace collection points.

Figure 15–18: Trace collection points when the entity manager related to the persistent context of the transaction scope is used outside a transaction


EJB
JPA
JPA provider
Invoke EntityManager APIs (excluding an API generated in the Query)
1
2
Generate persistence context
3
4
Invoke EntityManager APIs (excluding an API generated in the Query)
5
6
Discard persistence context
7
8
Legend:
: Indicates a trace collection point. The PRF trace collection level is "Standard".
: Indicates a trace collection point. The PRF trace collection level is "Advanced".

(c) When a Query generated outside a transaction is used outside the transaction

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–32: Details of trace collection points when a Query generated outside a transaction is used outside the transaction

Event ID	No. in the figure#	Trace acquisition points	Level
0xA51C	1	When the processing of EntityManager#createQuery(String qlString) starts	A
0xA51D	6	When the processing of EntityManager#createQuery(String qlString) ends	A
0xA51E	1	When the processing of EntityManager#createNamedQuery(String name) starts	A
0xA51F	6	When the processing of EntityManager#createNamedQuery(String name) ends	A
0xA520	1	When the processing of EntityManager#createNativeQuery(String sqlString) starts	A
0xA521	6	When the processing of EntityManager#createNativeQuery(String sqlString) ends	A
0xA522	1	When the processing of EntityManager#createNativeQuery(String sqlString, Class resultClass) starts	A
0xA523	6	When the processing of EntityManager#createNativeQuery(String sqlString, Class resultClass) ends	A
0xA524	1	When the processing of EntityManager#createNativeQuery(String sqlString, String resultSetMapping) starts	A
0xA525	6	When the processing of EntityManager#createNativeQuery(String sqlString, String resultSetMapping) ends	A
0xA540	1, 11	When the processing of Query#executeUpdate() starts	A
0xA541	6, 16	When the processing of Query#executeUpdate() ends	A
0xA542	1, 11	When the processing of Query#getResultList() starts	A
0xA543	6, 16	When the processing of Query#getResultList() ends	A
0xA544	1, 11	When the processing of Query#getSingleResult() starts	A
0xA545	6, 16	When the processing of Query#getSingleResult() ends	A
0xA546	1, 7	When the processing of Query#setFlushMode(FlushModeType flushMode) starts	A
0xA547	6, 10	When the processing of Query#setFlushMode(FlushModeType flushMode) ends	A
0xA548	1, 7	When the processing of Query#setFirstResult(int startPosition) starts	B
0xA549	6, 10	When the processing of Query#setFirstResult(int startPosition) ends	B
0xA54A	1, 7	When the processing of Query#setMaxResults(int maxResult) starts	B
0xA54B	6, 10	When the processing of Query#setMaxResults(int maxResult) ends	B
0xA54C	1, 7	When the processing of Query#setHint(String hintName, Object value) starts	B
0xA54D	6, 10	When the processing of Query#setHint(String hintName, Object value) ends	B
0xA54E	1, 7	When the processing of Query#setParameter(int position, Calendar value, TemporalType temporalType) starts	B

Event ID	No. in the figure#	Trace acquisition points	Level
0xA54F	6, 10	When the processing of Query#setParameter(int position, Calendar value, TemporalType temporalType) ends	B
0xA550	1, 7	When the processing of Query#setParameter(int position, Date value, TemporalType temporalType) starts	B
0xA551	6, 10	When the processing of Query#setParameter(int position, Date value, TemporalType temporalType) ends	B
0xA552	1, 7	When the processing of Query#setParameter(int position, Object value) starts	B
0xA553	6, 10	When the processing of Query#setParameter(int position, Object value) ends	B
0xA554	1, 7	When the processing of Query#setParameter(String name, Calendar value, TemporalType temporalType) starts	B
0xA555	6, 10	When the processing of Query#setParameter(String name, Calendar value, TemporalType temporalType) ends	B
0xA556	1, 7	When the processing of Query#setParameter(String name, Date value, TemporalType temporalType) starts	B
0xA557	6, 10	When the processing of Query#setParameter(String name, Date value, TemporalType temporalType) ends	B
0xA558	1, 7	When the processing of Query#setParameter(String name, Object value) starts	B
0xA559	6, 10	When the processing of Query#setParameter(String name, Object value) ends	B
0xA574	4	When the processing of createQuery(String qlString) of EntityManager of the JPA provider starts	B
0xA575	5	When the processing of createQuery(String qlString) of EntityManager of the JPA provider ends	B
0xA576	4	When the processing of createNamedQuery(String name) of EntityManager of the JPA provider starts	B
0xA577	5	When the processing of createNamedQuery(String name) of EntityManager of the JPA provider ends	B
0xA578	4	When the processing of createNativeQuery(String sqlString) of EntityManager of the JPA provider starts	B
0xA579	5	When the processing of createNativeQuery(String sqlString) of EntityManager of the JPA provider ends	B
0xA57A	4	When the processing of createNativeQuery(String sqlString, Class resultClass) of EntityManager of the JPA provider starts	B
0xA57B	5	When the processing of createNativeQuery(String sqlString, Class resultClass) of EntityManager of the JPA provider ends	B
0xA57C	4	When the processing of createNativeQuery(String sqlString, String resultSetMapping) of EntityManager of the JPA provider starts	B
0xA57D	5	When the processing of createNativeQuery(String sqlString, String resultSetMapping) of EntityManager of the JPA provider ends	B
0xA586	4, 12	When the processing of executeUpdate() of Query of the JPA provider starts	B
0xA587	5, 13	When the processing of executeUpdate() of Query of the JPA provider ends	B
0xA588	4, 12	When the processing of getResultList() of Query of the JPA provider starts	B
0xA589	5, 13	When the processing of getResultList() of Query of the JPA provider ends	B

Event ID	No. in the figure#	Trace acquisition points	Level
0xA58A	4, 12	When the processing of getSingleResult() of Query of the JPA provider starts	B
0xA58B	5, 13	When the processing of getSingleResult() of Query of the JPA provider ends	B
0xA58C	4, 8	When the processing of setFlushMode(FlushModeType flushMode) of Query of the JPA provider starts	B
0xA58D	5, 9	When the processing of setFlushMode(FlushModeType flushMode) of Query of the JPA provider ends	B
0xA58E	4, 8	When the processing of setFirstResult(int startPosition) of Query of the JPA provider starts	B
0xA58F	5, 9	When the processing of setFirstResult(int startPosition) of Query of the JPA provider ends	B
0xA590	4, 8	When the processing of setMaxResults(int maxResult) of Query of the JPA provider starts	B
0xA591	5, 9	When the processing of setMaxResults(int maxResult) of Query of the JPA provider ends	B
0xA592	4, 8	When the processing of setHint(String hintName, Object value) of Query of the JPA provider starts	B
0xA593	5, 9	When the processing of setHint(String hintName, Object value) of Query of the JPA provider ends	B
0xA594	4, 8	When the processing of setParameter(int position, Calendar value, TemporalType temporalType) of Query of the JPA provider starts	B
0xA595	5, 9	When the processing of setParameter(int position, Calendar value, TemporalType temporalType) of Query of the JPA provider ends	B
0xA596	4, 8	When the processing of setParameter(int position, Date value, TemporalType temporalType) of Query of the JPA provider starts	B
0xA597	5, 9	When the processing of setParameter(int position, Date value, TemporalType temporalType) of Query of the JPA provider ends	B
0xA598	4, 8	When the processing of setParameter(int position, Object value) of Query of the JPA provider starts	B
0xA599	5, 9	When the processing of setParameter(int position, Object value) of Query of the JPA provider ends	B
0xA59A	4, 8	When the processing of setParameter(String name, Calendar value, TemporalType temporalType) of Query of the JPA provider starts	B
0xA59B	5, 9	When the processing of setParameter(String name, Calendar value, TemporalType temporalType) of Query of the JPA provider ends	B
0xA59C	4, 8	When the processing of setParameter(String name, Date value, TemporalType temporalType) of Query of the JPA provider starts	B
0xA59D	5, 9	When the processing of setParameter(String name, Date value, TemporalType temporalType) of Query of the JPA provider ends	B
0xA59E	4, 8	When the processing of setParameter(String name, Object value) of Query of the JPA provider starts	B
0xA59F	5, 9	When the processing of setParameter(String name, Object value) of Query of the JPA provider ends	B
0xA5A8	2	At the time of start of the processing for generating the persistent context, when an API generated in the Query of EntityManager of the transaction	A

Event ID	No. in the figure#	Trace acquisition points	Level
		scope, or an API of the same Query object after searching or updating with the Query is used while the transaction does not exist	
0xA5A9	3	At the time of termination of the processing for generating the persistent context, when an API generated in the Query of EntityManager of the transaction scope, or an API of the same Query object after searching or updating with the Query is used while the transaction does not exist	A
0xA5AA	14	At the time of start of the processing for discarding the persistent context generated when an API generated in the Query of EntityManager of the transaction scope, or an API of the same Query object after searching or updating with the Query is used while the transaction does not exist	A
0xA5AB	15	At the time of termination of the processing for discarding the persistent context generated when an API generated in the Query of EntityManager of the transaction scope, or an API of the same Query object after searching or updating with the Query is used while the transaction does not exist	A

Legend:

A: Standard

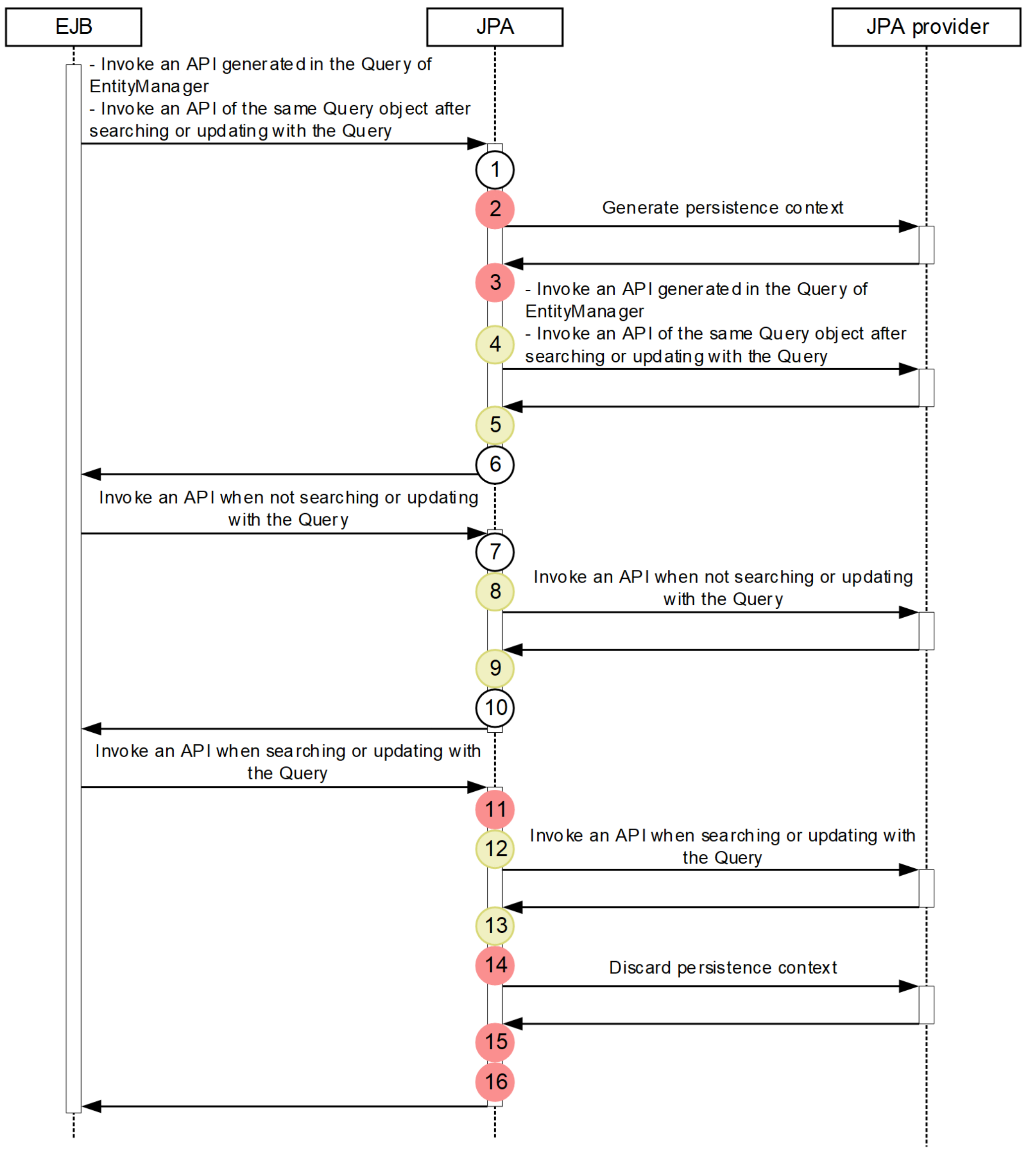
B: Advanced

#

Corresponds to the numbers in *Figure 15-19*.

The following figure shows the trace collection points.

Figure 15–19: Trace collection points when a Query generated outside a transaction is used outside the transaction


EJB
JPA
JPA provider
- Invoke an API generated in the Query of EntityManager
- Invoke an API of the same Query object after searching or updating with the Query
1
2
Generate persistence context
3
- Invoke an API generated in the Query of EntityManager
- Invoke an API of the same Query object after searching or updating with the Query
4
5
6
Invoke an API when not searching or updating with the Query
7
8
Invoke an API when not searching or updating with the Query
9
10
Invoke an API when searching or updating with the Query
11
12
Invoke an API when searching or updating with the Query
13
14
Discard persistence context
15
16

Legend: 🔴 : Indicates a trace collection point. The PRF trace collection level is "Standard".

🟡 : Indicates a trace collection point. The PRF trace collection level is "Advanced".

○ : Indicates a trace collection point.
The PRF trace collection level differs depending on the processing.

(d) When an extended persistent context is used

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–33: Details of trace collection points when an extended persistent context is used

Event ID	No. in the figure#	Trace acquisition points	Level
0xA508	3	When the processing of EntityManager#find(Class<T> entityClass, Object primaryKey) starts	A
0xA509	6	When the processing of EntityManager#find(Class<T> entityClass, Object primaryKey) ends	A

Event ID	No. in the figure#	Trace acquisition points	Level
0xA50A	3	When the processing of EntityManager#getReference(Class<T> entityClass, Object primaryKey) starts	A
0xA50B	6	When the processing of EntityManager#getReference(Class<T> entityClass, Object primaryKey) ends	A
0xA50C	3	When the processing of EntityManager#contains(Object entity) starts	A
0xA50D	6	When the processing of EntityManager#contains(Object entity) ends	A
0xA50E	3	When the processing of EntityManager#lock(Object entity, LockModeType lockMode) starts	A
0xA50F	6	When the processing of EntityManager#lock(Object entity, LockModeType lockMode) ends	A
0xA510	3	When the processing of EntityManager#merge(T entity) starts	A
0xA511	6	When the processing of EntityManager#merge(T entity) ends	A
0xA512	3	When the processing of EntityManager#persist(Object entity) starts	A
0xA513	6	When the processing of EntityManager#persist(Object entity) ends	A
0xA514	3	When the processing of EntityManager#refresh(Object entity) starts	A
0xA515	6	When the processing of EntityManager#refresh(Object entity) ends	A
0xA516	3	When the processing of EntityManager#remove(Object entity) starts	A
0xA517	6	When the processing of EntityManager#remove(Object entity) ends	A
0xA518	3	When the processing of EntityManager#clear() starts	A
0xA519	6	When the processing of EntityManager#clear() ends	A
0xA51A	3	When the processing of EntityManager#flush() starts	A
0xA51B	6	When the processing of EntityManager#flush() ends	A
0xA51C	3	When the processing of EntityManager#createQuery(String qlString) starts	A
0xA51D	6	When the processing of EntityManager#createQuery(String qlString) ends	A
0xA51E	3	When the processing of EntityManager#createNamedQuery(String name) starts	A
0xA51F	6	When the processing of EntityManager#createNamedQuery(String name) ends	A
0xA520	3	When the processing of EntityManager#createNativeQuery(String sqlString) starts	A
0xA521	6	When the processing of EntityManager#createNativeQuery(String sqlString) ends	A
0xA522	3	When the processing of EntityManager#createNativeQuery(String sqlString, Class resultClass) starts	A
0xA523	6	When the processing of EntityManager#createNativeQuery(String sqlString, Class resultClass) ends	A
0xA524	3	When the processing of EntityManager#createNativeQuery(String sqlString, String resultSetMapping) starts	A
0xA525	6	When the processing of EntityManager#createNativeQuery(String sqlString, String resultSetMapping) ends	A

Event ID	No. in the figure#	Trace acquisition points	Level
0xA526	3	When the processing of EntityManager#setFlushMode(FlushModeType flushMode) starts	A
0xA527	6	When the processing of EntityManager#setFlushMode(FlushModeType flushMode) ends	A
0xA528	3	When the processing of EntityManager#getFlushMode() starts	A
0xA529	6	When the processing of EntityManager#getFlushMode() ends	A
0xA52A	3	When the processing of EntityManager#joinTransaction() starts	A
0xA52B	6	When the processing of EntityManager#joinTransaction() ends	A
0xA52C	3	When the processing of EntityManager#getTransaction() starts	A
0xA52D	6	When the processing of EntityManager#getTransaction() ends	A
0xA52E	3	When the processing of EntityManager#getDelegate() starts	A
0xA52F	6	When the processing of EntityManager#getDelegate() ends	A
0xA530	3	When the processing of EntityManager#isOpen() starts	A
0xA531	6	When the processing of EntityManager#isOpen() ends	A
0xA532	3	When the processing of EntityManager#close() starts	A
0xA533	6	When the processing of EntityManager#close() ends	A
0xA540	3	When the processing of Query#executeUpdate() starts	A
0xA541	6	When the processing of Query#executeUpdate() ends	A
0xA542	3	When the processing of Query#getResultList() starts	A
0xA543	6	When the processing of Query#getResultList() ends	A
0xA544	3	When the processing of Query#getSingleResult() starts	A
0xA545	6	When the processing of Query#getSingleResult() ends	A
0xA546	3	When the processing of Query#setFlushMode(FlushModeType flushMode) starts	A
0xA547	6	When the processing of Query#setFlushMode(FlushModeType flushMode) ends	A
0xA548	3	When the processing of Query#setFirstResult(int startPosition) starts	B
0xA549	6	When the processing of Query#setFirstResult(int startPosition) ends	B
0xA54A	3	When the processing of Query#setMaxResults(int maxResult) starts	B
0xA54B	6	When the processing of Query#setMaxResults(int maxResult) ends	B
0xA54C	3	When the processing of Query#setHint(String hintName, Object value) starts	B
0xA54D	6	When the processing of Query#setHint(String hintName, Object value) ends	B
0xA54E	3	When the processing of Query#setParameter(int position, Calendar value, TemporalType temporalType) starts	B
0xA54F	6	When the processing of Query#setParameter(int position, Calendar value, TemporalType temporalType) ends	B
0xA550	3	When the processing of Query#setParameter(int position, Date value, TemporalType temporalType) starts	B

Event ID	No. in the figure#	Trace acquisition points	Level
0xA551	6	When the processing of Query#setParameter(int position, Date value, TemporalType temporalType) ends	B
0xA552	3	When the processing of Query#setParameter(int position, Object value) starts	B
0xA553	6	When the processing of Query#setParameter(int position, Object value) ends	B
0xA554	3	When the processing of Query#setParameter(String name, Calendar value, TemporalType temporalType) starts	B
0xA555	6	When the processing of Query#setParameter(String name, Calendar value, TemporalType temporalType) ends	B
0xA556	3	When the processing of Query#setParameter(String name, Date value, TemporalType temporalType) starts	B
0xA557	6	When the processing of Query#setParameter(String name, Date value, TemporalType temporalType) ends	B
0xA558	3	When the processing of Query#setParameter(String name, Object value) starts	B
0xA559	6	When the processing of Query#setParameter(String name, Object value) ends	B
0xA560	4	When the processing of find(Class<T> entityClass, Object primaryKey) of EntityManager of the JPA provider starts	B
0xA561	5	When the processing of find(Class<T> entityClass, Object primaryKey) of EntityManager of the JPA provider ends	B
0xA562	4	When the processing of getReference(Class<T> entityClass, Object primaryKey) of EntityManager of the JPA provider starts	B
0xA563	5	When the processing of getReference(Class<T> entityClass, Object primaryKey) of EntityManager of the JPA provider ends	B
0xA564	4	When the processing of contains(Object entity) of EntityManager of the JPA provider starts	B
0xA565	5	When the processing of contains(Object entity) of EntityManager of the JPA provider ends	B
0xA566	4	When the processing of lock(Object entity, LockModeType lockMode) of EntityManager of the JPA provider starts	B
0xA567	5	When the processing of lock(Object entity, LockModeType lockMode) of EntityManager of the JPA provider ends	B
0xA568	4	When the processing of merge(T entity) of EntityManager of the JPA provider starts	B
0xA569	5	When the processing of merge(T entity) of EntityManager of the JPA provider ends	B
0xA56A	4	When the processing of persist(Object entity) of EntityManager of the JPA provider starts	B
0xA56B	5	When the processing of persist(Object entity) of EntityManager of the JPA provider ends	B
0xA56C	4	When the processing of refresh(Object entity) of EntityManager of the JPA provider starts	B
0xA56D	5	When the processing of refresh(Object entity) of EntityManager of the JPA provider ends	B

Event ID	No. in the figure#	Trace acquisition points	Level
0xA56E	4	When the processing of remove(Object entity) of EntityManager of the JPA provider starts	B
0xA56F	5	When the processing of remove(Object entity) of EntityManager of the JPA provider ends	B
0xA570	4	When the processing of clear() of EntityManager of the JPA provider starts	B
0xA571	5	When the processing of clear() of EntityManager of the JPA provider ends	B
0xA572	4	When the processing of flush() of EntityManager of the JPA provider starts	B
0xA573	5	When the processing of flush() of EntityManager of the JPA provider ends	B
0xA574	4	When the processing of createQuery(String qlString) of EntityManager of the JPA provider starts	B
0xA575	5	When the processing of createQuery(String qlString) of EntityManager of the JPA provider ends	B
0xA576	4	When the processing of createNamedQuery(String name) of EntityManager of the JPA provider starts	B
0xA577	5	When the processing of createNamedQuery(String name) of EntityManager of the JPA provider ends	B
0xA578	4	When the processing of createNativeQuery(String sqlString) of EntityManager of the JPA provider starts	B
0xA579	5	When the processing of createNativeQuery(String sqlString) of EntityManager of the JPA provider ends	B
0xA57A	4	When the processing of createNativeQuery(String sqlString, Class resultClass) of EntityManager of the JPA provider starts	B
0xA57B	5	When the processing of createNativeQuery(String sqlString, Class resultClass) of EntityManager of the JPA provider ends	B
0xA57C	4	When the processing of createNativeQuery(String sqlString, String resultSetMapping) of EntityManager of the JPA provider starts	B
0xA57D	5	When the processing of createNativeQuery(String sqlString, String resultSetMapping) of EntityManager of the JPA provider ends	B
0xA57E	4	When the processing of setFlushMode(FlushModeType flushMode) of EntityManager of the JPA provider starts	B
0xA57F	5	When the processing of setFlushMode(FlushModeType flushMode) of EntityManager of the JPA provider ends	B
0xA580	4	When the processing of getFlushMode() of EntityManager of the JPA provider starts	B
0xA581	5	When the processing of getFlushMode() of EntityManager of the JPA provider ends	B
0xA582	4	When the processing of joinTransaction() of EntityManager of the JPA provider starts	B
0xA583	5	When the processing of joinTransaction() of EntityManager of the JPA provider ends	B
0xA584	4	When the processing of isOpen() of EntityManager of the JPA provider starts	B
0xA585	5	When the processing of isOpen() of EntityManager of the JPA provider ends	B
0xA586	4	When the processing of executeUpdate() of Query of the JPA provider starts	B

Event ID	No. in the figure#	Trace acquisition points	Level
0xA587	5	When the processing of executeUpdate() of Query of the JPA provider ends	B
0xA588	4	When the processing of getResultList() of Query of the JPA provider starts	B
0xA589	5	When the processing of getResultList() of Query of the JPA provider ends	B
0xA58A	4	When the processing of getSingleResult() of Query of the JPA provider starts	B
0xA58B	5	When the processing of getSingleResult() of Query of the JPA provider ends	B
0xA58C	4	When the processing of setFlushMode(FlushModeType flushMode) of Query of the JPA provider starts	B
0xA58D	5	When the processing of setFlushMode(FlushModeType flushMode) of Query of the JPA provider ends	B
0xA58E	4	When the processing of setFirstResult(int startPosition) of Query of the JPA provider starts	B
0xA58F	5	When the processing of setFirstResult(int startPosition) of Query of the JPA provider ends	B
0xA590	4	When the processing of setMaxResults(int maxResult) of Query of the JPA provider starts	B
0xA591	5	When the processing of setMaxResults(int maxResult) of Query of the JPA provider ends	B
0xA592	4	When the processing of setHint(String hintName, Object value) of Query of the JPA provider starts	B
0xA593	5	When the processing of setHint(String hintName, Object value) of Query of the JPA provider ends	B
0xA594	4	When the processing of setParameter(int position, Calendar value, TemporalType temporalType) of Query of the JPA provider starts	B
0xA595	5	When the processing of setParameter(int position, Calendar value, TemporalType temporalType) of Query of the JPA provider ends	B
0xA596	4	When the processing of setParameter(int position, Date value, TemporalType temporalType) of Query of the JPA provider starts	B
0xA597	5	When the processing of setParameter(int position, Date value, TemporalType temporalType) of Query of the JPA provider ends	B
0xA598	4	When the processing of setParameter(int position, Object value) of Query of the JPA provider starts	B
0xA599	5	When the processing of setParameter(int position, Object value) of Query of the JPA provider ends	B
0xA59A	4	When the processing of setParameter(String name, Calendar value, TemporalType temporalType) of Query of the JPA provider starts	B
0xA59B	5	When the processing of setParameter(String name, Calendar value, TemporalType temporalType) of Query of the JPA provider ends	B
0xA59C	4	When the processing of setParameter(String name, Date value, TemporalType temporalType) of Query of the JPA provider starts	B
0xA59D	5	When the processing of setParameter(String name, Date value, TemporalType temporalType) of Query of the JPA provider ends	B
0xA59E	4	When the processing of setParameter(String name, Object value) of Query of the JPA provider starts	B

Event ID	No. in the figure#	Trace acquisition points	Level
0xA59F	5	When the processing of setParameter(String name, Object value) of Query of the JPA provider ends	B
0xA5AC	1	At the time of start of the processing for generating the persistent context, when the persistent context of the extended scope is used	A
0xA5AD	2	At the time of termination of the processing for generating the persistent context, when the persistent context of the extended scope is used	A
0xA5AE	7	When the processing of EntityManager.close() starts while the persistent context of the extended scope is being used	A
0xA5AF	8	When the processing of EntityManager.close() ends while the persistent context of the extended scope is being used	A

Legend:

A: Standard

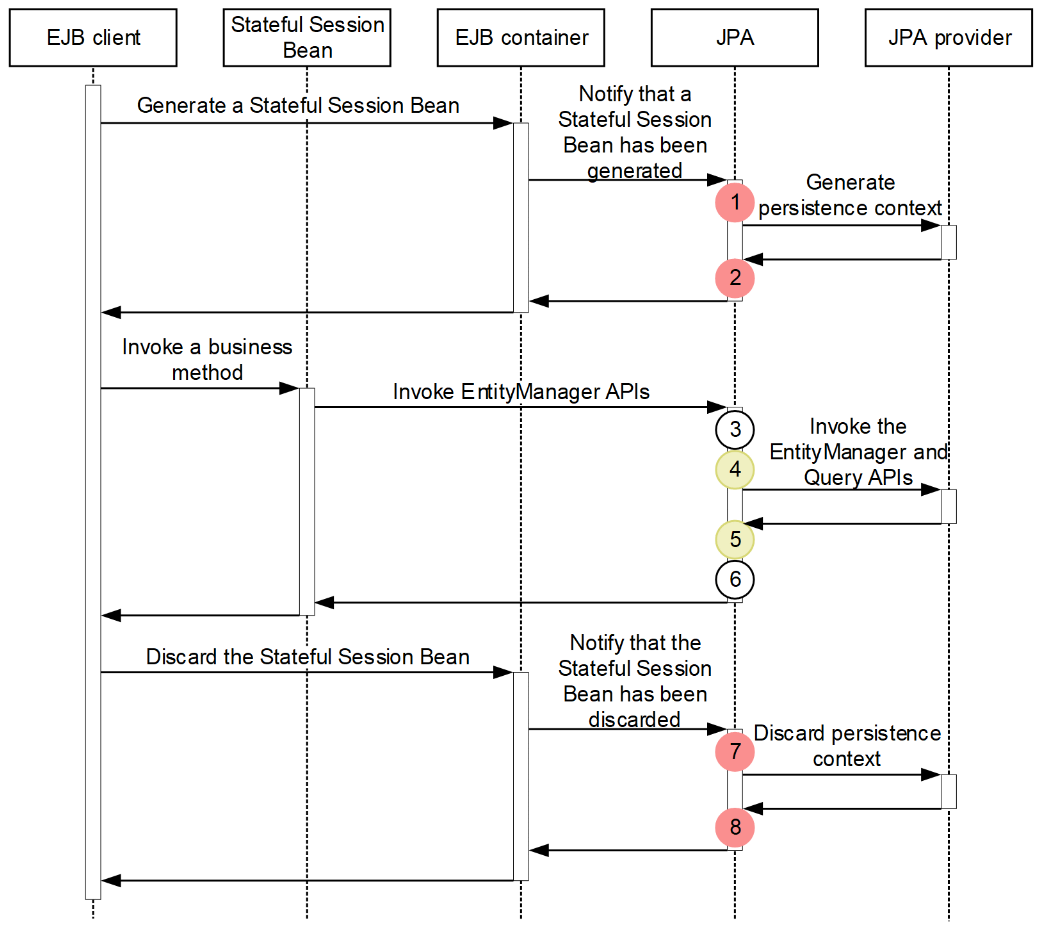
B: Advanced

#

Corresponds to the numbers in *Figure 15-20*.

The following figure shows the trace collection points.

Figure 15–20: Trace collection points when an extended persistent context is used


EJB client
Stateful Session Bean
EJB container
JPA
JPA provider
Generate a Stateful Session Bean
Notify that a Stateful Session Bean has been generated
Generate persistence context
1
2
Invoke a business method
Invoke EntityManager APIs
3
4
Invoke the EntityManager and Query APIs
5
6
Discard the Stateful Session Bean
Notify that the Stateful Session Bean has been discarded
Discard persistence context
7
8

Legend: 🔴 : Indicates a trace collection point. The PRF trace collection level is "Standard".

🟡 : Indicates a trace collection point. The PRF trace collection level is "Advanced".

○ : Indicates a trace collection point.
The PRF trace collection level differs depending on the processing.

(2) Trace information that can be collected

(a) When the persistent context of the transaction scope is used in the transaction

The following table describes the trace information that can be collected when the persistent context of the transaction scope is used in the transaction.

Table 15–34: Trace information that can be collected when the persistent context of the transaction scope is used in the transaction

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xA508	A	entity class name	--	--
	0xA50A	A	entity class name	--	--
	0xA50C	A	entity class name	--	--
	0xA50E	A	entity class name	lockMode value	--
	0xA510	A	entity class name	--	--
	0xA512	A	entity class name	--	--
	0xA514	A	entity class name	--	--
	0xA516	A	entity class name	--	--
	0xA518	A	--	--	--
	0xA51A	A	--	--	--
	0xA51C	A	--	--	--
	0xA51E	A	name	--	--
	0xA520	A	--	--	--
	0xA522	A	resultClass class name	--	--
	0xA524	A	resultSetMapping	--	--
	0xA526	A	flushMode value	--	--
	0xA528	A	--	--	--
	0xA52A	A	--	--	--
	0xA52C	A	--	--	--
	0xA52E	A	--	--	--
	0xA530	A	--	--	--
	0xA532	A	--	--	--
	0xA540	A	--	--	--
	0xA542	A	--	--	--
	0xA544	A	--	--	--
	0xA546	A	flushMode value	--	--
	0xA548	B	startPosition value	--	--
	0xA54A	B	maxResult value	--	--

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0xA54C	B	hintName	value class name	--
	0xA54E	B	position value	--	--
	0xA550	B	position value	--	--
	0xA552	B	position value	--	--
	0xA554	B	name value	--	--
	0xA556	B	name value	--	--
	0xA558	B	name value	--	--
2	0xA5A0	A	--	--	--
3	0xA5A1	A	--	--	#2
4	0xA560	B	entity class name	--	--
	0xA562	B	entity class name	--	--
	0xA564	B	entity class name	--	--
	0xA566	B	entity class name	lockMode value	--
	0xA568	B	entity class name	--	--
	0xA56A	B	entity class name	--	--
	0xA56C	B	entity class name	--	--
	0xA56E	B	entity class name	--	--
	0xA570	B	--	--	--
	0xA572	B	--	--	--
	0xA574	B	--	--	--
	0xA576	B	name	--	--
	0xA578	B	--	--	--
	0xA57A	B	resultClass class name	--	--
	0xA57C	B	resultSetMapping	--	--
	0xA57E	B	flushMode value	--	--
	0xA580	B	--	--	--
	0xA582	B	--	--	--
	0xA584	B	--	--	--
	0xA586	B	--	--	--
	0xA588	B	--	--	--
	0xA58A	B	--	--	--
	0xA58C	B	flushMode value	--	--
	0xA58E	B	startPosition value	--	--
	0xA590	B	maxResult value	--	--

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0xA592	B	hintName	value class name	--
	0xA594	B	position value	--	--
	0xA596	B	position value	--	--
	0xA598	B	position value	--	--
	0xA59A	B	name value	--	--
	0xA59C	B	name value	--	--
	0xA59E	B	name value	--	--
5	0xA561	B	--	--	#2
	0xA563	B	--	--	#2
	0xA565	B	--	--	#2
	0xA567	B	--	--	#2
	0xA569	B	--	--	#2
	0xA56B	B	--	--	#2
	0xA56D	B	--	--	#2
	0xA56F	B	--	--	#2
	0xA571	B	--	--	#2
	0xA573	B	--	--	#2
	0xA575	B	--	--	#2
	0xA577	B	--	--	#2
	0xA579	B	--	--	#2
	0xA57B	B	--	--	#2
	0xA57D	B	--	--	#2
	0xA57F	B	--	--	#2
	0xA581	B	--	--	#2
	0xA583	B	--	--	#2
	0xA585	B	--	--	#2
	0xA587	B	--	--	#2
	0xA589	B	--	--	#2
	0xA58B	B	--	--	#2
	0xA58D	B	--	--	#2
	0xA58F	B	--	--	#2
	0xA591	B	--	--	#2
	0xA593	B	--	--	#2
	0xA595	B	--	--	#2

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0xA597	B	--	--	#2
	0xA599	B	--	--	#2
	0xA59B	B	--	--	#2
	0xA59D	B	--	--	#2
	0xA59F	B	--	--	#2
6	0xA509	A	--	--	#2
	0xA50B	A	--	--	#2
	0xA50D	A	--	--	#2
	0xA50F	A	--	--	#2
	0xA511	A	--	--	#2
	0xA513	A	--	--	#2
	0xA515	A	--	--	#2
	0xA517	A	--	--	#2
	0xA519	A	--	--	#2
	0xA51B	A	--	--	#2
	0xA51D	A	--	--	#2
	0xA51F	A	--	--	#2
	0xA521	A	--	--	#2
	0xA523	A	--	--	#2
	0xA525	A	--	--	#2
	0xA527	A	--	--	#2
	0xA529	A	--	--	#2
	0xA52B	A	--	--	#2
	0xA52D	A	--	--	#2
	0xA52F	A	--	--	#2
	0xA531	A	--	--	#2
	0xA533	A	--	--	#2
	0xA541	A	--	--	#2
	0xA543	A	--	--	#2
	0xA545	A	--	--	#2
	0xA547	A	--	--	#2
	0xA549	B	--	--	#2
	0xA54B	B	--	--	#2
	0xA54D	B	--	--	#2

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0xA54F	B	--	--	#2
	0xA551	B	--	--	#2
	0xA553	B	--	--	#2
	0xA555	B	--	--	#2
	0xA557	B	--	--	#2
	0xA559	B	--	--	#2
7	0xA5A2	A	--	--	--
8	0xA5A3	A	--	--	#2

Legend:

A: Standard

B: Advanced

--: Not applicable

#1

Corresponds to the numbers in *Figure 15-17*.

#2

When the processing is performed normally, the entrance time is displayed.

When an exception occurs, the entrance time and exception are displayed.

(b) When the entity manager related to the persistent context of the transaction scope is used outside a transaction

The following table describes the trace information that can be collected when the entity manager related to the persistent context of the transaction scope is used outside a transaction.

Table 15–35: Trace information that can be collected when the entity manager related to the persistent context of the transaction scope is used outside a transaction

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xA508	A	entity class name	--	--
	0xA50A	A	entity class name	--	--
	0xA50C	A	entity class name	--	--
	0xA50E	A	entity class name	lockMode value	--
	0xA510	A	entity class name	--	--
	0xA512	A	entity class name	--	--
	0xA514	A	entity class name	--	--
	0xA516	A	entity class name	--	--
	0xA518	A	--	--	--
	0xA51A	A	--	--	--
	0xA526	A	flushMode value	--	--

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0xA528	A	--	--	--
	0xA52A	A	--	--	--
	0xA52C	A	--	--	--
	0xA52E	A	--	--	--
	0xA530	A	--	--	--
	0xA532	A	--	--	--
2	0xA5A4	A	--	--	--
3	0xA5A5	A	--	--	#2
4	0xA560	B	entity class name	--	--
	0xA562	B	entity class name	--	--
	0xA564	B	entity class name	--	--
	0xA566	B	entity class name	lockMode value	--
	0xA568	B	entity class name	--	--
	0xA56A	B	entity class name	--	--
	0xA56C	B	entity class name	--	--
	0xA56E	B	entity class name	--	--
	0xA570	B	--	--	--
	0xA572	B	--	--	--
	0xA57E	B	flushMode value	--	--
	0xA580	B	--	--	--
	0xA582	B	--	--	--
	0xA584	B	--	--	--
5	0xA561	B	--	--	#2
	0xA563	B	--	--	#2
	0xA565	B	--	--	#2
	0xA567	B	--	--	#2
	0xA569	B	--	--	#2
	0xA56B	B	--	--	#2
	0xA56D	B	--	--	#2
	0xA56F	B	--	--	#2
	0xA571	B	--	--	#2
	0xA573	B	--	--	#2
	0xA57F	B	--	--	#2
	0xA581	B	--	--	#2

No. in the figure#1	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0xA583	B	--	--	#2
	0xA585	B	--	--	#2
6	0xA5A6	A	--	--	--
7	0xA5A7	A	--	--	#2
8	0xA509	A	--	--	#2
	0xA50B	A	--	--	#2
	0xA50D	A	--	--	#2
	0xA50F	A	--	--	#2
	0xA511	A	--	--	#2
	0xA513	A	--	--	#2
	0xA515	A	--	--	#2
	0xA517	A	--	--	#2
	0xA519	A	--	--	#2
	0xA51B	A	--	--	#2
	0xA527	A	--	--	#2
	0xA529	A	--	--	#2
	0xA52B	A	--	--	#2
	0xA52D	A	--	--	#2
	0xA52F	A	--	--	#2
	0xA531	A	--	--	#2
	0xA533	A	--	--	#2

Legend:

A: Standard

B: Advanced

--: Not applicable

#1

Corresponds to the numbers in *Figure 15-18*.

#2

When the processing is performed normally, the entrance time is displayed.

When an exception occurs, the entrance time and exception are displayed.

(c) When a Query generated outside a transaction is used outside the transaction

The following table describes the trace information that can be collected when a Query generated outside a transaction is used outside the transaction.

Table 15–36: Trace information that can be collected when a Query generated outside a transaction is used outside the transaction

No. in the figure#1	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xA51C	A	--	--	--
	0xA51E	A	name	--	--
	0xA520	A	--	--	--
	0xA522	A	resultClass class name	--	--
	0xA524	A	resultSetMapping	--	--
1, 11	0xA540	A	--	--	--
	0xA542	A	--	--	--
	0xA544	A	--	--	--
1, 7	0xA546	A	flushMode value	--	--
	0xA548	B	startPosition value	--	--
	0xA54A	B	maxResult value	--	--
	0xA54C	B	hintName	value class name	--
	0xA54E	B	position value	--	--
	0xA550	B	position value	--	--
	0xA552	B	position value	--	--
	0xA554	B	name value	--	--
	0xA556	B	name value	--	--
	0xA558	B	name value	--	--
2	0xA5A8	A	--	--	--
3	0xA5A9	A	--	--	#2
4	0xA574	B	--	--	--
	0xA576	B	name	--	--
	0xA578	B	--	--	--
	0xA57A	B	resultClass class name	--	--
	0xA57C	B	resultSetMapping	--	--
4, 12	0xA586	B	--	--	--
	0xA588	B	--	--	--
	0xA58A	B	--	--	--
4, 8	0xA58C	B	flushMode value	--	--
	0xA58E	B	startPosition value	--	--
	0xA590	B	maxResult value	--	--
	0xA592	B	hintName	value class name	--

No. in the figure#1	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0xA594	B	position value	--	--
	0xA596	B	position value	--	--
	0xA598	B	position value	--	--
	0xA59A	B	name value	--	--
	0xA59C	B	name value	--	--
	0xA59E	B	name value	--	--
5	0xA575	B	--	--	#2
	0xA577	B	--	--	#2
	0xA579	B	--	--	#2
	0xA57B	B	--	--	#2
	0xA57D	B	--	--	#2
5, 13	0xA587	B	--	--	#2
	0xA589	B	--	--	#2
	0xA58B	B	--	--	#2
5, 9	0xA58D	B	--	--	#2
	0xA58F	B	--	--	#2
	0xA591	B	--	--	#2
	0xA593	B	--	--	#2
	0xA595	B	--	--	#2
	0xA597	B	--	--	#2
	0xA599	B	--	--	#2
	0xA59B	B	--	--	#2
	0xA59D	B	--	--	#2
	0xA59F	B	--	--	#2
6	0xA51D	A	--	--	#2
	0xA51F	A	--	--	#2
	0xA521	A	--	--	#2
	0xA523	A	--	--	#2
	0xA525	A	--	--	#2
6, 16	0xA541	A	--	--	#2
	0xA543	A	--	--	#2
	0xA545	A	--	--	#2
6, 10	0xA547	A	--	--	#2
	0xA549	B	--	--	#2

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0xA54B	B	--	--	#2
	0xA54D	B	--	--	#2
	0xA54F	B	--	--	#2
	0xA551	B	--	--	#2
	0xA553	B	--	--	#2
	0xA555	B	--	--	#2
	0xA557	B	--	--	#2
	0xA559	B	--	--	#2
14	0xA5AA	A	--	--	--
15	0xA5AB	A	--	--	#2

Legend:

A: Standard

B: Advanced

--: Not applicable

#1

Corresponds to the numbers in *Figure 15-19*.

#2

When the processing is performed normally, the entrance time is displayed.

When an exception occurs, the entrance time and exception are displayed.

(d) When an extended persistent context is used

The following table describes the trace information that can be collected when an extended persistent context is used.

Table 15–37: Trace information that can be collected when an extended persistent context is used

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xA5AC	A	--	--	--
2	0xA5AD	A	--	--	#2
3	0xA508	A	entity class name	--	--
	0xA50A	A	entity class name	--	--
	0xA50C	A	entity class name	--	--
	0xA50E	A	entity class name	lockMode value	--
	0xA510	A	entity class name	--	--
	0xA512	A	entity class name	--	--
	0xA514	A	entity class name	--	--
	0xA516	A	entity class name	--	--
	0xA518	A	--	--	--
	0xA51A	A	--	--	--

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0xA51C	A	--	--	--
	0xA51E	A	name	--	--
	0xA520	A	--	--	--
	0xA522	A	resultClass class name	--	--
	0xA524	A	resultSetMapping	--	--
	0xA526	A	flushMode value	--	--
	0xA528	A	--	--	--
	0xA52A	A	--	--	--
	0xA52C	A	--	--	--
	0xA52E	A	--	--	--
	0xA530	A	--	--	--
	0xA532	A	--	--	--
	0xA540	A	--	--	--
	0xA542	A	--	--	--
	0xA544	A	--	--	--
	0xA546	A	flushMode value	--	--
	0xA548	B	startPosition value	--	--
	0xA54A	B	maxResult value	--	--
	0xA54C	B	hintName	value class name	--
	0xA54E	B	position value	--	--
	0xA550	B	position value	--	--
	0xA552	B	position value	--	--
	0xA554	B	name value	--	--
	0xA556	B	name value	--	--
	0xA558	B	name value	--	--
4	0xA560	B	entity class name	--	--
	0xA562	B	entity class name	--	--
	0xA564	B	entity class name	--	--
	0xA566	B	entity class name	lockMode value	--
	0xA568	B	entity class name	--	--
	0xA56A	B	entity class name	--	--
	0xA56C	B	entity class name	--	--
	0xA56E	B	entity class name	--	--
	0xA570	B	--	--	--

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0xA572	B	--	--	--
	0xA574	B	--	--	--
	0xA576	B	name	--	--
	0xA578	B	--	--	--
	0xA57A	B	resultClass class name	--	--
	0xA57C	B	resultSetMapping	--	--
	0xA57E	B	flushMode value	--	--
	0xA580	B	--	--	--
	0xA582	B	--	--	--
	0xA584	B	--	--	--
	0xA586	B	--	--	--
	0xA588	B	--	--	--
	0xA58A	B	--	--	--
	0xA58C	B	flushMode value	--	--
	0xA58E	B	startPosition value	--	--
	0xA590	B	maxResult value	--	--
	0xA592	B	hintName	value class name	--
	0xA594	B	position value	--	--
	0xA596	B	position value	--	--
	0xA598	B	position value	--	--
	0xA59A	B	name value	--	--
	0xA59C	B	name value	--	--
	0xA59E	B	name value	--	--
5	0xA561	B	--	--	#2
	0xA563	B	--	--	#2
	0xA565	B	--	--	#2
	0xA567	B	--	--	#2
	0xA569	B	--	--	#2
	0xA56B	B	--	--	#2
	0xA56D	B	--	--	#2
	0xA56F	B	--	--	#2
	0xA571	B	--	--	#2
	0xA573	B	--	--	#2
	0xA575	B	--	--	#2

No. in the figure#1	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0xA577	B	--	--	#2
	0xA579	B	--	--	#2
	0xA57B	B	--	--	#2
	0xA57D	B	--	--	#2
	0xA57F	B	--	--	#2
	0xA581	B	--	--	#2
	0xA583	B	--	--	#2
	0xA585	B	--	--	#2
	0xA587	B	--	--	#2
	0xA589	B	--	--	#2
	0xA58B	B	--	--	#2
	0xA58D	B	--	--	#2
	0xA58F	B	--	--	#2
	0xA591	B	--	--	#2
	0xA593	B	--	--	#2
	0xA595	B	--	--	#2
	0xA597	B	--	--	#2
	0xA599	B	--	--	#2
	0xA59B	B	--	--	#2
	0xA59D	B	--	--	#2
	0xA59F	B	--	--	#2
6	0xA509	A	--	--	#2
	0xA50B	A	--	--	#2
	0xA50D	A	--	--	#2
	0xA50F	A	--	--	#2
	0xA511	A	--	--	#2
	0xA513	A	--	--	#2
	0xA515	A	--	--	#2
	0xA517	A	--	--	#2
	0xA519	A	--	--	#2
	0xA51B	A	--	--	#2
	0xA51D	A	--	--	#2
	0xA51F	A	--	--	#2
	0xA521	A	--	--	#2

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0xA523	A	--	--	#2
	0xA525	A	--	--	#2
	0xA527	A	--	--	#2
	0xA529	A	--	--	#2
	0xA52B	A	--	--	#2
	0xA52D	A	--	--	#2
	0xA52F	A	--	--	#2
	0xA531	A	--	--	#2
	0xA533	A	--	--	#2
	0xA541	A	--	--	#2
	0xA543	A	--	--	#2
	0xA545	A	--	--	#2
	0xA547	A	--	--	#2
	0xA549	B	--	--	#2
	0xA54B	B	--	--	#2
	0xA54D	B	--	--	#2
	0xA54F	B	--	--	#2
	0xA551	B	--	--	#2
	0xA553	B	--	--	#2
	0xA555	B	--	--	#2
	0xA557	B	--	--	#2
	0xA559	B	--	--	#2
7	0xA5AE	A	--	--	--
8	0xA5AF	A	--	--	#2

Legend:

A: Standard

B: Advanced

--: Not applicable

#1

Corresponds to the numbers in *Figure 15-20*.

#2

When the processing is performed normally, the entrance time is displayed.

When an exception occurs, the entrance time and exception are displayed.

15.12 Trace collection points of the Cosminexus JPA provider

This section describes the trace collection points of the Cosminexus JPA provider, and also describes the trace information that can be collected.

15.12.1 Trace collection points and trace information that can be collected during the acquisition or release processing of EntityManagerFactory

(1) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–38: Details of trace collection points during the acquisition or release processing of EntityManagerFactory

Event ID	No. in the figure[#1]	Trace acquisition points	Level
0xA304[#2]	3	When the processing of EntityManagerFactory#close() starts	B
0xA305[#2]	4	When the processing of EntityManagerFactory#close() ends	B
0xA320[#3]	1	When the processing of PersistenceProvider#createEntityManagerFactory(String, Map) starts	A
0xA321[#3]	2	When the processing of PersistenceProvider#createEntityManagerFactory(String, Map) ends	A
0xA322[#3]	1	When the processing of PersistenceProvider#createContainerEntityManagerFactory(PersistenceUnitInfo, Map) starts	A
0xA323[#3]	2	When the processing of PersistenceProvider#createContainerEntityManagerFactory(PersistenceUnitInfo, Map) ends	A

Legend:

A: Standard

B: Advanced

#1

Corresponds to the numbers in *Figure 15-21*.

#2

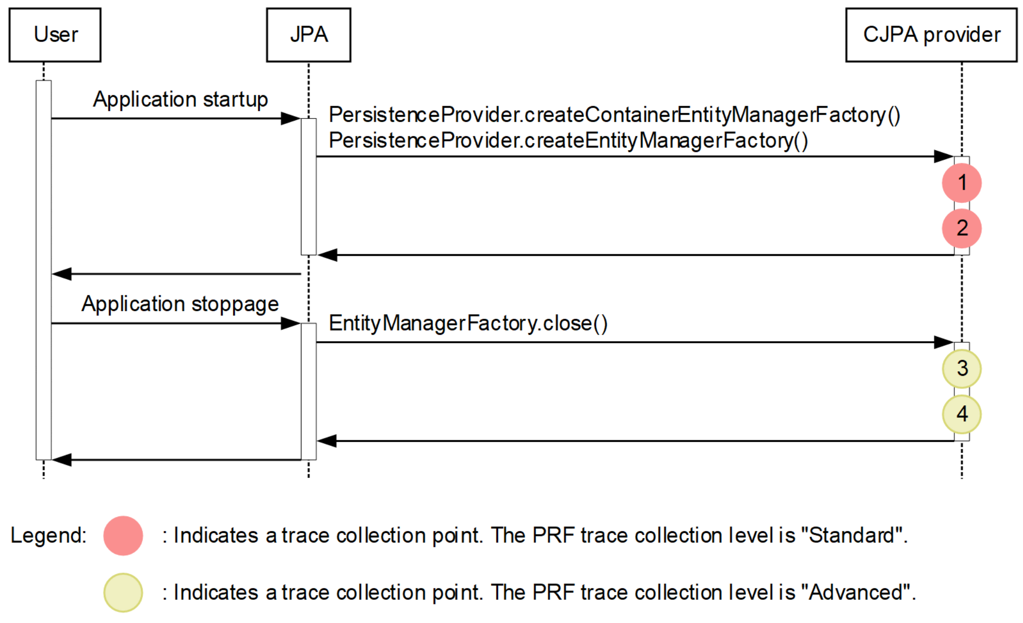
Trace collection point for `javax.persistence.EntityManagerFactory`.

#3

Trace collection point for `javax.persistence.spi.PersistenceProvider`.

The following figure shows the trace collection points.

Figure 15–21: Trace collection points of the acquisition or release processing of EntityManagerFactory


User
JPA
CJPA provider
Application startup
PersistenceProvider.createContainerEntityManagerFactory()
PersistenceProvider.createEntityManagerFactory()
1
2
Application stoppage
EntityManagerFactory.close()
3
4
Legend:
: Indicates a trace collection point. The PRF trace collection level is "Standard".
: Indicates a trace collection point. The PRF trace collection level is "Advanced".

(2) Trace information that can be collected

The following table describes the trace information that can be collected during the acquisition or release processing of EntityManagerFactory.

Table 15–39: Trace information that can be collected during the acquisition or release processing of EntityManagerEntityFactory

No. in the figure#1	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1#2	0xA320#3	A	--	--	--
	0xA322	A	--	--	--
2#2	0xA321#3	A	--	--	#4
	0xA323	A	--	--	#4
3#5	0xA304	B	--	--	--
4#5	0xA305	B	--	--	#4

Legend:

A: Standard

B: Advanced

--: Not applicable

#1

Corresponds to the numbers in *Figure 15-21*.

#2

Trace collection point for `javax.persistence.spi.PersistenceProvider`.

#3

Not output when the Cosminexus JPA provider is used.

#4

When an exception occurs, the exception is displayed.

#5

Trace collection point for `javax.persistence.EntityManagerFactory`.

15.12.2 Trace collection points and trace information that can be collected during the acquisition processing of EntityManager

(1) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–40: Details of trace collection points for javax.persistence.EntityManagerFactory

Event ID	No. in the figure#	Trace acquisition points	Level
0xA300	1	When the processing of EntityManagerFactory#createEntityManager() starts	B
0xA301	2	When the processing of EntityManagerFactory#createEntityManager() ends	B
0xA302	1	When the processing of EntityManagerFactory#createEntityManager(Map) starts	B
0xA303	2	When the processing of EntityManagerFactory#createEntityManager(Map) ends	B

Legend:

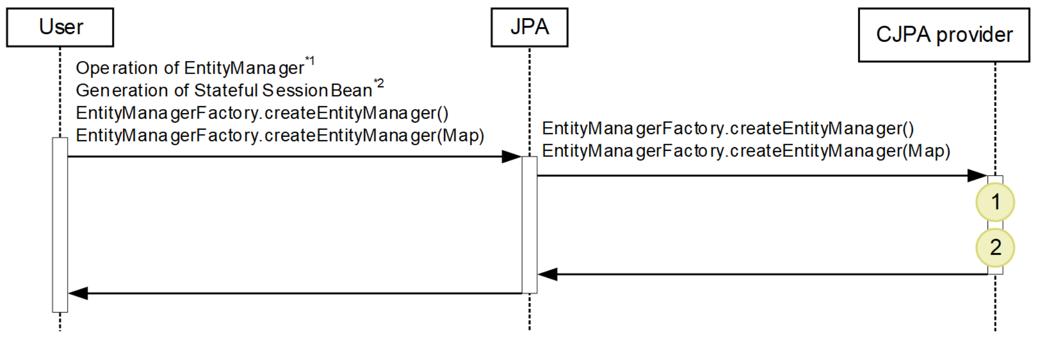
B: Advanced

#

Corresponds to the numbers in *Figure 15-22*.

The following figure shows the trace collection points.

Figure 15–22: Trace collection points for javax.persistence.EntityManagerFactory


User
JPA
CJPA provider
Operation of EntityManager*1
Generation of Stateful Session Bean*2
EntityManagerFactory.createEntityManager()
EntityManagerFactory.createEntityManager(Map)
EntityManagerFactory.createEntityManager()
EntityManagerFactory.createEntityManager(Map)
1
2

Legend: ● : Indicates a trace collection point. The PRF trace collection level is "Advanced".

*1 EntityManager provided by the CJPA provider is wrapped by the EJB container.
As such, in the case of the persistence context of the transaction scope, if there is no reference to EntityManager of the CJPA provider when the operation of EntityManager is performed, EntityManagerFactory.createEntityManager() might not be invoked.
*2 This is carried out when the extended persistence context is used.

(2) Trace information that can be collected

The following table describes the trace information that can be collected during the acquisition processing of EntityManager.

Table 15–41: Trace information that can be collected for javax.persistence.EntityManagerFactory

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xA300	B	--	--	--
	0xA302	B	--	--	--
2	0xA301	B	--	--	#2
	0xA303	B	--	--	#2

Legend:

B: Advanced

--: Not applicable

#1

Corresponds to the numbers in *Figure 15-22*.

#2

When an exception occurs, the exception is displayed.

15.12.3 Trace collection points and trace information that can be collected during the operation of EntityManager

(1) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–42: Details of trace collection points for javax.persistence.EntityManager

Event ID	No. in the figure[#1]	Trace acquisition points	Level
0xA340	1	When the processing of EntityManager#persist(Object) starts	B
0xA341	2	When the processing of EntityManager#persist(Object) ends	B
0xA342	1	When the processing of EntityManager#merge(T) starts	B
0xA343	2	When the processing of EntityManager#merge(T) ends	B
0xA344	1	When the processing of EntityManager#remove(Object) starts	B
0xA345	2	When the processing of EntityManager#remove(Object) ends	B
0xA346	1	When the processing of EntityManager#find(Class<T>, Object) starts	B
0xA347	2	When the processing of EntityManager#find(Class<T>, Object) ends	B
0xA348	1	When the processing of EntityManager#getReference(Class<T>, Object) starts	B
0xA349	2	When the processing of EntityManager#getReference(Class<T>, Object) ends	B
0xA34A	1	When the processing of EntityManager#flush() starts	B
0xA34B	2	When the processing of EntityManager#flush() ends	B
0xA34C	1	When the processing of EntityManager#lock(Object, LockModeType) starts	B

Event ID	No. in the figure[#1]	Trace acquisition points	Level
0xA34D	2	When the processing of EntityManager#lock(Object, LockModeType) ends	B
0xA34E	1	When the processing of EntityManager#refresh(Object) starts	B
0xA34F	2	When the processing of EntityManager#refresh(Object) ends	B
0xA350	1	When the processing of EntityManager#clear() starts	B
0xA351	2	When the processing of EntityManager#clear() ends	B
0xA352	1	When the processing of EntityManager#contains(Object) starts	B
0xA353	2	When the processing of EntityManager#contains(Object) ends	B
0xA35E	1	When the processing of EntityManager#joinTransaction() starts	B
0xA35F	2	When the processing of EntityManager#joinTransaction() ends	B

Legend:

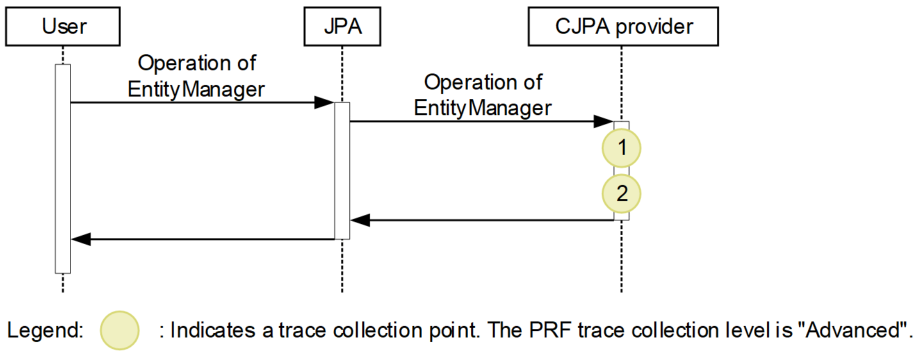
B: Advanced

#1

Corresponds to the numbers in *Figure 15-23*.

The following figure shows the trace collection points.

Figure 15–23: Trace collection points for javax.persistence.EntityManager


User
JPA
CJPA provider
Operation of EntityManager
Operation of EntityManager
1
2
Legend:
: Indicates a trace collection point. The PRF trace collection level is "Advanced".

(2) Trace information that can be collected

The following table describes the trace information that can be collected during the operation of EntityManager.

Table 15–43: Trace information that can be collected for javax.persistence.EntityManager

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xA340	B	--	--	--
	0xA342	B	--	--	--
	0xA344	B	--	--	--
	0xA346	B	--	--	--
	0xA348	B	--	--	--
	0xA34A	B	--	--	--
	0xA34C	B	--	--	--

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0xA34E	B	--	--	--
	0xA350	B	--	--	--
	0xA352	B	--	--	--
	0xA35E	B	--	--	--
2	0xA341	B	--	--	#2
	0xA343	B	--	--	#2
	0xA345	B	--	--	#2
	0xA347	B	--	--	#2
	0xA349	B	--	--	#2
	0xA34B	B	--	--	#2
	0xA34D	B	--	--	#2
	0xA34F	B	--	--	#2
	0xA351	B	--	--	#2
	0xA353	B	--	--	#2
	0xA35F	B	--	--	#2

Legend:

B: Advanced

--: Not applicable

#1

Corresponds to the numbers in *Figure 15-23*.

#2

When an exception occurs, the exception is displayed.

15.12.4 Trace collection points and trace information that can be collected during the release processing of EntityManager

(1) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–44: Details of trace collection points for javax.persistence.EntityManager

Event ID	No. in the figure[#]	Trace acquisition points	Level
0xA360	1	When the processing of EntityManager#close() starts	B
0xA361	2	When the processing of EntityManager#close() ends	B

Legend:

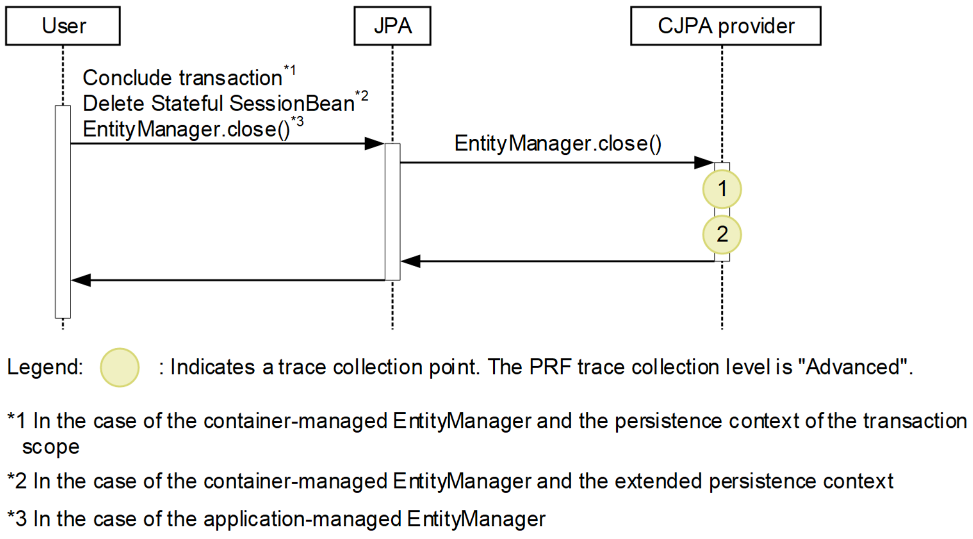
B: Advanced

#

Corresponds to the numbers in *Figure 15-24*.

The following figure shows the trace collection points.

Figure 15–24: Trace collection points for javax.persistence.EntityManager


User
JPA
CJPA provider
Conclude transaction*1
Delete Stateful SessionBean*2
EntityManager.close()*3
EntityManager.close()
1
2
Legend: : Indicates a trace collection point. The PRF trace collection level is "Advanced".
*1 In the case of the container-managed EntityManager and the persistence context of the transaction scope
*2 In the case of the container-managed EntityManager and the extended persistence context
*3 In the case of the application-managed EntityManager

(2) Trace information that can be collected

The following table describes the trace information that can be collected during the release processing of EntityManager.

Table 15–45: Trace information that can be collected for javax.persistence.EntityManager

No. in the figure#1	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xA360	B	--	--	--
2	0xA361	B	--	--	#2

Legend:

B: Advanced

--: Not applicable

#1

Corresponds to the numbers in *Figure 15-24*.

#2

When an exception occurs, the exception is displayed.

15.12.5 Trace collection points and trace information that can be collected during the operation of Query

(1) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–46: Details of trace collection points of the Query operation

Event ID	No. in the figure[#1]	Trace acquisition points	Level
0xA354[#2]	1	When the processing of EntityManager#createQuery(String) starts	B
0xA355[#2]	2	When the processing of EntityManager#createQuery(String) ends	B
0xA356[#2]	1	When the processing of EntityManager#createNamedQuery(String) starts	B
0xA357[#2]	2	When the processing of EntityManager#createNamedQuery(String) ends	B
0xA358[#2]	1	When the processing of EntityManager#createNativeQuery(String) starts	B
0xA359[#2]	2	When the processing of EntityManager#createNativeQuery(String) ends	B
0xA35A[#2]	1	When the processing of EntityManager#createNativeQuery(String, Class) starts	B
0xA35B[#2]	2	When the processing of EntityManager#createNativeQuery(String, Class) ends	B
0xA35C[#2]	1	When the processing of EntityManager#createNativeQuery(String, String) starts	B
0xA35D[#2]	2	When the processing of EntityManager#createNativeQuery(String, String) ends	B
0xA370[#3]	5	When the processing of Query#getResultList() starts	B
0xA371[#3]	6	When the processing of Query#getResultList() ends	B
0xA372[#3]	5	When the processing of Query#getSingleResult() starts	B
0xA373[#3]	6	When the processing of Query#getSingleResult() ends	B
0xA374[#3]	5	When the processing of Query#executeUpdate() starts	B
0xA375[#3]	6	When the processing of Query#executeUpdate() ends	B
0xA376[#3]	3	When the processing of Query#setParameter(String, Object) starts	B
0xA377[#3]	4	When the processing of Query#setParameter(String, Object) ends	B
0xA378[#3]	3	When the processing of Query#setParameter(String, Date, TemporalType) starts	B
0xA379[#3]	4	When the processing of Query#setParameter(String, Date, TemporalType) ends	B
0xA37A[#3]	3	When the processing of Query#setParameter(String, Calendar, TemporalType) starts	B
0xA37B[#3]	4	When the processing of Query#setParameter(String, Calendar, TemporalType) ends	B
0xA37C[#3]	3	When the processing of Query#setParameter(int, Object) starts	B
0xA37D[#3]	4	When the processing of Query#setParameter(int, Object) ends	B
0xA37E[#3]	3	When the processing of Query#setParameter(int, Date, TemporalType) starts	B
0xA37F[#3]	4	When the processing of Query#setParameter(int, Date, TemporalType) ends	B
0xA380[#3]	3	When the processing of Query#setParameter(int, Calendar, TemporalType) starts	B

Event ID	No. in the figure[#1]	Trace acquisition points	Level
0xA381[#3]	4	When the processing of Query#setParameter(int, Calendar, TemporalType) ends	B

Legend:

B: Advanced

#1

Corresponds to the numbers in *Figure 15-25*.

#2

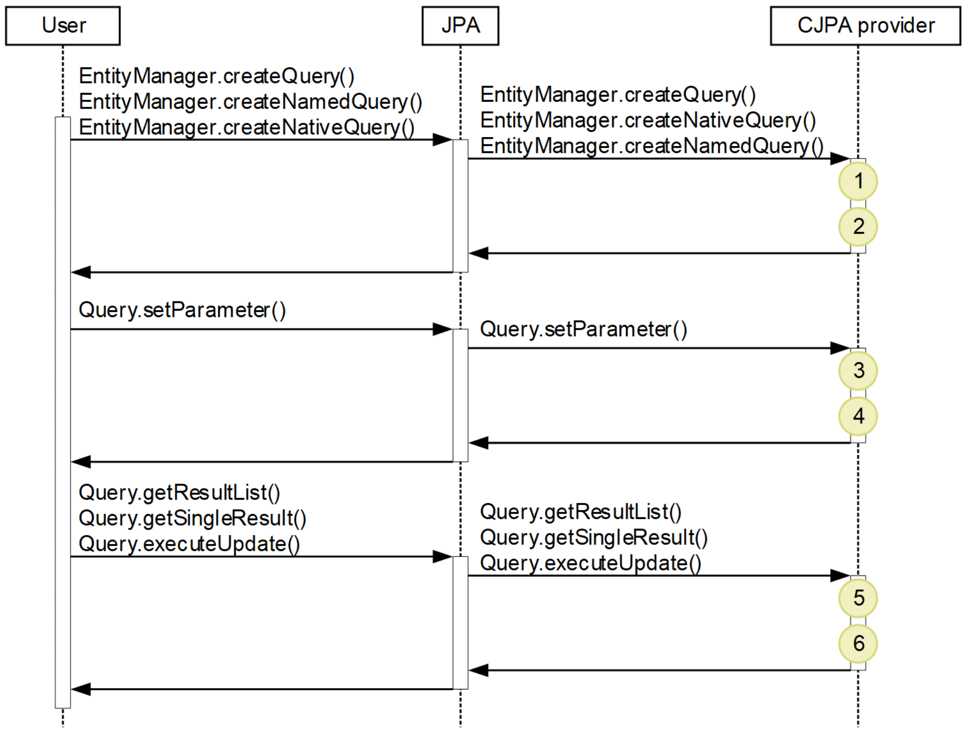
Trace collection point for `javax.persistence.EntityManager`.

#3

Trace collection point for `javax.persistence.Query`.

The following figure shows the trace collection points.

Figure 15–25: Trace collection points of the Query operation



Legend: ● : Indicates a trace collection point. The PRF trace collection level is "Advanced".

(2) Trace information that can be collected

The following table describes the trace information that can be collected during the Query operation.

Table 15–47: Trace information that can be collected during the Query operation

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1[#2]	0xA354	B	--	--	--
	0xA356	B	--	--	--

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0xA358	B	--	--	--
	0xA35A	B	--	--	--
	0xA35C	B	--	--	--
2[#2]	0xA355	B	--	--	#3
	0xA357	B	--	--	#3
	0xA359	B	--	--	#3
	0xA35B	B	--	--	#3
	0xA35D	B	--	--	#3
3[#4]	0xA376	B	--	--	--
	0xA378	B	--	--	--
	0xA37A	B	--	--	--
	0xA37C	B	--	--	--
	0xA37E	B	--	--	--
	0xA380	B	--	--	--
4[#4]	0xA377	B	--	--	#3
	0xA379	B	--	--	#3
	0xA37B	B	--	--	#3
	0xA37D	B	--	--	#3
	0xA37F	B	--	--	#3
	0xA381	B	--	--	#3
5[#4]	0xA370	B	--	--	--
	0xA372	B	--	--	--
	0xA374	B	--	--	--
6[#4]	0xA371	B	--	--	#3
	0xA373	B	--	--	#3
	0xA375	B	--	--	#3

Legend:

B: Advanced

--: Not applicable

#1

Corresponds to the numbers in *Figure 15-25*.

#2

Trace collection point for `javax.persistence.EntityManager`.

#3

When an exception occurs, the exception is displayed.

#4

Trace collection point for `javax.persistence.Query`.

15.12.6 Trace collection points and trace information that can be collected during the operation of EntityTransaction

(1) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–48: Details of trace collection points during the operation of EntityTransaction

Event ID	No. in the figure[#1]	Trace acquisition points	Level
0xA310[#2]	3	When the processing of EntityTransaction#begin() starts	B
0xA311[#2]	4	When the processing of EntityTransaction#begin() ends	B
0xA312[#2]	5	When the processing of EntityTransaction#commit() starts	B
0xA313[#2]	6	When the processing of EntityTransaction#commit() ends	B
0xA314[#2]	5	When the processing of EntityTransaction#rollback() starts	B
0xA315[#2]	6	When the processing of EntityTransaction#rollback() ends	B
0xA362[#3]	1	When the processing of EntityManager#getTransaction() starts	B
0xA363[#3]	2	When the processing of EntityManager#getTransaction() ends	B

Legend:

B: Advanced

#1

Corresponds to the numbers in *Figure 15-26*.

#2

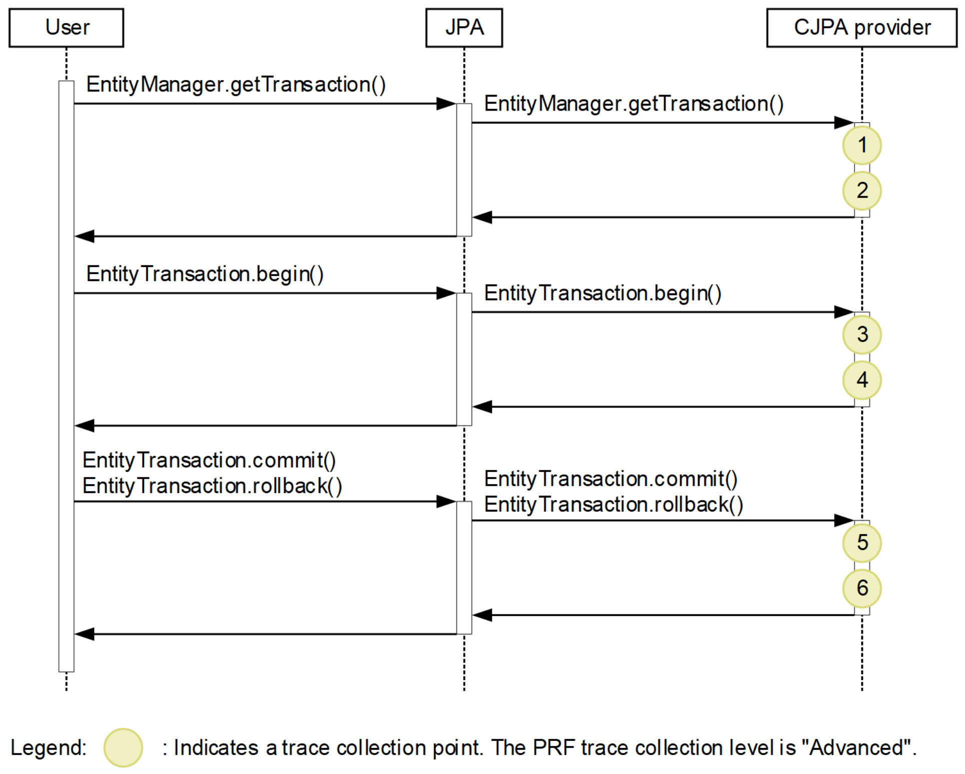
Trace collection point for `javax.persistence.EntityTransaction`.

#3

Trace collection point for `javax.persistence.EntityManager`.

The following figure shows the trace collection points.

Figure 15–26: Trace collection points for the operation of EntityTransaction


User
JPA
CJPA provider
EntityManager.getTransaction()
EntityManager.getTransaction()
1
2
EntityTransaction.begin()
EntityTransaction.begin()
3
4
EntityTransaction.commit()
EntityTransaction.rollback()
EntityTransaction.commit()
EntityTransaction.rollback()
5
6
Legend:
: Indicates a trace collection point. The PRF trace collection level is "Advanced".

(2) Trace information that can be collected

The following table describes the trace information that can be collected during the operation of EntityTransaction.

Table 15–49: Trace information that can be collected during the operation of EntityTransaction

No. in the figure#1	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1#2	0xA362	B	--	--	--
2#2	0xA363	B	--	--	#3
3#4	0xA310	B	--	--	--
4#4	0xA311	B	--	--	#3
5#4	0xA312	B	--	--	--
	0xA314	B	--	--	--
6#4	0xA313	B	--	--	#3
	0xA315	B	--	--	#3

Legend:

B: Advanced

--: Not applicable

#1

Corresponds to the numbers in *Figure 15-26*.

#2

Trace collection point for `javax.persistence.EntityManager`.

#3

When an exception occurs, the exception is displayed.

#4

Trace collection point for `javax.persistence.EntityTransaction`.

15.12.7 Trace collection points and trace information that can be collected in the case of callback method to the user

(1) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–50: Details of trace collection points in the case of callback method to the user

Event ID	No. in the figure#1	Trace acquisition points	Level
0xA390	1	Immediately before invoking the PrePersist() callback method	A
0xA391	2	Immediately after returning from the PrePersist() callback method	A
0xA392	1	Immediately before invoking the PostPersist() callback method	A
0xA393	2	Immediately after returning from the PostPersist() callback method	A
0xA394	1	Immediately before invoking the PreRemove() callback method	A
0xA395	2	Immediately after returning from the PreRemove() callback method	A
0xA396	1	Immediately before invoking the PostRemove() callback method	A
0xA397	2	Immediately after returning from the PostRemove() callback method	A
0xA398	1	Immediately before invoking the PreUpdate() callback method	A
0xA399	2	Immediately after returning from the PreUpdate() callback method	A
0xA39A	1	Immediately before invoking the PostUpdate() callback method	A
0xA39B	2	Immediately after returning from the PostUpdate() callback method	A
0xA39C	1	Immediately before invoking the PostLoad() callback method	A
0xA39D	2	Immediately after returning from the PostLoad() callback method	A

Legend:

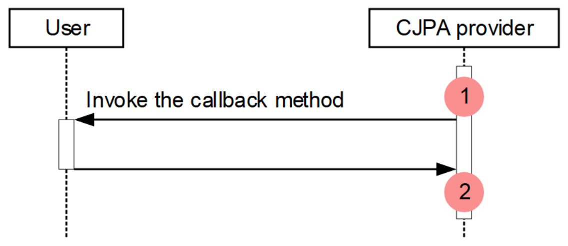
A: Standard

#1

Corresponds to the numbers in *Figure 15-27*.

The following figure shows the trace collection points.

Figure 15–27: Trace collection points of the callback method to the user



Legend: ● : Indicates a trace collection point. The PRF trace collection level is "Standard".

(2) Trace information that can be collected

The following table describes the trace information that can be collected in the case of the callback method to the user.

Table 15–51: Trace information that can be collected in the case of the callback method to the user

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xA390	A	Callback method name specified by the user	--	--
	0xA392	A		--	--
	0xA394	A		--	--
	0xA396	A		--	--
	0xA398	A		--	--
	0xA39A	A		--	--
	0xA39C	A		--	--
2	0xA391	A	Callback method name specified by the user	--	#2
	0xA393	A		--	#2
	0xA395	A		--	#2
	0xA397	A		--	#2
	0xA399	A		--	#2
	0xA39B	A		--	#2
	0xA39D	A		--	#2

Legend:

A: Standard

--: Not applicable

#1

Corresponds to the numbers in *Figure 15-27*.

#2

When an exception occurs, the exception is displayed.

15.12.8 Trace collection points and trace information that can be collected during binary conversion of the entity class

(1) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–52: Details of trace collection points during the binary conversion of the entity class

Event ID	No. in the figure#1	Trace acquisition points	Level
0xA330	1	Immediately before the binary conversion processing for the entity class starts	B
0xA331	2	Immediately after the binary conversion processing for the entity class is complete	B

Legend:

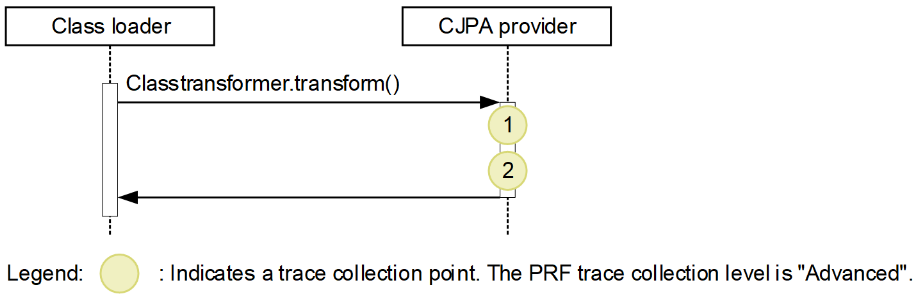
B: Advanced

#1

Corresponds to the numbers in *Figure 15-28*.

The following figure shows the trace collection points.

Figure 15–28: Trace collection points during binary conversion of the entity class


Class loader
CJPA provider
Classtransformer.transform()
1
2
Legend:
: Indicates a trace collection point. The PRF trace collection level is "Advanced".

Tip

In an application using JPA, this trace information is collected even for classes other than the entity class, when the class is loaded from the class loader.

(2) Trace information that can be collected

The following table describes the trace information that can be collected during the binary conversion of the entity class.

Table 15–53: Trace information that can be collected during the binary conversion of the entity class

No. in the figure#1	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xA330	B	--	--	--
2	0xA331	B	--	--	#2

Legend:

B: Advanced

--: Not applicable

#1

Corresponds to the numbers in *Figure 15-28*.

#2

When an exception occurs, the exception is displayed.

15.12.9 Trace collection points and trace information that can be collected during transaction linkage with the transaction manager

(1) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–54: Details of trace collection points during transaction linkage with the transaction manager

Event ID	No. in the figure[#1]	Trace acquisition points	Level
0xA39E	1	Immediately before the process for correlating the persistent context and transaction starts	B
0xA39F	2	Immediately after the process for correlating the persistent context and transaction is complete	B
0xA3A0	3	Immediately before the pre-processing of the JTA transaction conclusion starts	B
0xA3A1	4	Immediately after the pre-processing of the JTA transaction conclusion ends	B
0xA3A2	5	Immediately before the post-processing of the JTA transaction conclusion starts	B
0xA3A3	6	Immediately after the post-processing of the JTA transaction conclusion ends	B

Legend:

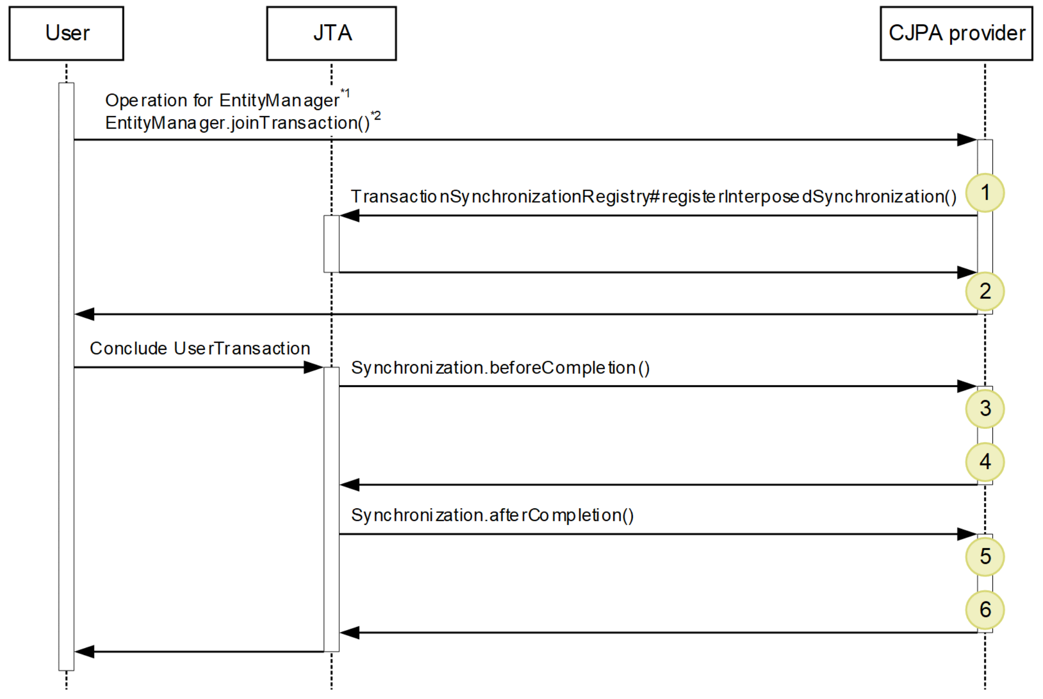
B: Advanced

#1

Corresponds to the numbers in *Figure 15-29*.

The following figure shows the trace collection points.

Figure 15–29: Trace collection points during transaction linkage with the transaction manager


User
JTA
CJPA provider
Operation for EntityManager*1
EntityManager.joinTransaction()*2
TransactionSynchronizationRegistry#registerInterposedSynchronization()
1
2
Conclude UserTransaction
Synchronization.beforeCompletion()
3
4
Synchronization.afterCompletion()
5
6

Legend: ● : Indicates a trace collection point. The PRF trace collection level is "Advanced".

*1 In the case that an operation is performed for EntityManager after UserTransaction is started.

*2 The container might internally invoke EntityManager.joinTransaction().

(2) Trace information that can be collected

The following table describes the trace information that can be collected during transaction linkage with the transaction manager.

Table 15–55: Trace information that can be collected during transaction linkage with the transaction manager

No. in the figure#1	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xA39E	B	--	--	--
2	0xA39F	B	--	--	#2
3	0xA3A0	B	--	--	--
4	0xA3A1	B	--	--	#2
5	0xA3A2	B	--	Status (`int` value of `javax.transaction.Status`) of the transaction passed by an argument of `afterCompletion()`	--
6	0xA3A3	B	--	--	#2

Legend:

B: Advanced

--: Not applicable

#1

Corresponds to the numbers in *Figure 15-29*.

#2

When an exception occurs, the exception is displayed.

15.12.10 Trace collection points and trace information that can be collected during the connection operation of the DB Connector

(1) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–56: Details of trace collection points during the connection operation of the DB Connector

Event ID	No. in the figure#	Trace acquisition points	Level
0xA3A4	1	Immediately before invoking DataSource#getConnection()	B
0xA3A5	2	Immediately after returning from DataSource#getConnection()	B
0xA3A6	1	Immediately before invoking DataSource#getConnection(String, String)	B
0xA3A7	2	Immediately after returning from DataSource#getConnection(String, String)	B
0xA3A8	1	Immediately before invoking DriverManager#getConnection(String, Properties)	B
0xA3A9	2	Immediately after returning from DriverManager#getConnection(String, Properties)	B
0xA3AA	1	Immediately before invoking DriverManager#getConnection(String, String, String)	B
0xA3AB	2	Immediately after returning from DriverManager#getConnection(String, String, String)	B
0xA3AC	9	Immediately before invoking Connection#close()	B
0xA3AD	10	Immediately after returning from Connection#close()	B
0xA3AE	7	Immediately before invoking Connection#commit()	B
0xA3AF	8	Immediately after returning from Connection#commit()	B
0xA3B0	3	Immediately before invoking Connection#createStatement()	B
0xA3B1	4	Immediately after returning from Connection#createStatement()	B
0xA3B2	3	Immediately before invoking Connection#prepareCall(String)	B
0xA3B3	4	Immediately after returning from Connection#prepareCall(String)	B
0xA3B4	3	Immediately before invoking Connection#prepareCall(String, int, int)	B
0xA3B5	4	Immediately after returning from Connection#prepareCall(String, int, int)	B
0xA3B6	3	Immediately before invoking Connection#prepareStatement(String)	B
0xA3B7	4	Immediately after returning from Connection#prepareStatement(String)	B
0xA3B8	3	Immediately before invoking Connection#prepareStatement(String, int, int)	B
0xA3B9	4	Immediately after returning from Connection#prepareStatement(String , int, int)	B
0xA3BA	7	Immediately before invoking Connection#rollback()	B

Event ID	No. in the figure#	Trace acquisition points	Level
0xA3BB	8	Immediately after returning from Connection#rollback()	B
0xA3BC	5	Immediately before invoking Statement#executeQuery(String)	B
0xA3BD	6	Immediately after returning from Statement#executeQuery(String)	B
0xA3BE	5	Immediately before invoking Statement#executeUpdate(String)	B
0xA3BF	6	Immediately after returning from Statement#executeUpdate(String)	B
0xA3C0	5	Immediately before invoking PreparedStatement#executeUpdate()	B
0xA3C1	6	Immediately after returning from PreparedStatement#executeUpdate()	B
0xA3C2	5	Immediately before invoking PreparedStatement#executeQuery()	B
0xA3C3	6	Immediately after returning from PreparedStatement#executeQuery()	B

Legend:

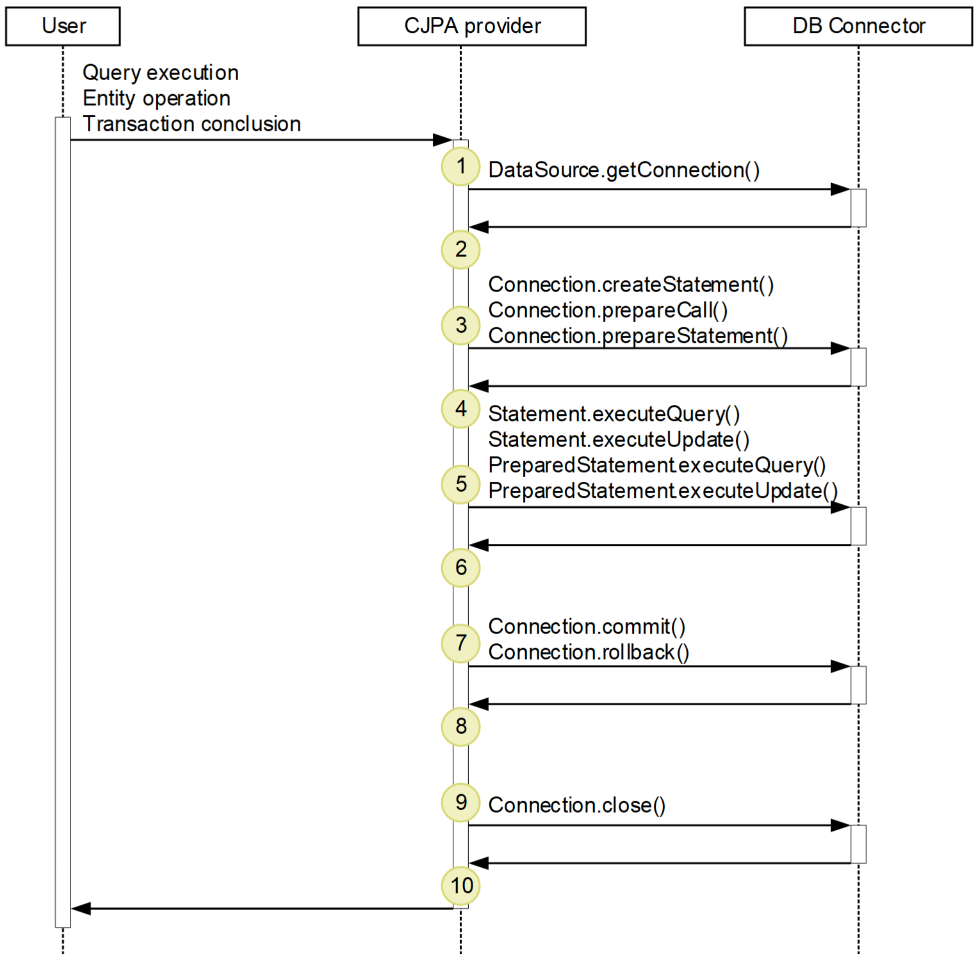
B: Advanced

#

Corresponds to the numbers in *Figure 15-30*.

The following figure shows the trace collection points.

Figure 15–30: Trace collection points during the connection operation of the DB Connector



Legend: ● : Indicates a trace collection point. The PRF trace collection level is "Advanced".

(2) Trace information that can be collected

The following table describes the trace information that can be collected during the connection operation of the DB Connector.

Table 15–57: Trace information that can be collected during the connection operation of the DB Connector

No. in the figure#1	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xA3A4	B	--	--	--
	0xA3A6	B	--	--	--
	0xA3A8	B	--	--	--
	0xA3AA	B	--	--	--
2	0xA3A5	B	--	--	#2
	0xA3A7	B	--	--	#2
	0xA3A9	B	--	--	#2
	0xA3AB	B	--	--	#2
3	0xA3B0	B	--	--	--
	0xA3B2	B	--	--	--
	0xA3B4	B	--	--	--
	0xA3B6	B	--	--	--
	0xA3B8	B	--	--	--
4	0xA3B1	B	--	--	#2
	0xA3B3	B	--	--	#2
	0xA3B5	B	--	--	#2
	0xA3B7	B	--	--	#2
	0xA3B9	B	--	--	#2
5	0xA3BC	B	--	--	--
	0xA3BE	B	--	--	--
	0xA3C0	B	--	--	--
	0xA3C2	B	--	--	--
6	0xA3BD	B	--	--	#2
	0xA3BF	B	--	--	#2
	0xA3C1	B	--	--	#2
	0xA3C3	B	--	--	#2
7	0xA3AE	B	--	--	--
	0xA3BA	B	--	--	--
8	0xA3AF	B	--	--	#2

No. in the figure#1	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0xA3BB	B	--	--	#2
9	0xA3AC	B	--	--	--
10	0xA3AD	B	--	--	#2

Legend:

B: Advanced

--: Not applicable

#1

Corresponds to the numbers in *Figure 15-30*.

#2

When an exception occurs, the exception is displayed.

15.13 Trace collection points of CDI

This section describes the trace collection points of CDI and the trace information that can be collected.

15.13.1 Trace collection points of CDI and the trace information that can be collected

This section describes the trace collection points of CDI and the trace information that can be collected. The following two cases will be described separately:

- When a combination of JSF and CDI is used
- When a combination of servlets and CDI is used

(1) Trace collection points and PRF trace collection levels

(a) When a combination of JSF and CDI is used

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–58: Details of the trace collection points when a combination of JSF and CDI is used

Event ID	No. in the figure#	Trace collection points	Level
0xb002	1	When the reading of the JSF settings required for using CDI starts	A
0xb003	2	When the reading of the JSF settings required for using CDI ends (normal termination)	A
0xb004	3	When the JSF preparations required for using CDI start	A
0xb005	4	When the JSF preparations required for using CDI end (normal termination)	A
0xb006	5	When the EL assessment starts	B
0xb007	6	When the EL assessment ends (normal termination)	B

Legend:

A: Standard

B: Advanced

#

Corresponds to the numbers in *Figure 15-31* and *Figure 15-32*.

The following figure shows the trace collection points.

Figure 15–31: Trace collection points when a combination of JSF and CDI is used (when the JSF settings are read and prepared)

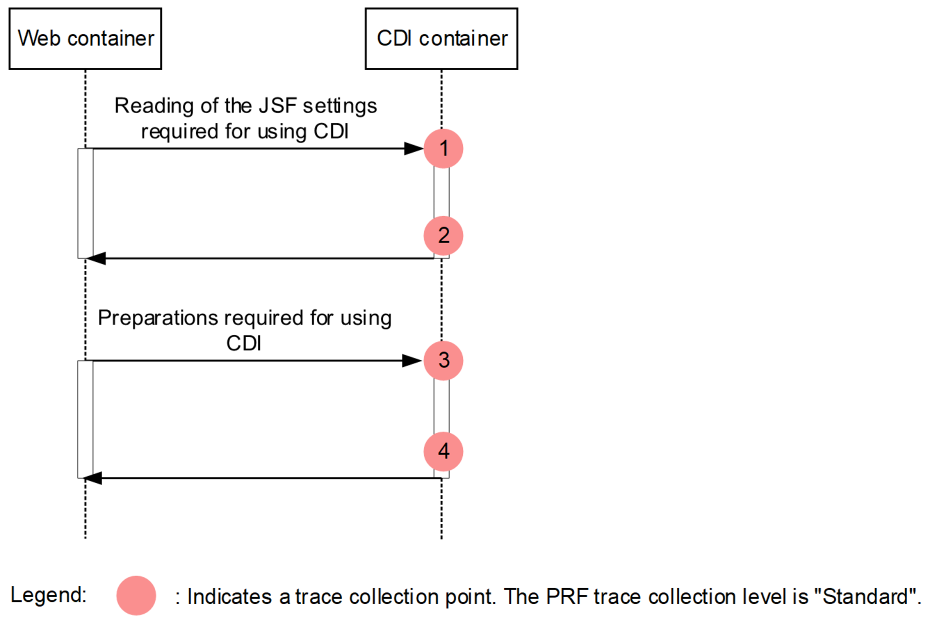
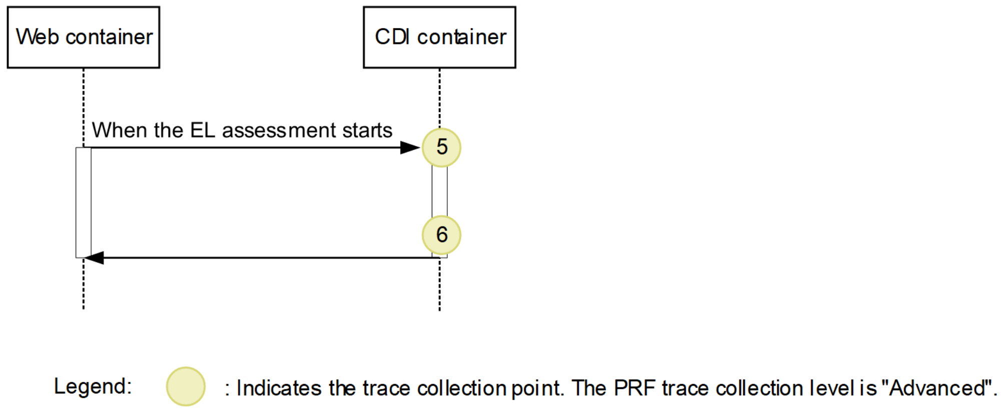

Web container
CDI container
Reading of the JSF settings required for using CDI
1
2
Preparations required for using CDI
3
4
Legend:
: Indicates a trace collection point. The PRF trace collection level is "Standard".

Figure 15–32: Trace collection points when a combination of JSF and CDI is used (for EL evaluation)


Web container
CDI container
When the EL assessment starts
5
6
Legend:
: Indicates the trace collection point. The PRF trace collection level is "Advanced".

(b) When a combination of servlets, filters, listeners, and CDI is used

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 15–59: Details of the trace collection points when a combination of servlets, filters, listeners, and CDI is used

Event ID	No. in the figure#	Trace collection points	Level
0xb008	1	When the generation of servlet/filter/listener instances starts	A
0xb009	2	When the generation of servlet/filter/listener instances ends (normal termination)	A

Legend:

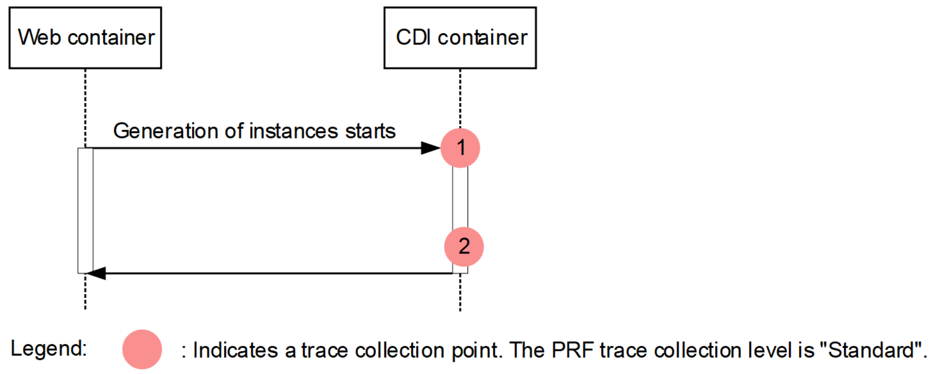
A: Standard

#

Corresponds to the numbers in *Figure 15-33*.

The following figure shows the trace collection points.

Figure 15–33: Trace collection points when a combination of servlets, filters, listeners, and CDI is used


Web container
CDI container
Generation of instances starts
1
2
Legend:
: Indicates a trace collection point. The PRF trace collection level is "Standard".

(2) Trace information that can be collected

(a) When a combination of JSF and CDI is used

The following table describes the trace information that can be collected when a combination of JSF and CDI is used.

Table 15–60: Trace information that can be collected when a combination of JSF and CDI is used

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xb002[#2]	A	`WeldFacesConfigProvider`	--	Context information of the Web container
2	0xb003[#2]	A	`WeldFacesConfigProvider`	--	Entrance time
3	0xb004[#2]	A	`WeldApplicationFactory`	--	--
4	0xb005[#2]	A	`WeldApplicationFactory`	--	Entrance time
5	0xb006[#3]	B	`WeldApplication`	--	--
6	0xb007[#3]	B	`WeldApplication`	--	Entrance time

Legend:

A: Standard

B: Advanced

--: Not applicable

#1

Corresponds to the numbers in *Figure 15-31* and *Figure 15-32*.

#2

The trace information for the reading of the JSF settings required for using CDI and for the JSF preparations required for using CDI is collected when the application starts.

#3

The trace information for EL assessment is collected when `FacesServlet` is initialized and when Expression Language specified in JSF is evaluated.

(b) When servlets are invoked from CDI

The following table describes the trace information that can be collected when servlets are invoked from CDI.

Table 15–61: Trace information that can be collected when servlets are invoked from CDI

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xb008	A	`CDIServiceImpl`	--	The following information is output: • `Managed` class • Class name • Context root
2	0xb009	A	`CDIServiceImpl`	--	Entrance time

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in *Figure 15-33*.

Note that the trace information is collected when the servlet/filter/listener interfaces are generated.

16 Output Log Information and Log Acquisition Settings

This chapter describes the log information that is output and the settings for acquiring logs.

16.1 Log Information Output for Each Functionality

This section describes the log information output when you use a specific functionality.

The log information output when you use the following functionality is described below:

- Cosminexus JPA provider operation log

16.1.1 Cosminexus JPA Provider operation log

This appendix describes the operation log messages that are output by Cosminexus JPA Provider for each category of the log output.

(1) When category is SQL

SQL [*aa....aa*] SQL = *bb....bb*

aa....aa: Persistence unit name

bb....bb: Issued SQL statement

Description

JPA issues SQL.

SQL [*aa....aa*] PARAM = *bb....bb*

aa....aa: Persistence unit name

bb....bb: Set value of `?` parameter (place holder)

Description

Sets the value in `?` parameter (place holder) of the SQL statement.

Result where the value set to `?` parameter (place holder) is converted into a character expression is output in *bb....bb*. For the character or numeral value the result is output as it is. However, for example, when the binary data is stored in the byte type array of java, the `toString` method is executed for the byte type array the output is `B@f7ba93`.

SQL [*aa....aa*] RETURN = *bb....bb*

aa....aa: Persistence unit name

bb....bb: Return value

Description

Return value of `PreparedStatement#executeUpdate()`. The return value is the number of lines when the SQL statement is executed. For details, see *java.sql.PreparedStatement* of Javadoc.

(2) When category is TRANSACTION

TRN [*aa....aa*] JPA processing was bound to a JTA transaction. (status = *bb....bb*)

aa....aa: Persistence unit name

bb....bb: Transaction state when bound to JTA transaction

Description

JTA transaction is bounded. Outputs the character string that indicates the transaction state (status). For details about the character string that indicates the status, see *javax.transaction.Status* of Javadoc.

TRN [*aa....aa*] A JTA transaction was committed. (status = *bb....bb*)

aa....aa: Persistence unit name

bb....bb: Transaction state when JTA transaction is concluded

Description

The conclusion of JTA transaction is notified. Outputs the character string that indicates the state (status) where the transaction passed from JTA is concluded. For details about the character string indicating the status, see *javax.transaction.Status* of Javadoc.

TRN [*aa....aa*] An EntityTransaction started.

aa....aa: Persistence unit name

Description

EntityTransaction is started.

TRN [*aa....aa*] An EntityTransaction was committed.

aa....aa: Persistence unit name

Description

EntityTransaction is committed.

TRN [*aa....aa*] An EntityTransaction was rolled back.

aa....aa: Persistence unit name

Description

EntityTransaction is rolled back.

16.2 Settings for acquiring the in-process HTTP server log

This section describes the items that can be set up for acquiring the in-process HTTP server log.

In the in-process HTTP server, the access log, trace based performance analysis, thread trace, and the communication trace are output for supporting the application development, for performance analysis at operation time, and for troubleshooting at failure detection time. For these files, you can change the number of files and the file size in the Easy Setup definition file.

The following table describes the settings that you can change for acquiring the in-process HTTP server log and parameters of the Easy Setup definition file corresponding to the items.

Table 16–1: Settings for acquiring the in-process HTTP server log

Log or trace	Items	Corresponding parameters of the Easy Setup definition file
Access log	Availability access log output	`webserver.logger.access_log.inprocess_http.enabled` in the `<configuration>` tag on a logical J2EE server (j2ee-server) (By default access log is output)
	Access log file name	`webserver.logger.access_log.inprocess_http.filename` in the `<configuration>` tag on a logical J2EE server (j2ee-server)
	File size of the access log	`webserver.logger.access_log.inprocess_http.filesize` in the `<configuration>` tag on a logical J2EE server (j2ee-server)
	Number of files of access log	`webserver.logger.access_log.inprocess_http.filenum` in the `<configuration>` tag on a logical J2EE server (j2ee-server)
	Format name of the access log	`webserver.logger.access_log.format_list` in the `<configuration>` tag on a logical J2EE server (j2ee-server)#
	Output format of access log	`webserver.logger.access_log.`*format-name* in the `<configuration>` tag on a logical J2EE server (j2ee-server)#
	Format when the access log is output	`webserver.logger.access_log.inprocess_http.usage_format` in the `<configuration>` tag on a logical J2EE server (j2ee-server)#
Trace based performance analysis	--	Specify acquisition condition, when you want to execute the `cprfed` command for performing a daily system operation same as for other trace based performance analysis. For details on acquiring the performance analysis trace file, see *15. Performance Analysis Trace*.
Thread trace	Number of files of thread trace	`webserver.logger.thread_trace.inprocess_http.filenum` in the `<configuration>` tag on a logical J2EE server (j2ee-server)
	File size of thread trace	File size of the thread trace =(((A+B) × 32,786)+ 32,914) byte A= Value of `webserver.connector.inprocess_http.max_connections` parameter in the `<configuration>` tag on a logical J2EE server (j2ee-server) B=For `0`, the value of the `webserver.connector.inprocess_http.send_timeout` parameter is 0, and when other than 0, the value is `1` in the `<configuration>` tag on a logical J2EE server (j2ee-server)

Log or trace	Items	Corresponding parameters of the Easy Setup definition file
Communication trace	Number of files of the communication trace	`webserver.logger.communication_trace.inprocess_http.filenum` in the `<configuration>` tag on a logical J2EE server (j2ee-server).
	File size of the communication trace	File size of the communication trace=(((A+B) × 172,050)+128) byte A=Value of `webserver.connector.inprocess_http.max_connections` parameter in the `<configuration>` tag on a logical J2EE server (j2ee-server) B=For `0`, the value of the `webserver.connector.inprocess_http.send_timeout` parameter is 0, and when other than 0, the value is `1` in the `<configuration>` tag on a logical J2EE server (j2ee-server).

Legend:

--: Not applicable

#

In the access log, you can customize the log output format by defining the format with the above-mentioned keys. For details on customizing the access log of the in-process HTTP server, see *6.17.2 Customizing the access log of the in-process HTTP server*.

17 System Design Guide (V9 Compatibility Mode)

This chapter describes how to tune the performance of a system for executing J2EE applications.

You can maximize the performance of the system by optimizing the operating environment through performance tuning.

For details on performance tuning for the batch application execution platform, see *9. Performance Tuning (Batch Application Execution Platform)* in the manual *uCosminexus Application Server System Design Guide*.

17.1 Points to be considered for performance tuning

This section explains the points to be considered for performance tuning of the J2EE application execution platform.

17.1.1 Viewpoints for performance tuning

Tune the performance of the J2EE application execution platform with the following viewpoints:

- **Optimizing the number of concurrent executions**
- **Optimizing the method of invoking the Enterprise Bean**
- **Optimizing the method of accessing the database**
- **Setting the timeout**
- **Optimizing the operation of the Web application**
- **Optimizing the operation of CTM**
- **Tuning of other items**

These points are explained below:

(1) Optimizing the number of concurrent executions

The objective of optimizing the number of concurrent executions is to enhance the throughput of the system by multi-processing in order to maximize the CPU performance. In the following cases, however, only multi-processing may not enhance throughput; sometimes the throughput may even deteriorate:

- Bottlenecks in I/O processing and lock processing
- Maximum throughput has already been reached
- Load exceeds the multiplicity when the CPU usage is already full
- Pending queue size is inappropriate
- Settings for maximum number of hierarchical executions is inappropriate

Performance tuning aims at optimizing the number of concurrent executions through proper tuning for the above points.

(2) Optimizing the method of invoking the Enterprise Bean

The objective of optimizing the method of invoking the Enterprise Bean is to restrict unnecessary network access by using the local invocation functionality of the local interface and the remote interface when invoking the components in the same J2EE application and the same J2EE server.

You can restrict the unnecessary network access caused by RMI-IIOP communication, by using the following functionality:

- Use of local interface
- Use of local invocation functionality of the remote interface

In addition, you can further enhance the processing performance by using pass by reference method of passing the argument and return value. Performance tuning aims at enhancing the processing performance by effectively using these functions depending on the features of the application and the system.

(3) Optimizing the database access method

The purpose of optimizing the database access method is to reduce the overheads during database access by generating in advance the connections and statements that are likely to need more time for processing.

Performance tuning enhances throughput by optimizing the database access by using the following functionality effectively:

- Connection pooling
- Statement pooling (pooling of PreparedStatement and CallableStatement)

(4) Setting the timeout

The purpose of setting the timeout is to detect a failure that may occur in the system and release the resources whenever required to avoid a delay in responding to the requests.

There are following types of timeout settings:

- Timeout of Web front-end system
- Timeout of back-end system
- Timeout of transaction
- Timeout of database

(5) Optimizing the Web application operations

The purpose of optimizing the Web application operations is to increase the processing speed by restricting unnecessary network access that results from the use of cache and determination of delivery method of contents, and enhance the system throughput with the help of load balancing.

Note that the items that can be tuned differ depending on whether you are integrating with a Web server in which a redirector module is embedded, or whether you are using an in-process HTTP server.

You can execute the following processes for connecting to the Web server:

- Dividing processes in the Web application and static contents
- Caching static contents
- Dividing requests according to session information

You can execute the following processes when you use an in-process HTTP server:

- Separating the deployment of static contents from Web applications
- Caching static contents

(6) Optimizing the operation of CTM

The purpose of optimizing the operation of CTM is to improve the performance of the system by reducing the communication overhead through optimization of the communication interval between the processes used in CTM, and by promptly detecting and taking action when a trouble occurs. Moreover, by prioritizing the processing of requests by CTM, you can tune to execute quick processing of the important requests.

(7) Tuning other items

In addition to those explained above in points (1) to (6), application server has other items that can be tuned. These can be tuned whenever required.

17.1.2 Items that can be tuned for each type of application

The tuning items differ as per the type of application. The table below describes the tuning items for each component of the application.

Table 17–1: Tuning items of an application consisting of Servlet and JSP (Web application)

Tuning Items	Available functionality	Reference
Optimizing the number of request-processing threads (when using an in-process HTTP server)	Control the number of request-processing threads (when using an in-process HTTP server)#	*17.3.1*
Optimizing the number of concurrent executions	Concurrently executed thread control in the Web application (each Web container, Web application, or URL group)	*17.3.2*
Optimizing the method for accessing the database	Connection pooling	*8.5.1* in the *uCosminexus Application Server System Design Guide*
	Statement pooling	*8.5.2* in the *uCosminexus Application Server System Design Guide*
Setting the timeout	Setup of timeout in the Web front-end system	*17.4.2*
	Setup of timeout for the method execution times of J2EE applications	*8.6.7* in the *uCosminexus Application Server System Design Guide*
Optimizing the operation of Web application	Separation of the deployment of static contents and Web application	*8.7.1* in the *uCosminexus Application Server System Design Guide*
	Caching static contents	*8.7.2* in the *uCosminexus Application Server System Design Guide*
	Distributing the requests using the redirector (for Web server integration)	*5.2*
Tuning other items	Control the Persistent Connection (when using an in-process HTTP server)	*17.6*

#
When integrating with a Web server, tune using the Web server functionality.

Table 17–2: Tuning items of application configured by Enterprise Bean

Tuning items	Available functionality	Reference
Optimizing the number of concurrent executions	Pooling of Stateless Session Bean instances	*8.3.5* in the *uCosminexus Application Server System Design Guide*
	Session control of Stateful Session Bean	
	Pooling of Message-driven Bean instances	

Tuning items	Available functionality	Reference
	Control the number of concurrent executions with CTM# (when using CTM)	*8.3.6* in the *uCosminexus Application Server System Design Guide*
Optimizing the method of invoking Enterprise Bean	Use of local interface	*8.4.1* in the *uCosminexus Application Server System Design Guide*
	Optimization of local invocation of remote interface	*8.4.2* in the *uCosminexus Application Server System Design Guide*
	Pass by reference for the remote interface	*8.4.3* in the *uCosminexus Application Server System Design Guide*
Optimizing the method of accessing database	Connection pooling	*8.5.1* in the *uCosminexus Application Server System Design Guide*
	Statement pooling	*8.5.2* in the *uCosminexus Application Server System Design Guide*
Setting the timeout	Setup of timeout in the back-end system	*8.6.3* in the *uCosminexus Application Server System Design Guide*
	Setup of transaction timeout	*8.6.4* in the *uCosminexus Application Server System Design Guide*
	Setup of timeout for database	*8.6.6* in the *uCosminexus Application Server System Design Guide*
	Setup of timeout for the method execution times of J2EE applications	*8.6.7* in the *uCosminexus Application Server System Design Guide*

#
Applicable only for the Stateless Session Bean.

The following table describes the tuning items of CTM operation that can be set up in systems that use CTM. You can use CTM when Stateless Session Beans configure an application.

Table 17–3: Tuning items of CTM operation

Tuning items	Available functionality	Reference
Optimizing the operation of CTM	Tuning of the monitoring interval of the operation state of CTM domain managers and CTM daemons	*8.8.1* in the *uCosminexus Application Server System Design Guide*
	Tuning of the monitoring interval of the load status	*8.8.2* in the *uCosminexus Application Server System Design Guide*
	Setup of a timeout lock for CTM daemon	*8.8.3* in the *uCosminexus Application Server System Design Guide*

Tuning items	Available functionality	Reference
	Setup of a priority order for the requests distributed with CTM	*8.8.4* in the *uCosminexus Application Server System Design Guide*

17.2 Tuning Method

This section describes the tuning method and the way it differs according to the type of the object that will be set.

17.2.1 Tuning of J2EE server and Web server (including redirector)

You use the Easy Setup definition file of the Smart Composer functionality for tuning the J2EE server and Web server (including redirector). In the Easy Setup definition file, specify type of the logical server (J2EE server or Web server) you want to set in <*logical-server-type*> under the `<configuration>` tag, and specify the parameter name and its value under the `<param>` tag. For details about the Easy Setup definition file, see *4.3 Easy Setup definition file* in the *uCosminexus Application Server Definition Reference Guide*.

Reference note

When you cannot use the Smart Composer functionality, define the tuning of Web server (Including redirector) by editing the file.

The following table describes the files to be used for tuning the Web server (including redirector) when you cannot use the Smart Composer functionality:

Table 17–4: Files to be used for tuning the Web server (including redirector) when the Smart Composer functionality cannot be used

Target	Tuning method
Web server	Edit `httpsd.conf`
Web server (redirector)	Edit `mod_jk.conf` (for Cosminexus HTTP Server)
	Edit `isapi_redirect.conf` (for Microsoft IIS)
	Edit `workers.properties` (for worker settings)

For details on `mod_jk.conf`, see *14.2.2 mod_jk.conf (Redirector action definition file for Cosminexus HTTP Server)*. For details on `isapi_redirect.conf`, see *14.2.1 isapi_redirect.conf (Redirector action definition file for Microsoft IIS)*. For details on `workers.properties`, see *14.2.4 workers.properties (Worker definition file)*. For details about `httpsd.conf`, see the manual *Cosminexus HTTP Server*.

17.3 Optimizing the number of concurrent executions

This section describes the various tuning methods and the concept of optimizing the number of concurrent executions of application requests.

17.3.1 Controlling the number of request-processing threads in a Web server

In the case of Web front-end systems, the request-processing thread created by the Web server processes the requests from clients like the Web browser. The processing efficiency can be improved by appropriately controlling the number of request-processing threads.

This section describes the purpose of controlling the number of request-processing threads in Web servers, and the guidelines for tuning.

This section also describes tuning methods if you use an in-process HTTP server.

Reference note

When Cosminexus HTTP Server is used for Web server integration, similar tuning can be done for the settings of Cosminexus HTTP Server. For details, see the manual *Cosminexus HTTP Server*.

In order to build a system using the Smart Composer functionality, you can use abstract parameters for setting the number of request-processing threads in the Web server. An abstract parameter means several mutually related parameters merged into a single parameter. Use an abstract parameter to define the settings for the number of request-processing threads of Web server along with the related parameters such as the number of concurrent executions. For details on abstract parameters, see *Appendix I. Abstract parameters available with the Smart Composer Functionality (in V9 compatibility mode)* in the manual *uCosminexus Application Server Command Reference Guide*.

(1) Purpose of controlling the number of request-processing threads

The performance can be improved by tuning the number of request-processing threads according to the quality of the host on which the J2EE server is running, and according to the status of access from the client.

The process of generating request-processing threads puts a heavy load on the system. By generating and pooling the request-processing threads beforehand, you can reduce the load for a processing request from the client, such as a Web browser and can increase the processing performance.

When you use an in-process HTTP server, you can generate and pool the request-processing threads in batch when the J2EE server is started, and can use this pool in the event of a processing request from the client, such as a Web browser. This leads to an improvement in the processing performance when a processing request is received. By monitoring the number of pooled threads, you can generate additional threads when the number of pooled threads becomes less, and can secure these threads in the pool.

However, if you pool a large number of unused threads, it will lead to unnecessary consumption of resources. Therefore, depending on the processing contents of the system, you must appropriately control the number of request-processing threads to be pooled, and delete the unnecessary threads, if required.

When controlling the request-processing threads, set up appropriate values in the parameters by considering the above.

(2) Guidelines for setup

When controlling the number of request-processing threads, you can tune using the following parameters:

- Number of request-processing threads generated when the J2EE server is started
- Maximum number of connections to the Web client (number of request-processing threads)
- Maximum value of the Listen queue (back log) of TCP/IP, when the maximum number of connections is exceeded
- Maximum and minimum number of spare threads
- Choosing whether or not to maintain the number of request-processing threads generated when the J2EE server is started

Make a note of the following points when setting up these parameters:

- Depending on the contents of the service to be provided, a large number of requests must be processed immediately after the J2EE server is started. In such a case, set up a large value in the number of the request-processing threads to be generated when the J2EE server is started.
- If you increase the maximum number of spare threads, you can promptly handle a sudden increase in the number of accesses from the client, without a decline in the processing performance. However, if you pool a large number of spare threads, many resources will be consumed. Therefore, be careful when setting the appropriate number of pooled spare threads, after estimating the presumed sudden increase in the number of accesses.
- If a fixed number of request-processing threads are secured, and you want to control the increase and decrease in the number of request-processing threads over and above the fixed number by specifying a maximum number and minimum number, specify the settings for maintaining the number of request-processing threads generated when the J2EE server is started. By doing this, you can handle an increase or decrease in the number of the request-processing threads during peak access from the client, when the minimum number of request-processing threads desired for the system is secured. The number of request-processing threads generated when the J2EE server is started will be maintained even when the number of unused request-processing threads exceeds the maximum number of spare threads.
- If you want to keep on pooling the threads that are created, without deleting them, set up the maximum number of spare threads to a value same as the maximum number of connections to the Web client.

Apart from the above, consider the relationship with the number of concurrently executed threads of the Web application. For details about the number of concurrently executed threads of a Web application, see *17.3.2 Controlling the number of concurrent executions of a Web application*.

17.3.2 Controlling the number of concurrent executions of a Web application

In the case of a Web front-end system, the control of concurrently executed threads of a Web Application controls the number of threads in which the Web server will concurrently process the requests received from clients like the Web browser.

The number of concurrent executions is controlled in each URL group, Web application or Web container. The number of concurrently executed threads can be controlled when you integrate with a Web server, or when you use an in-process HTTP server.

(1) Difference in the control of number of concurrently executed threads

The difference in the control of the number of concurrently executed threads in the case of each Web container, Web application and URL group is explained below:

Web container

You can set the number of threads for simultaneous processing of requests in the entire Web container.

Web application

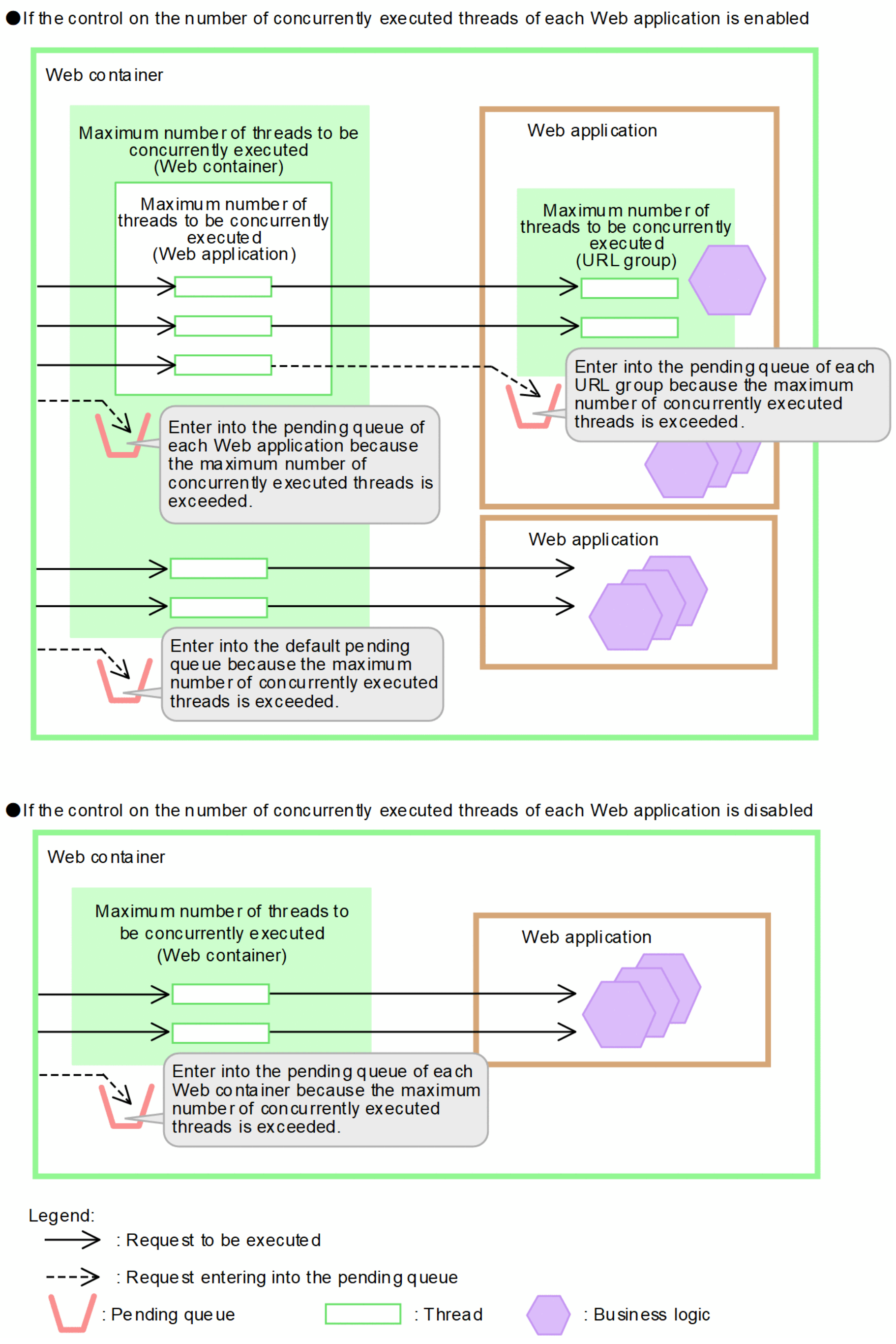
You can set the number of threads for simultaneous processing of requests for each Web application running in the Web container.

URL group

You can set the number of threads for simultaneous processing of requests for each process of distribution destination URL, when a request is distributed to a URL corresponding to the specific business process (business logic) in the Web application.

The figure below illustrates the relationship between concurrently executed threads of each Web container, Web application, and URL group.

Figure 17–1: Relationship between concurrently executed threads of each Web container, Web application, and URL group


●If the control on the number of concurrently executed threads of each Web application is enabled
Web container
Maximum number of threads to be concurrently executed (Web container)
Maximum number of threads to be concurrently executed (Web application)
Web application
Maximum number of threads to be concurrently executed (URL group)
Enter into the pending queue of each URL group because the maximum number of concurrently executed threads is exceeded.
Enter into the pending queue of each Web application because the maximum number of concurrently executed threads is exceeded.
Web application
Enter into the default pending queue because the maximum number of concurrently executed threads is exceeded.
●If the control on the number of concurrently executed threads of each Web application is disabled
Web container
Maximum number of threads to be concurrently executed (Web container)
Web application
Enter into the pending queue of each Web container because the maximum number of concurrently executed threads is exceeded.
Legend:
: Request to be executed
: Request entering into the pending queue
: Pending queue
: Thread
: Business logic

Important note

Control of the maximum number of threads to be concurrently executed for each Web container is enabled only when the control of the number of threads to be concurrently executed for each Web application is disabled.

When control of the number of threads to be concurrently executed for each Web application is enabled, the maximum number of threads to be concurrently executed for each Web container is checked by the control of the number of threads to be concurrently executed for each Web application. For details, see the manual *uCosminexus Application Server Web Container Functionality Guide*.

The execution of requests for a Web application is controlled by the number of concurrently executed threads set in each Web container, Web application, and URL group. A request that exceeds the number of concurrently executed threads that are set in each Web container, Web application, and URL group enters the respective pending queue.

(2) Guidelines for selection

The guidelines for selecting the unit of control for concurrently executed threads are explained below:

For details about the functionality for controlling the number of concurrently executed threads, see *2.13 Overview of control over the number of concurrently executed threads* in the *uCosminexus Application Server Web Container Functionality Guide*.

- **Guidelines for selecting a Web application**

  The J2EE server can manage not only the TCP connection requests but also the pending queue of the Web applications by controlling the number of concurrently executed threads of the Web application. Therefore, Hitachi recommends setting a number of concurrently executed threads of a Web application even when a single Web application is running on the J2EE server.

  Setting the number of concurrently executed threads in the Web application has the following advantages as compared to the setting of the number concurrently executed threads in each Web container:

  - Setting the maximum number of concurrently executed threads for each Web application will prevent the Web application with a large number of requests, corresponding to a particular business, from using the processing efficiency of the entire Web container. As a result, the other businesses can also be executed without any delay.
  - If there are several Web applications with different loads required for CPU and I/O processing, you can set the number of concurrently executed threads according to the conditions of respective Web applications.
  - As the size of the pending queue for requests can be set for each Web application separately, it is possible to control the pending queue according to the features of the Web application. In addition, if the requests exceeding the pending queue size of each Web application are sent, it can be communicated to the client using the HTTP response code.

  The number of concurrently executed threads of each Web application can be dynamically changed in the running J2EE server. For details about the procedure for dynamically changing the number of concurrently executed threads of Web applications executed on running J2EE servers, see *2.17.2 Flow of dynamically changing the number of concurrently executed threads* in the *uCosminexus Application Server Web Container Functionality Guide*.

- **Guidelines for selecting a URL group**

  If concurrently executed threads are controlled in a Web application, control the concurrently executed threads in a URL group if you want to further control the concurrently executed threads in business logic.

  Determine the settings of a URL group to include the following business logic in the Web application:

  - Business logic that you want to prioritize and execute without being influenced by other processes
  - Business logic where the CPU and I/O load is high or the required processing time is more as compared to other processes

  Setting the concurrently executed threads in a URL group has the following advantages as compared to setting only in a Web application.

  - The threads that need to be executed are allocated to the business logic (URL group) with a high priority. Even when the requests for another business logic increase, you can execute the business logic with a high priority instead of assigning the concurrently executed threads of the entire Web application to the business logic with more requests.
  - Setting the upper limit for concurrently executed threads of business logic (URL group) that require more processing time makes it possible to prevent a particular business logic from using up the concurrently executed threads of the entire Web application.

- If a Web application has several business logics (URL groups) with CPU and I/O having different loads, you can set the number of concurrently executed threads according to the business logic.
- Setting the queue size (pending queue size) for the request of each business logic (URL group) in the Web application allows the controlling of the pending queue according to the features of the business logic. If the pending queue of each URL group exceeds the upper limit, the same is notified to the client by using the HTTP response code 503 (Service Temporarily Unavailable).

17.4 Setting a timeout

In the Application Server, you can set a timeout at several points in order to prevent the state where there is no response to a request in case of an error.

This section describes the points where you can set a timeout in the entire system and the guidelines for setting the timeout.

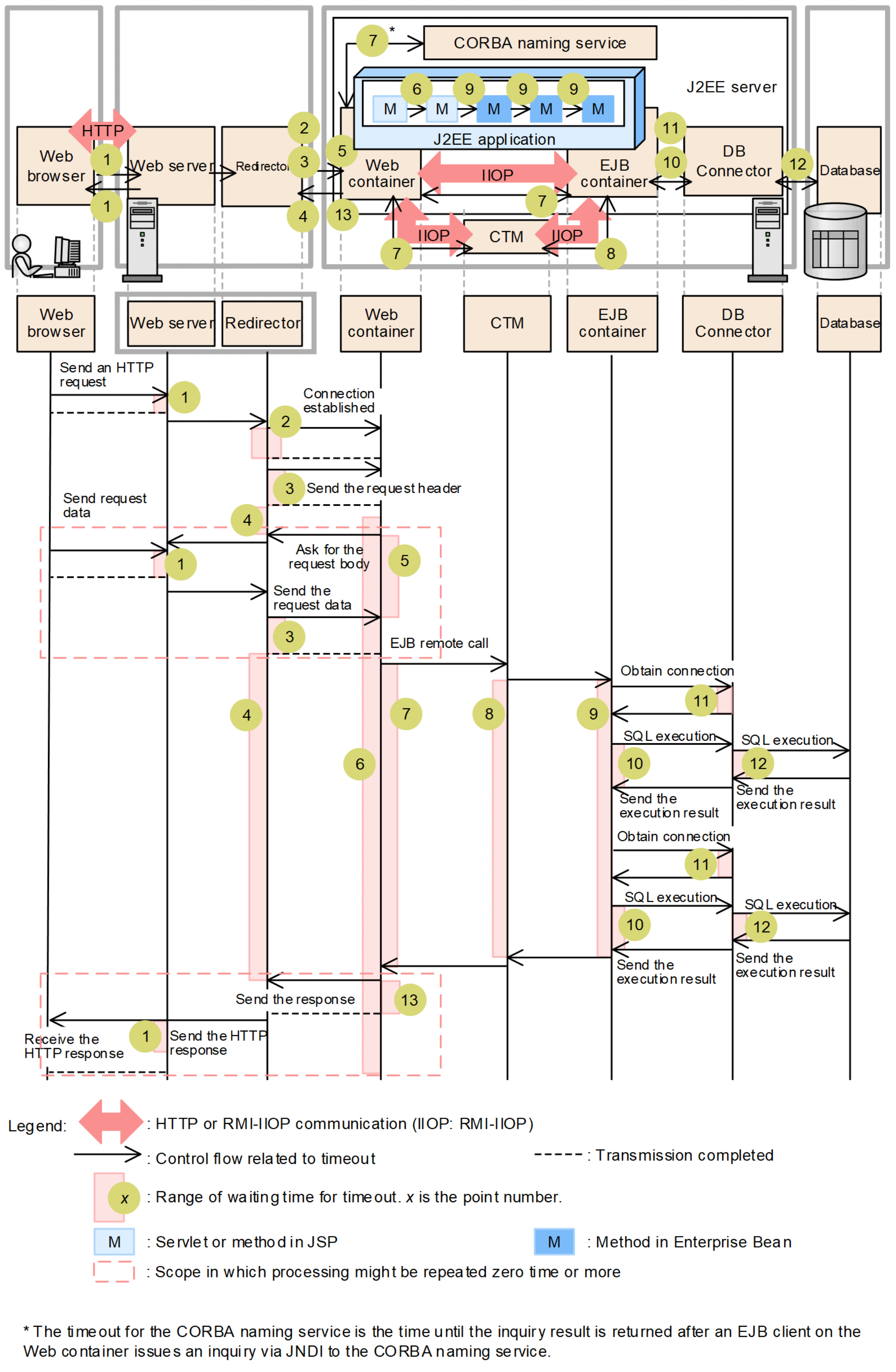
Reference note

When invoking Application Server from OpenTP1 using the TP1 inbound integrated function, you must perform timeout settings in view of the settings in OpenTP1 besides the contents described in this section. For details, see *4. Invoking Application Server from OpenTP1 (TP1 inbound integrated function)* in the *uCosminexus Application Server Common Container Functionality Guide*.

17.4.1 Points where a timeout can be set

In the systems used for executing J2EE applications, you can set up timeouts at the points shown in the following figure. In the following figure, a Web browser is used as the client. The points will differ when you integrate with a Web server, and when you use an in-process HTTP server.

Figure 17–2: Points where a timeout can be set (for Web server integration)

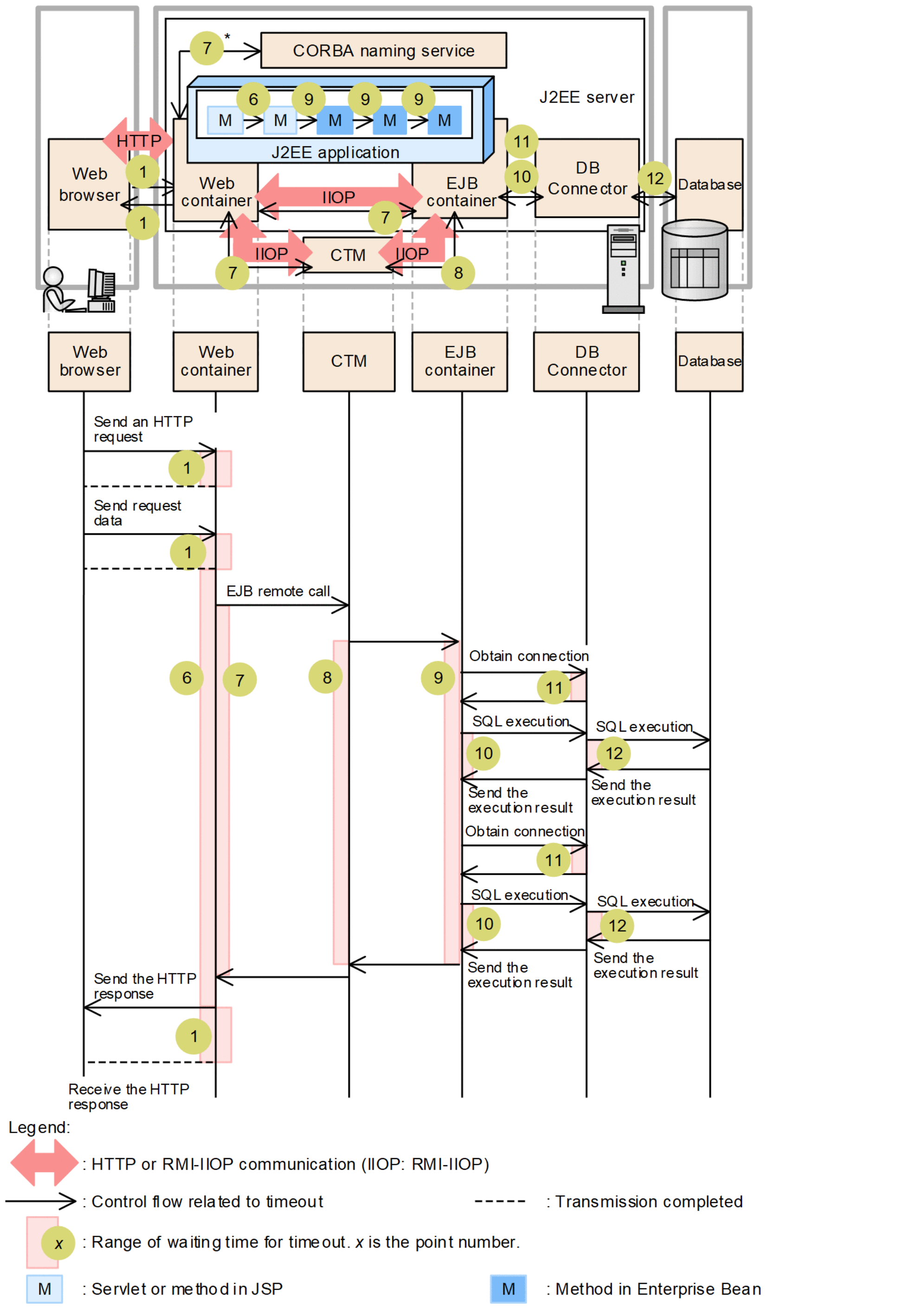


* The timeout for the CORBA naming service is the time until the inquiry result is returned after an EJB client on the Web container issues an inquiry via JNDI to the CORBA naming service.

If the client is an EJB client, replace the Web container with the EJB client. You can set the timeout ranging from the EJB client up to the database.

A redirector will not be applicable when you use an in-process HTTP server, and therefore, a timeout will not be set up at points 2 to 5, and 13. The following figure shows the points where you can set up timeouts to use in-process HTTP servers:

Figure 17–3: Points where timeouts can be set up (for in-process HTTP servers)



* The timeout for the CORBA naming service is the time until the inquiry result is returned after an EJB client on the Web container issues an inquiry via JNDI to the CORBA naming service.

The timeout specified at each point has a specific use that is described in the table below:

Table 17–5: Purpose of timeout set at each point and default timeout settings

Points	Type of timeout	Primary usage
1	Timeout set in the server for receiving the request from the client and sending the data to the client	For Web server integration Detecting failures in the communication path or the Web server

Points	Type of timeout	Primary usage
		For in-process HTTP servers Detecting failures in the communication path, or access from an invalid client
2	Connection timeout specified in the redirector in the processes for sending the requests to the Web container	Detecting failures in the communication path or the Web container
3	Timeout for sending the request header and request body set in the redirector in the processes for sending the requests to the Web container	Detecting failures in the communication path or the Web container
4	Timeout set in the redirector for receiving data from the Web container	Detecting failures in business processing (such as infinite loop and deadlock) of the J2EE server or the communication path
5	Timeout set in the Web container for receiving data from the redirector	Detecting failures in the communication path or the Web server
6	Timeout set in the Web application for the method execution time	Detecting failures in business processing (such as infinite loop and deadlock) of the J2EE server
7	Timeout set in the EJB client for remotely invoking the Enterprise Bean (RMI-IIOP communication) and for invoking the JNDI Naming Service	Detecting failures in business processing (such as infinite loop and deadlocks) of the J2EE server or the communication path
8#	Timeout set up in the EJB client for invoking the Enterprise Bean from CTM	Detecting failures in business processing (such as infinite loop and deadlocks) of the J2EE server or the communication path
9	Timeout set in the EJB for the method execution time	Detecting failures in business processing (such as infinite loop and deadlock) of the J2EE server
10	Timeout set in the EJB container for the database transaction	Detecting failures in database server (such as server is down or a deadlock has occurred) or preventing the extended exclusive use of the resources
11	Timeout set in DB Connector for acquiring a connection	Detecting errors when a connection is acquired (communication path errors or resource depletion)
12	Database timeout	Detecting failures in database server (such as server is down or a deadlock has occurred) or preventing the extended exclusive use of the resources
13	Timeout set in the Web container for sending a response to the redirector	Detecting failures in the communication path or the redirector

#

This point exists only when you are using CTM. For a configuration in which CTM is not used, the range of point 7 extends from the time of execution of remote invocation of the EJB from the Web container to the EJB container, until the dispatch of execution result from the EJB container to the Web container.

The basic guidelines for setting the above timeouts are as follows:

- The general rule for setting a timeout value is that the closer a point is from the invocation origin (Web client or EJB client), the higher the timeout value. It is, therefore, recommended to use the following relationship for setting the timeout.
  - Point 1 < Point 5
  - Point 4 > Point 6 > Point 7
  - Point 7 = Point 8 > Point 9 > Point 10
  - Point 10 > Point 11
  - Point 9 > Point 12

    - Point 1 < Point 13
- When setting the timeout values for points 4, 7, 10 and 12, first check the amount of time normally taken by the invocation process, and then calculate and set the timeout value for each invocation process (business).

The points 1 to 13 can be divided into the following three categories depending on their location in the system:

- For more information on points (1 to 6, and 13) that need to be considered in a Web front-end system.
  For details, see *17.4.2 Setting the timeout in a Web front-end system*.
- Points (7 to 9) that need to be considered in the back system
  For details, see *8.6.3 Setting a timeout in the back-end system* in the manual *uCosminexus Application Server System Design Guide*.
- Points (10 to 12) that need to be considered during database connection
  This point needs to be further classified into a transaction timeout, DB Connector timeout, and database timeout. For details, see *8.6.4 Setting the transaction timeout* and *8.6.6 Setting the database timeout* in the *uCosminexus Application Server System Design Guide*.

For details on settings at the each point, see *17.4.3 Tuning parameters for setting the timeout*, or *8.6.8 Tuning parameters for setting the timeout* in the *uCosminexus Application Server System Design Guide*.

Reference note

The default values for each point are as follows:

Point	Default value
1	300 seconds
2	30 seconds
3	100 seconds
4	3,600 seconds
5	600 seconds
6	Not set. No timeout.
7	Not set. Continues to wait for response.
8	A value same as point 7 is automatically inherited and set up when the Enterprise Bean is invoked.
9	Not set. No timeout.
10	180 seconds
11	Differs according to the location of the timeout setup. • A timeout in establishing a physical connection: 8 seconds • A timeout in the request for connection during connection depletion: 30 seconds • A timeout in detecting a connection error: 5 seconds
12	Differs according to the type of database and the location of timeout setting. For HiRDB Unlock waiting timeout: 180 seconds Response timeout: 0 seconds (The HiRDB client continues to wait until there is a response from the HiRDB server) Request interval timeout: 600 seconds

Point	Default value
	For Oracle (when global transaction is used) Unlock waiting timeout: 60 seconds For SQL Server Timeout in acquiring memory: -1 (For details about the operations when -1 is specified, see the SQL Server documentation) Unlock waiting timeout: -1 (Continues to wait until the lock is released) For XDM/RD E2 Unlock waiting timeout: None (timeout is not monitored) CPU timeout during SQL execution: 10 seconds SQL execution timeout: 0 seconds (timeout is not monitored) Transaction timeout: 600 seconds Response timeout: 0 seconds (The HiRDB client continues to wait until there is a response from the XDM/RD E2 server)
13	600 seconds

17.4.2 Setting the timeout in a Web front-end system

This section explains the settings of timeout in a Web front-end system.

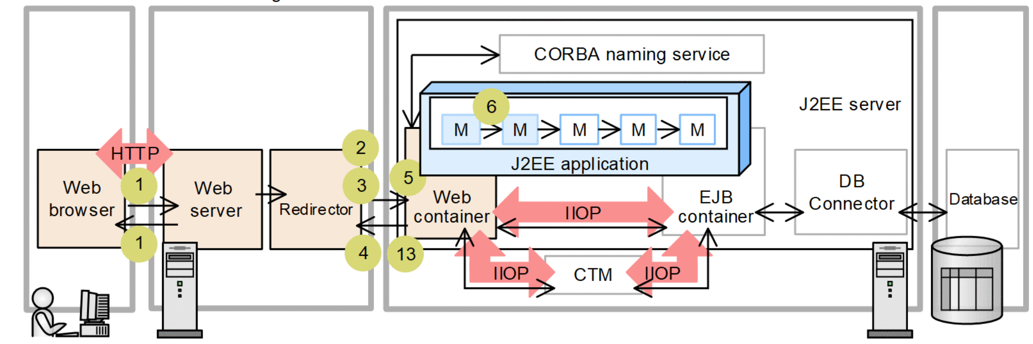
When setting the timeout in a Web front-end system, amongst all the timeout values for the entire system, you need to consider points 1 to 6 and 13 shown in the following figure. These numbers correspond to *Figure 17-2* or *Figure 17-3*.

Tip

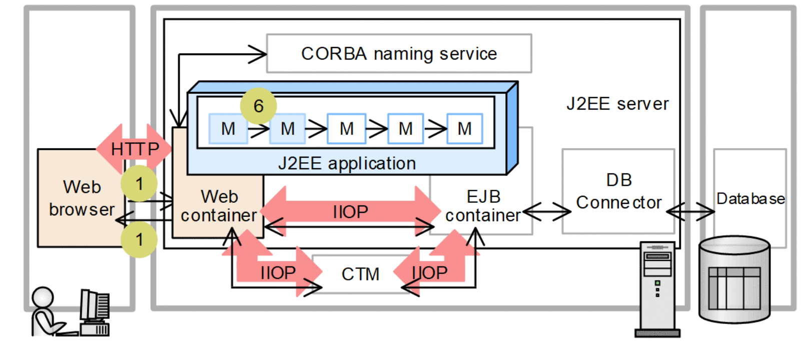
When you use an in-process HTTP server, you can set up points 1 and 6. The points 2 to 5, and 13 are not applicable.

Figure 17–4: The timeout points (points 1 to 6 and 13) to be considered in the case of a Web front-end system

●In the case of Web server integration



●If an in-process HTTP server is used



Legend:

▨	: Object to which a timeout is set in a Web front-end system
⟺	: HTTP or RMI-IIOP communication (IIOP: RMI-IIOP)
→	: Control flow related to timeouts
M	: Servlet or method in JSP

- **Waiting time in the Web server for receiving requests from the client and sending the data to the client (point 1)**

  When there is a backlog of requests from the Web browser, the redirector resources will be released according to the timeout. When there is a backlog of responses to the Web browser, (when the Web browser does not receive the responses), the resources of the Web container in the redirector and the J2EE server will be released according to the timeout.

  In the case of a Web server integration, the same values are set in the above waiting time settings.

  When you use an in-process HTTP server, you can specify different values for the waiting time for receiving requests from the client, and the waiting time for sending data to the client.

- **Waiting time for sending requests to the Web container of the redirector that is registered in the Web server (points 2 and 3)**

  When the redirector sends a request to the Web container and the control does not return due to an error in the Web container or an error in the communication path between the redirector and the Web container, the redirector resources will be released according to the timeout. At the same time, the error is notified to the Web browser. You can set the timeout at this point only in the case of Web server integration.

  Point 2 is the waiting time for establishing connection with the Web container and Point 3 is the waiting time for the process of sending requests to the Web container.

- **Waiting time for receiving data from the Web container of the redirector registered in the Web server (point 4)**

If there is an error in the J2EE application and the control does not return, the redirector resources will be released according to the timeout. At the same time, the error is notified to the Web browser. You can set the timeout at this point only in the case of Web server integration.

> **Tip**
>
> The unit for the settings is worker. Hitachi, therefore, recommends that when the processing time differs according to the business, define the worker for each Web application corresponding to the business and set the timeout value.

- **Waiting time for receiving data from the redirector, in the Web container (point 5)**
  When there is a backlog of requests from the browser, the J2EE server (Web container) resources will be released. You can set the timeout at this point only in the case of Web server integration.
- **Waiting time for processing request in the Web container (point 6)**
  The functionality of monitoring the execution time of J2EE application is used to set this timeout. For details, see *8.6.7 Setting the method timeout in the J2EE application* in the manual *uCosminexus Application Server System Design Guide*.
- **Waiting time for sending a response from the Web container to the redirector (point 13)**
  When the Web container sends a response to the redirector and the control does not return due to an error in the redirector or an error in the communication path between the redirector and the Web container, the Web container resources will be released according to the timeout. At the same time, the error is notified to the Web browser. You can set the timeout at this point only in the case of Web server integration.

17.4.3 Tuning parameters for setting the timeout

This section explains how to set up tuning parameters used for timeout settings.

(1) Timeout set in the Web server for receiving requests from the client and sending the data to the client

This is a tuning parameter for setting the timeout at point 1 of *Figure 17-2* or *Figure 17-3*. The location of setup differs according to the Web server used.

In the case of Web server integration, set the tuning parameter for each Web server. You edit the files for specifying the settings.

Table 17–6: Tuning parameters for the timeout to be set in the Web server for receiving requests from the client and sending the data to the client (for Web server integration)

Setup item	Location of setup
Timeout for receiving requests from the client and sending data to the client	`Timeout` directive of `httpsd.conf`

Note:
When you are using Microsoft IIS as the Web server, edit the `receive_client_timeout` key in `isapi_redirect.conf`.

For an in-process HTTP server, specify the settings in each J2EE server.

You set up the items listed in the following table using the Smart Composer functionality. Define the parameters in the Easy Setup definition file.

Table 17–7: Tuning parameters for the timeout to be set in the Web server for receiving requests from the client and sending the data to the client (for an in-process HTTP server)

Setup item	Setup target	Location of setup (parameter name)
Timeout for receiving requests from the client	Logical J2EE server (`j2ee-server`)	`webserver.connector.inprocess_http.receive_timeout`
Timeout for sending data to the client	Logical J2EE server (`j2ee-server`)	`webserver.connector.inprocess_http.send_timeout`

(2) Timeout set in the redirector for sending the data to the Web container

This is a tuning parameter for setting the timeout at point 2 and point 3 of *Figure 17-2*.The following table describes the tuning parameters for the timeout to be set in the redirector. You can specify the tuning parameter only in the case of Web server integration.

Specify the items listed in the following table with the Smart Composer functionality. You define the parameters in the Easy Setup definition file.

Table 17–8: Tuning parameters for the timeout to be set in the redirector

Point	Setup item	Setup target	Location of setup (parameter name)#
2	Connection timeout for Web container when sending requests	Logical Web server (`web-server`)	`JkConnectTimeout`
3	Timeout for sending requests	Logical Web server (`web-server`)	`JkSendTimeout`

#

When you are using Microsoft IIS as the Web server, edit the `connect_timeout` key in `isapi_redirect.conf`.

(3) Timeout set in the redirector for receiving the data from the Web container

This is a tuning parameter for setting the timeout at point 4 of *Figure 17-2*.

You set up the tuning parameters for each worker definition of the redirector. The following table describes the tuning parameters for the timeout to be set up in the redirector.

You specify the items listed in the following table using the Smart Composer functionality and define the parameters in the Easy Setup definition file.

Table 17–9: Tuning parameters for the timeout to be set in the redirector

Setup item	Setup target	Location of setup (parameter name)
Communication timeout of waiting for response data	Logical Web server (`web-server`)	`worker.`*`worker-name`*`.receive_timeout`

You can specify this tuning parameter only in the case of Web server integration.

(4) Timeout set in the Web container for receiving the data from the redirector

This is a tuning parameter for setting the timeout at point 5 of *Figure 17-2*.

You set up the tuning parameter for each J2EE server. The following table describes tuning parameters for the timeout to be set in the Web container.

You specify the items listed in the following table using the Smart Composer functionality and define the parameters in the Easy Setup definition file.

Table 17–10: Tuning parameters for the timeout to be set in the Web container

Setup item	Setup target	Location of setup (parameter name)
Timeout in waiting for reply from redirector	Logical J2EE server (`j2ee-server`)	`webserver.connector.ajp13.receive_timeout`

You can specify this tuning parameter only in the case of Web server integration.

(5) Timeout set in the Web container for receiving the data from the redirector

This is a tuning parameter for setting the timeout at the point 13 of *Figure 17-2*.

You set up the tuning parameter for each J2EE server. The following table describes the tuning parameters for the timeout to be set up in the Web container.

You specify the items listed in the following table using the Smart Composer functionality and define the parameters in the Easy Setup definition file.

Table 17–11: Tuning parameters for the timeout to be set in the Web container

Setup item	Setup target	Parameter name
Timeout of response sending process	Logical J2EE server (`j2ee-server`)	`webserver.connector.ajp13.send_timeout`

You can specify this tuning parameter only in the case of Web server integration.

17.5 Optimizing the operations of the Web application

This chapter explains how to tune the performance of the Web application. Determine the tuning in the case of a Web front-end system.

The following three types of tuning methods are explained below:

- Separate the deployment of the static contents and the Web application
- Caching static contents
- Distribute the requests by using the redirector (in the case of Web server integration)

17.5.1 Tuning parameters for optimizing the operations of the Web application

This section explains how to set up the tuning parameters used for optimizing the operations of a Web application.

(1) Tuning parameter for separating the deployment of the static contents and the Web application

Specify the separation of the deployment of the static contents and the Web application as the parameter of the file that defines the operations of the Web server. The setup locations, files and parameters differ depending on the type of Web server used.

When you use Cosminexus HTTP Server for a Web server integration, you use the redirector module for separation. When you use an in-process HTTP server, use the reverse proxy module deployed in the reverse proxy server (Cosminexus HTTP Server) for separation.

The following table explains the method and location of setup:

Table 17–12: Tuning parameters for separating the deployment of the static contents and the Web application

Web server to be used	Method of setup	Location of setup
Cosminexus HTTP Server (separation using the redirector module#1)	Smart Composer functionality	Definition file Easy Setup definition file Setup target Logical Web server (`web-server`) Parameter name `JkMount`
In-process HTTP server (separation using the reverse proxy module)	Edit file	Definition file `httpsd.conf` Setup target Reverse proxy server Parameter name `ProxyPass` directive#2

#1

Specify in `uriworkermap.properties`, if you are using Microsoft IIS as a Web server.

#2

For details about `httpsd.conf`, see the *uCosminexus Application Server HTTP Server User Guide*.

(2) Tuning parameters for caching static contents

The tuning parameters for cache of static contents are explained below. These tuning parameters are set for each Web container or Web application.

The following table describes how to set up the tuning parameters for each Web container. You specify these items using the Smart Composer functionality.

Table 17–13: Tuning parameter for caching static contents (items to be set for each Web container)

Setup items	Locations of setup
Select whether static contents cache is to be used	Definition file Easy Setup definition file Setup target Logical J2EE server (`j2ee-server`) Parameter name `webserver.static_content.cache.enabled`
Setup of maximum memory size for each Web application	Definition file Easy Setup definition file Setup target Logical J2EE server (`j2ee-server`) Parameter name `webserver.static_content.cache.size`
Setup of maximum file size of the static contents for cache	Definition file Easy Setup definition file Setup target Logical J2EE server (`j2ee-server`) Parameter name `webserver.static_content.cache.filesize.threshold`

The tuning parameters to be set for each Web application are described below. Set the items in the Web application either directly by editing `web.xml` or using the server management commands. Edit `web.xml` for setting the items in the Web application before deployment. You use the server management command (`cjsetappprop`) for setting the items in the Web application after deployment.

The following table describes the settings:

Table 17–14: Tuning parameters for caching of static contents (items to be set for each Web application)

Setup items	Settings#
Select whether to use the cache of static contents	`<param-name>` tag `com.hitachi.software.web.static_content.cache.enabled` `<param-value>` tag (Setup value)
Setup of maximum size of memory for each Web application	`<param-name>` tag `com.hitachi.software.web.static_content.cache.size`

Setup items	Settings#
	`<param-value>` tag (Setup value)
Setup of maximum file size of the static contents for caching	`<param-name>` tag `com.hitachi.software.web.static_content.cache.filesize.threshold` `<param-value>` tag (Setup value)

Note

For details about the values that can be specified in (setup value), see *2.19.2 Definitions in DD (Settings for each Web application)* in the *uCosminexus Application Server Web Container Functionality Guide*.

#

For directly editing `web.xml`, add `<context-param>` tag within the `<web-app>` tag and add the `<param-name>` and `<param-value>` tags within the `<context-param>` tag.

To use the server management commands, add `<context-param>` tag within the `<hitachi-war-property>` tag of the `WAR property` file and add `<context-param>` and `<param-value>` tags within the `<context-param>` tag.

(3) Tuning parameters for distributing the requests using a redirector

Specify the tuning parameters for distributing the requests using a redirector as the parameters of the file that defines the operations of the Web server.

You can specify this definition only when integrating with a Web server. You cannot specify this definition when you are using an in-process HTTP server.

The following table describes the method and location of setup:

Table 17–15: Tuning parameters for distributing the requests by using a redirector

Setup items	Method of setup	Location of setup
Mapping definition of URL pattern#	Smart Composer functionality	Definition file Easy Setup definition file Setup target Logical J2EE server (`j2ee-server`) Parameter name `JkMount`

#

When you are using Microsoft IIS as the Web server, specify in `uriworkermap.properties`.

17.6 Tuning other items

This section explains the tuning items other than those described till the previous section.

The following item is explained here:

- **Tuning of the Persistent Connection**

Determine the tuning of this item to use in-process HTTP servers in Web front-end systems.

In HTTP/1.1, a Persistent Connection is defined for the persistent use of the same TCP connection established between a Web client and Web server among multiple HTTP requests. Using a Persistent Connection, the time taken to establish a connection between a Web client and Web server can be shortened, and the communication traffic can be reduced.

However, the use of a Persistent Connection causes a specific Web client to occupy the request-processing threads, thereby leading to a decline in the processing performance of the entire server. Therefore, you are required to tune so as to be able to use the Persistent Connection effectively, and maintain the server processing performance.

When you use an in-process HTTP server, you can tune the following items of a Persistent Connection:

- **Upper-limit value of the number of Persistent Connections**
  When a TCP connection exceeds this upper-limit value, it gets disconnected after the completion of request processing. Therefore, a thread can be secured for processing a new connection that can prevent the request-processing threads from being occupied by a specific client.
- **Upper-limit value of the request-processing frequency of a Persistent Connection**
  Even when requests are received continuously from the same Web client, the TCP connection is disconnected once after the completion of request processing, if this upper-limit value is exceeded.
  This can prevent the request-processing threads from being occupied by a specific client.
- **Timeout of a Persistent Connection**
  You can set up timeouts for the request-waiting time of the Persistent Connection. If no processing request is received even after the lapse of the specified timeout period, the TCP connection will be disconnected. This can prevent the TCP connection from being occupied when it is not in use.

These items are specified as parameters of the Easy Setup definition file used with the Smart Composer functionality. The following table describes the tuning parameters to be set up for a Persistent Connection:

Table 17–16: Tuning parameters to be set up for a Persistent Connection

Setup item	Location of setup
Upper-limit value of the number of Persistent Connections	Definition file Easy Setup definition file Setup target Logical J2EE server (`j2ee-server`) Parameter name `webserver.connector.inprocess_http.persistent_connection.max_connections`
Upper-limit value of the request-processing frequency	Definition file Easy Setup definition file Setup target Logical J2EE server (`j2ee-server`)

Setup item	Location of setup
	Parameter name `webserver.connector.inprocess_http.persistent_connection.max_requests`
Timeout	Definition file Easy Setup definition file Setup target Logical J2EE server (`j2ee-server`) Parameter name `webserver.connector.inprocess_http.persistent_connection.timeout`

For details on each parameter, see *12.2 Parameters applicable to logical J2EE servers*.

17.7 TCP/UDP port numbers used by Application Server

When the port number is not explicitly defined, a port number is automatically assigned by Application Server to a port with the default value `(Floating)`.

The following table describes the TCP/UDP port numbers used by Application Server. Depending on the OS being used, the firewall setting may be at the host level instead of the network level. In the case of such firewalls, filtering takes place for all communication other than that with the localhost (127.0.0.1), including communication within the same host. In such cases, even for the ports that communicate only within the host, you need to set the filter settings so as to permit that communication.

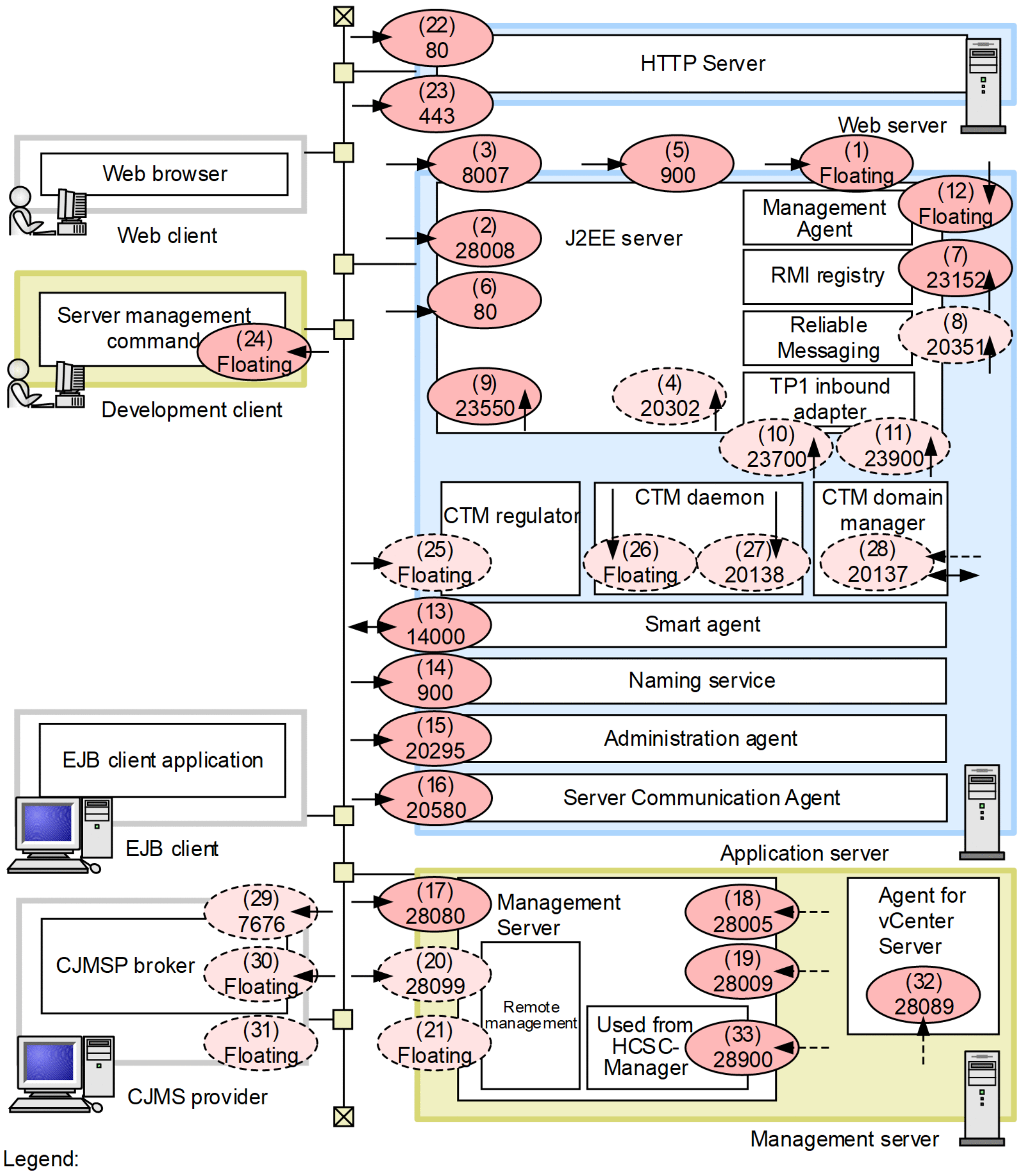
Table 17–17: TCP/UDP port numbers used by Application Server

No.	Processes	Explanation	Default values
(1)	J2EE server	Request reception port of an EJB container.	`(Floating)`
(2)		Management communication port.	`28008`
(3)		A port for request reception from the Web server (redirector).	`8007`
(4)		Transaction recovery processing communication port when using transaction services. Required when using transaction services.	`20302`
(5)		Request reception port of the Naming Service that is invoked by the in-process.	`900`
(6)		Request reception port of in-process HTTP server. Mandatory when using in-process HTTP server.	`80`
(7)		Request reception port of the RMI registry.	`23152`
(8)		A port for the event reception during application integration between multiple systems using shared queue.	`20351`
(9)		Request reception port when acquiring operation information.	`23550`
(10)		A port awaiting RPC request from OpenTP1.	`23700`
(11)		A port awaiting the synchronization request from OpenTP1.	`23900`
(12)	Management Agent	A port used for the communication of Management agent.	`(Floating)`
(13)	Smart agent	Port environment variable for communication of smart agent.	`14000`
(14)	Naming Service	Request reception port argument of Naming Service (use Cosminexus TPBroker).	`900`
(15)	Administration Agent	A port used by the Administration Agent in the communication with the Management Server.	`20295`
(16)	Server Communication Agent	A port used by Server Communication Agent for communicating with Virtual server manager.	`20580`
(17)	Management Server	An http port of the Management Server.	`28080`
(18)		A port for termination request of Management Server. Required for communication within the host.	`28005`
(19)		A port for internal communication of Management Server. Required for communication within the host.	`28009`
(20)		A port for connection to the Manager remote management function.	`28099`
(21)		A port for client connection to the Manager remote management function.	`(Floating)`

No.	Processes	Explanation	Default values
(22)	Cosminexus HTTP Server	An `http` port of Cosminexus HTTP Server.	`80`
(23)		An `https` port of Cosminexus HTTP Server.	`443`
(24)	Server management command	A port with which the server management command communicates with the J2EE server.	`(Floating)`
(25)	CTM regulator	Basic value of the port where CTM regulator receives requests from the EJB client. Use basic value + process count only. Mandatory when using CTM.	`(Floating)`
(26)	CTM daemon	Port where the CTM daemon receives the requests from the EJB client. Mandatory when using CTM.	`(Floating)`
(27)		Port for communication by CTM daemon with other daemon or J2EE server. Mandatory when using CTM.	`20138`
(28)	CTM domain manager	Port for communication by CTM domain manager with other CTM domain manager. Mandatory when using CTM, to communicate TCP and UDP (broadcast).	`20137`
(29)	CJMSP Broker	Port of broker of Cosminexus JMS provider for receiving requests from resource adapter or commands.	`7676`
(30)		Port of broker of Cosminexus JMS provider for establishing connection with resource adapter.	`(Floating)`
(31)		Port of broker of Cosminexus JMS provider for establishing connection with commands.	`(Floating)`
(32)	Management Server	Virtual server manager (Management Server) of 08-50 mode is a process port (Agent for vCenter Server), which runs internally to connect to vCenter Server.	`28089`
(33)		Internal communication port of Management Server used from HCSC-Manager.	`28900`

The following figure shows the TCP/UDP port numbers used by the Application Server process. (x) corresponds to the item numbers in the table:

Figure 17–5: TCP/UDP port numbers used by Application Server



Legend:

(X) YYYYY	: The port must be configured if a firewall is to be used. *YYYYY* is the default port number. (*X*) corresponds to a number used in the table.
(X) YYYYY (dashed)	: If a firewall is to be used, the port should be specified, depending on the functions to be used. *YYYYY* is the default port number. (*X*) corresponds to a number used in the table.
──►	: Indicates that a service is provided to an object outside the host via the port.
- - -►	: Indicates that internal communication occurs inside the host via the port.
◄──►	: Indicates that bidirectional communication occurs with an object outside the host via the port.

For other legend details, see *3.2 Description of the system configuration* in the manual *uCosminexus Application Server System Design Guide*.

The following table lists the locations for specifying port numbers. The item number in the table corresponds to the item number in the figure.

Table 17–18: Locations for specifying the TCP/UDP port numbers used in Application Server

No.	Definition files	Setup target	Parameter name#1
(1)	Easy Setup definition file	Logical J2EE server (j2ee-server)	vbroker.se.iiop_tp.scm.iiop_tp.listener.port
(2)	Easy Setup definition file	Logical J2EE server (j2ee-server)	ejbserver.http.port
(3)	Easy Setup definition file	Logical J2EE server (j2ee-server)	webserver.connector.ajp13.port
(4)	Easy Setup definition file	Logical J2EE server (j2ee-server)	ejbserver.distributedtx.recovery.port
(5)	Easy Setup definition file	Logical J2EE server (j2ee-server)	inprocess.ns.port
(6)	Easy Setup definition file	Logical J2EE server (j2ee-server)	webserver.connector.inprocess_http.port
(7)	Easy Setup definition file	Logical J2EE server (j2ee-server)	ejbserver.rmi.naming.port
(8)	Connector attribute file	Reliable Messaging	RMSHPort#2 specified in the <config-property> tag
(9)	Easy Setup definition file	Logical J2EE server (j2ee-server)	ejbserver.rmi.remote.listener.port
(10)	Connector property file	TP1 inbound adapter	scd_port#3 specified in <config-property> tag
(11)	Connector property file	TP1 inbound adapter	scd_port#4 specified in <config-property> tag#3
(12)	Easy Setup definition file	Logical J2EE server (j2ee-server)	mngagent.connector.port
(13)	Easy Setup definition file	Logical smart agent (smart-agent)	smartagent.port
(14)	Easy Setup definition file	Logical J2EE server (j2ee-server)	ejbserver.naming.port
(15)	Adminagent.properties	Administration Agent	adminagent.adapter.port key
(16)	sinaviagent.properties #4	Server Communication Agent	sinaviagent.port key
(17)	mserver.properties	Management Server	webserver.connector.http.port key
(18)	mserver.properties	Management Server	webserver.shutdown.port key
(19)	mserver.properties	Management Server	webserver.connector.ajp13.port key
(20)	mserver.properties	Management Server	com.cosminexus.mngsvr.management.port key
(21)	mserver.properties	Management Server	com.cosminexus.mngsvr.management.listen.port key
(22)	Easy Setup definition file	Logical Web server (web-server)	Listen

No.	Definition files	Setup target	Parameter name[#1]
(23)	Easy Setup definition file	Logical Web server (`web-server`)	`Listen`
(24)	`usrconf.properties` (system property file for server management command)	Server management command	`vbroker.se.iiop_tp.scm.iiop_tp.listener.port` key
(25)	Easy Setup definition file	Logical CTM (`component-transaction-monitor`)	`ctm.RegOption`
(26)	Easy Setup definition file	Logical CTM (`component-transaction-monitor`)	`ctm.EjbPort`
(27)	Easy Setup definition file	Logical CTM (`component-transaction-monitor`)	`ctm.port`
(28)	Easy Setup definition file	Logical CTM domain manager (`ctm-domain-manager`)	`cdm.port`
(29)	`config.properties`	CJMSP Broker	`imq.portmapper.port` key
(30)	`config.properties`	CJMSP Broker	`imq.jms.tcp.port` key
(31)	`config.properties`	CJMSP Broker	`imq.admin.tcp.port` key
(32)	vmx.properties	Virtual server manager of 08-50 mode (Management Server)	`vmx.vcenterserver.agent.port` key
(33)	`mserver.properties`	Management Server	`ejbserver.naming.port` key

#1

If the setup file is an Easy Setup definition file, specify the value specified in `<param-name>` in the `<configuration>` tag.

#2

`RMSHPort` is a configuration property specified in the property definition of the resource adapter Cosminexus RM. For details about `RMSHPort`, see *6. Configuration Properties* in the manual *Cosminexus Reliable Messaging*.

#3

`scd_port` and `trn_port` are the configuration properties specified in the property definition of the resource adapter TP1 inbound adapter. For details about `scd_port` and `trn_port`, see *4.12.2 Setting up a resource adapter* in the *uCosminexus Application Server Common Container Functionality Guide*.

#4

For details about Server Communication Agent, see the documents related to Server Communication Agent.

Reference note

The following table describes the locations for specifying TCP/UDP port numbers when setting up Application Server:

Table 17–19: Locations for specifying TCP/UDP port numbers when setting up Application Server

No.	Setup location when setting up by editing the file
(1)	`vbroker.se.iiop_tp.scm.iiop_tp.listener.port` key in `usrconf.properties`
(2)	`ejbserver.http.port` key in `usrconf.properties`
(3)	`webserver.connector.ajp13.port` key in `usrconf.properties`

No.	Setup location when setting up by editing the file
(4)	`ejbserver.distributedtx.recovery.port` key in `usrconf.properties`
(5)	`ejbserver.naming.port` key in `usrconf.properties`
(6)	`webserver.connector.inprocess_http.port` key of `usrconf.properties`
(7)	`ejbserver.rmi.naming.port` key in `usrconf.properties`
(8)	`RMSHPort`#1 specified in the `<config-property>` tag of the Connector property file
(9)	`ejbserver.rmi.remote.listener.port` key of `usrconf.properties`
(10)	`scd_port` specified in the `<config-property>` tag of the resource adapter of Connector property file of TP1 inbound adapter
(11)	`trn_port` specified in the `<config-property>` tag of the resource adapter of Connector property file of TP1 inbound adapter
(12)	`mngagent.connector.port` key in the `mngagent.`*real-server-name*`.properties` file
(13)	Environment variable *OSAGENT_PORT*
(14)	• When auto-starting CORBA Naming Service in in-process or out-process `ejbserver.naming.port` key of `usrconf.properties` • When manually starting CORBA Naming Service Specify `-Dvbroker.se.iiop_tp.scm.iiop_tp.listener.port=`*port-number* in the command argument of the `nameserv` command.
(15)	`adminagent.adapter.port` key in `adminagent.properties`
(16)	`sinaviagent.port` key in `sinaviagent.properties`#2
(17)	`webserver.connector.http.port` key in `mserver.properties`
(18)	`webserver.shutdown.port` key in `mserver.properties`
(19)	`webserver.connector.ajp13.port` key in `mserver.properties`
(20)	`com.cosminexus.mngsvr.management.port` key in `mserver.properties`
(21)	`com.cosminexus.mngsvr.management.listen.port` key in `mserver.properties`
(22)	`Listen` directive or `Port` directive of `httpsd.conf`
(23)	`Listen` directive or `Port` directive of `httpsd.conf`
(24)	`vbroker.se.iiop_tp.scm.iiop_tp.listener.port` key in `usrconf.properties` (system properties file for server management commands)
(25)	Argument of the `ctmregltd` command or `ctmstart` command - `CTMEjbPort`
(26)	Argument of the `ctmstart` command - `CTMEjbPort`
(27)	Argument of the `ctmstart` command - `CTMPort`
(28)	Argument of the `ctmdmstart` command - `CTMPort`.
(29)	`imq.portmapper.port` key in `config.properties`
(30)	`imq.jms.tcp.port` key in `config.properties`
(31)	`imq.admin.tcp.port` key in `config.properties`
(32)	`vmx.vcenterserver.agent.port` key in `vmx.properties`

No.	Setup location when setting up by editing the file
(33)	`ejbserver.naming.port` key in `mserver.properties`

#1

`RMSHPort` is a configuration property specified in the property definition of the resource adapter Cosminexus RM. For details about `RMSHPort`, see *6. Configuration Properties* in the manual *Cosminexus Reliable Messaging*.

#2

For the details about Server Communication Agent, see the documents related to Server Communication Agent.

Important note

Notes on the standby port for a server (In UNIX)

In UNIX, when all the following conditions are satisfied, a connection might be successfully established with a TCP port that is not in a standby status:

- An attempt is made to establish a connection with a port that is not in a standby status
- The host itself is the connection target, and the port is in the range of the temporary port numbers (the range of port numbers that are dynamically allocated by the OS)

When this event occurs, the assumed process communication cannot be executed and a timeout occurs. To avoid this event, specify a value outside the range of the temporary port numbers as the standby port of the server. You can check the range of the temporary port numbers in the following files:

In AIX

Minimum value (`32768`): `no -o tcp_ephemeral_low`

Maximum value (`65535`): `no -o tcp_ephemeral_high`

In Linux

`/proc/sys/net/ipv4/ip_local_port_range`

For details on how to set up the standby port of a server, see the documentation for the OSs.

18 Functionality Compatible with the Basic and Development Functionality (Connecting a Database by Using DABroker Library) (INTENTIONALLY DELETED)

(INTENTIONALLY DELETED)

18.1 (INTENTIONALLY DELETED)

(INTENTIONALLY DELETED)

19 Functionality Compatible with the Basic and Development Functionality (Using Annotations in EJB 2.1 and Servlet 2.4) (INTENTIONALLY DELETED)

(INTENTIONALLY DELETED)

19.1 (INTENTIONALLY DELETED)

(INTENTIONALLY DELETED)

20 Settings for Using the Connection Pool Clustering Functionality

This chapter describes the connection pool clustering functionality.

20.1 Functionality for connection pool clustering

This section describes the connection pool clustering functionality.

The following table describes the organization of this section.

Table 20–1: Organization of this section (Connection pool clustering functionality)

Category	Title	Reference location
Description	Connecting to Oracle using Oracle RAC	*20.1.1*
	Overview of connection pool clustering	*20.1.2*
	Resource adapters used	*20.1.3*
	Connection pool clustering operations	*20.1.4*
	Procedure for stopping or starting a connection pool manually	*20.1.5*
Settings	Settings required for clustering a connection pool	*20.1.6*

Note:

There is no specific description of *Implementation* and *Operations* for this functionality.

The connection pool clustering functionality optimizes the operations in a system where the database is used in a cluster configuration. You can use this functionality when you connect to Oracle using Oracle RAC. By using the connection pool clustering functionality, you can prevent the drop in system availability during errors and during maintenance. The following points describe the operations when an error occurs in the database node and when maintenance is performed for the database node while you are using the connection pool clustering functionality.

- **When an error occurs in the database node**

  When a connection cannot be obtained, such as during an OS, hardware, or software error, you can automatically suspend the connection pool connected to the database node where the error occurred (auto-suspension functionality). Even when a connection request is sent from the J2EE application to the resource adapter, no connection request is sent to the suspended connection pool, so the processing is not cancelled until a TCP/IP timeout occurs. This enables the J2EE application to continue business by obtaining a connection from a connection pool connected to another normal database node.

  Also, when the database node error is recovered, you can automatically restart the connection pool (auto-restart functionality). If the connection pool is restarted, the automatically recovered database node is accessed again, so you need not execute the `cjclearpool` command to delete the connection pool for recovering the database node.

- **When maintenance is performed for the database node**

  When you perform maintenance for a database node, you can use commands to suspend the member connection pool at any time (manual suspension functionality). Due to this, you can separate that database node and perform maintenance.

  Also, when you restart the database node after maintenance finishes, you can use commands to restart the connection pool at any time (manual restart functionality).

Note that in the connection pool clustering functionality, when a failure is detected while a connection is being obtained, the connection will be obtained from another normal member resource adapter. At this time, no error occurs in the application.

This section describes how to connect to Oracle clustered by using the Oracle RAC functionality, and the features and functionality of a connection pool when the connection pool is clustered (connection pool clustering).

20.1.1 Connecting to Oracle using Oracle RAC

The method of connecting to Oracle using Oracle RAC differs depending on the Oracle version or the functionality used for load balancing. Note that the connectable transaction type is a local transaction.

For details on compatibility with the Oracle version, functionality used for load balancing, and RAR files to use, see the manual *uCosminexus Application Server Common Container Functionality Guide*.

This section describes the Oracle connection method for each functionality used for load balancing.

(1) Connections using the Application Server functionality

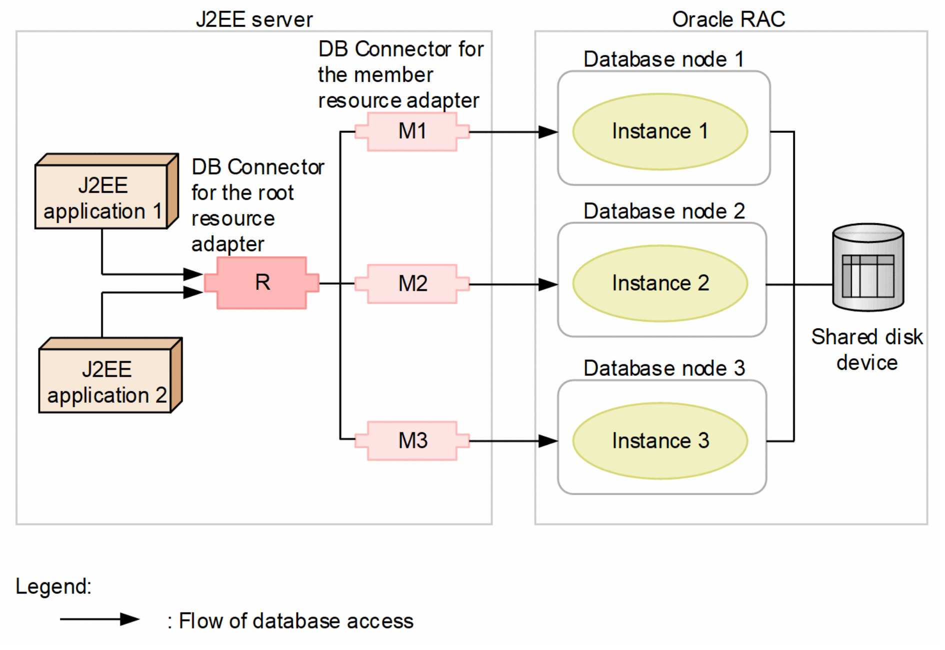
You use the connection pool clustering functionality of Application Server to connect to Oracle RAC. Application Server distributes the load of database access.

For details on the connection pool clustering functionality, see the description in section *20.1.2* and later.

(a) Load balancing procedure and settings

The following figure shows the load balancing procedure and settings when you use the connection pool clustering functionality.

Figure 20–1: Connection using the connection pool clustering functionality



This figure shows an Oracle RAC system with a 3-node configuration where the database node 1 contains instance 1, database node 2 contains instance 2, and database node 3 contains instance 3. The settings for connecting to the database are as follows:

1. Generate DB Connector M1, M2, and M3 for the member resource adapters mapped to instances 1, 2, and 3. Also, generate DB Connector R for the root resource adapter that contains the functionality for distribution to the member resource adapters.
2. Associate the J2EE applications with DB Connector R for the root resource adapter.

With these settings, the database access from the J2EE applications 1 and 2 is distributed to the database nodes 1, 2, and 3.

(b) Operations during database errors and recovery

When a database error occurs, Application Server detects the error. The member resource adapter mapped to the database where the error occurred is blocked and the processing is continued with the remaining instances.

When the database error is recovered, Application Server automatically releases the blockade. You can also release the blockade manually.

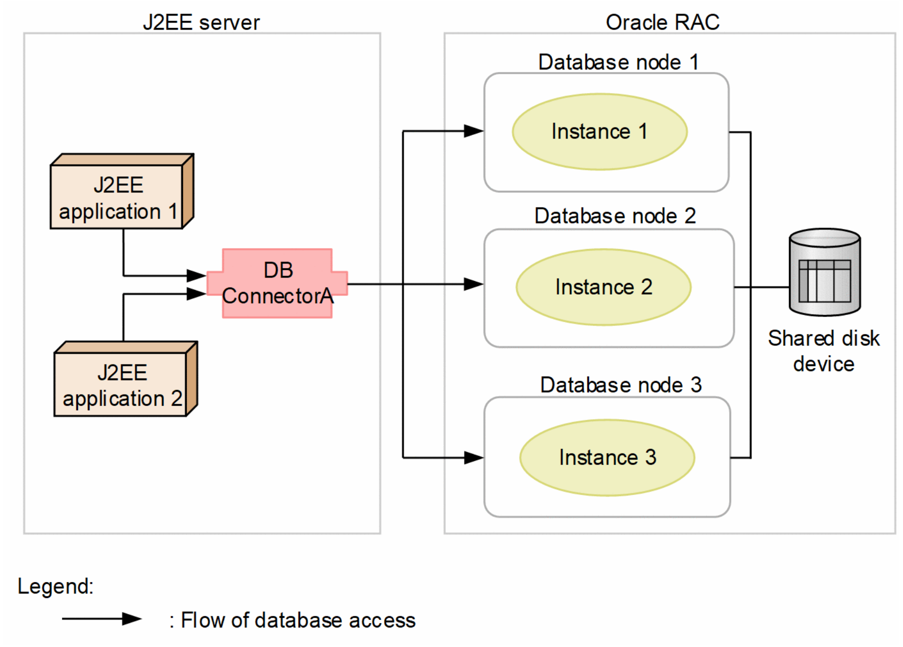
(2) Connections using the Oracle functionality

You connect to Oracle RAC from DB Connector and distribute the database access load with the Oracle RAC functionality.

(a) Load balancing procedure and settings

The following figure shows the load balancing procedure and settings when you use the Oracle RAC functionality.

Figure 20–2: Connection using the Oracle RAC functionality



This figure shows an Oracle RAC system with a 3-node configuration where the database node 1 contains instance 1, database node 2 contains instance 2, and database node 3 contains instance 3. The settings for connecting to the database are as follows:

1. Generate DB Connector A that distributes and connects to each instance.
2. Set up DB Connector A to use a global database name or service name.
3. Associate J2EE application 1 and 2 with DB Connector A.

With these settings, the database access from the J2EE applications 1 and 2 is distributed to the database nodes 1, 2, and 3.

(b) Operations during database errors and recovery

When a database error occurs, the Oracle RAC functionality separates the instance where the error occurred and the processing is continued with the remaining instances.

If you are using an Application Server connection pool, execute one of the following operations during database recovery. The connection pool is cleared and the subsequent access is distributed normally.

- Execute the `cjclearpool` command.
- Restart the J2EE server.

20.1.2 Overview of connection pool clustering

A connection pool that is clustered is called *connection pool clustering*. This section describes the connection pool clustering configuration and the functionality available with connection pool clustering.

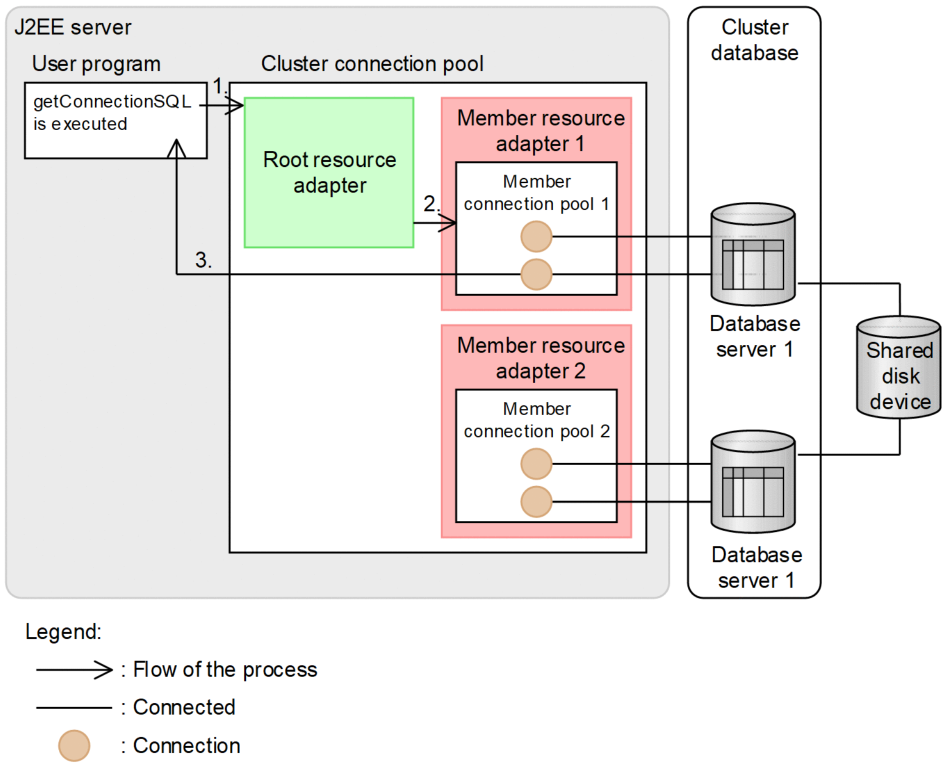
(1) Connection pool clustering configuration

The member resource adapters connected to each database node and the root resource adapter that bundles these multiple member resource adapters together configure a clustered connection pool. A description of the root resource adapter and member resource adapter is as follows:

- **Root resource adapter**
  This resource adapter is accessed from the J2EE applications when a clustered connection pool (connection pool clustering) is used. The root resource adapter bundles the member resource adapters together. With the root resource adapter, the processing requests to the root resource adapter are distributed to the member resource adapters. Note that the root resource adapter does not have a connection pool.
- **Member resource adapter**
  This resource adapter is connected to clustered individual database nodes. A member resource adapter is necessarily accessed via the root resource adapter. The connection pool of a member resource adapter is called a *member connection pool*.

The following figure shows the flow of processing for obtaining a connection when the connection pool is clustered.

Figure 20–3: Flow of processing for obtaining a connection



1. The J2EE application makes a connection request to the root resource adapter.

2. The root resource adapter selects one member connection pool and sends the connection request.

3. A connection is selected from the member connection pool and returned to the J2EE application.

(2) Preconditions

A database can use the connection pool clustering functionality only while the Oracle RAC functionality is used. The JDBC driver that can use the connection pool clustering functionality is Oracle JDBC Thin Driver.

(3) J2EE components and functionality available for database connections

The following table describes the J2EE components and functionality available for database connections when the connection pool clustering functionality is used.

Table 20–2: J2EE components and functionality available for database connections (Connection pool clustering functionality)

Items		Oracle (when the connection pool clustering functionality is used)
J2EE components	Servlet/JSP	Y
	Stateless Session Bean	Y
	Stateful Session Bean	Y
	Singleton Session Bean	Y
	Entity Bean (BMP)	Y

Items		Oracle (when the connection pool clustering functionality is used)
	Entity Bean (CMP 1.1)	N
	Entity Bean (CMP 2.0)	N
	Message-driven Bean	N
Available functionality	Connection pooling	Y
	Connection pool warming up	Y
	Displaying the connection pool information (`cjlistpool` command)	Y
	Clearing the connection pool	Y
	Connection test for resources	Y
	Detecting the connection errors	Y
	Statement pooling	Y
	Statement cancellation	Y#
	Statement `setQueryTimeout` method	Y#
	PRF trace output for the connection ID	Y
	Output of the SQL statement for troubleshooting	Y

Legend:

Y: Available

N: Not available

#

Precautions need to be taken when you connect to Oracle. For details on the precautions, see the manual *uCosminexus Application Server Common Container Functionality Guide*.

Important note

To use a resource adapter, you must resolve the references from the J2EE application to the resource adapter. When you customize a J2EE application that uses the resource adapters, resolve the references from the J2EE application to the resource adapter.

(4) Available resource connection and transaction management functionality

For details on the functionality that can be used by root resource adapters and member resource adapters, see the manual *uCosminexus Application Server Common Container Functionality Guide*.

20.1.3 Resource adapters used

This section describes the commands that can be executed with the root resource adapter and member resource adapter, gives an overview of the settings, and describes the notes.

(1) Executable commands

The following table describes the executability of commands for the root resource adapter and member resource adapter. Note that the commands other than those listed in the table can be executed with any resource adapter.

Table 20–3: Executability of commands in root resource adapter and member resource adapter

Command	Types of resource adapters	
	Root resource adapter	Member resource adapter
`cjstartrar`	Y	Y
`cjstoprar`	Y	Y
`cjtestres`	Y	Y
`cjlistrar`	Y	Y
`cjclearpool`	N	Y
`cjlistpool`	N	Y
`cjsuspendpool`	N	Y
`cjresumepool`	N	Y

Legend:

Y: Can be executed

N: Cannot be executed

(2) DB Connector settings

The following table describes the items specified in DB Connector for the root resource adapter and member resource adapter.

Table 20–4: Items specified in DB Connector for the root resource adapter and member resource adapter

Settings	Types of resource adapters	
	Root resource adapter	Member resource adapter
Transaction support level	N	Y[#1]
Can the log be collected	Y	Y
Database connection information	--	Y
DB Connector-specific settings (such as the statement pool)	--	Y
Security information (user name and password)	N	Y[#2]
Connection pool size	N	Y
Connection pool clustering-specific settings	Y	--

Legend:

Y: Settings must be specified

N: Settings need not be specified

--: There are no settings

#1

The transaction support levels of all the member resource adapters configuring one clustered connection pool must be the same.

#2

The user names of all the member resource adapters configuring one clustered connection pool must be the same.

For details on the DB Connector settings, see *4.2 Settings for connecting to the database* in the *uCosminexus Application Server Application Setup Guide*, or *20.6 HITACHI Connector Property file*.

(3) Notes on the root resource adapters

- You cannot execute the `cjclearpool` command. If the command is executed, a message is output and the command terminates abnormally. To clear the connections in a clustered connection pool with the `cjclearpool` command, execute the command for the member resource adapter.
- You cannot execute the `cjlistpool` command. If the command is executed, a message is output and the command terminates abnormally. If you want to display the information of a connection pool within the clustered connection pool, execute the `cjlistpool` command by specifying the member resource adapter display name in `-resname`, or by specifying `-resall`.
- You cannot mix and use a container-managed sign-on and component-managed sign-on in one clustered connection pool. When you use a component-managed sign-on, leave the user names of all the member resource adapters belonging to the root resource adapter blank.
- You can start a root resource adapter when all the member resource adapters belonging to the root resource adapter are running. If you start the root resource adapter in the other cases, an error message is displayed on the console or on the screen, and the resource adapter fails to start.

(4) Notes on the member resource adapters

- With the member resource adapters, the prerequisite functionalities are as follows. These functionalities are enabled by default and cannot be disabled.
    - Connection pooling

      Specify a value greater than 0 as the maximum connection pool value. If 0 or less is specified, the operations are performed assuming that the default values are specified for the maximum and minimum connection pool values.
    - Detecting the connection errors when connections are obtained

      Detecting the connection errors when connections are obtained and timeout for detecting the connection errors are enabled regardless of the set values.
    - Waiting for a connection when connections deplete

      Waiting for a connection when connections deplete is enabled regardless of the set value.
    - loginTimeout

      Specify a value greater than 0 in the `loginTimeout` property. If 0 or less is specified, the operations are performed assuming that the default value is specified.
- The following functionalities cannot be used with the member resource adapters. These functionalities are disabled by default and cannot be enabled.
    - Retrying to obtain a connection
    - User-specified Namespace for the J2EE resources
- All the connections in the member connection pool must be connected to the same database node; therefore, do not use the functionality to change the connection destination for a database. The examples of these functionalities are as follows. For the settings on disabling the functionality, see the Oracle documentation.
    - Client load balancing functionality
    - Connection failover functionality

    - Database service
    - Listener-based load balancing functionality
- The member resource adapters are necessarily accessed via the root resource adapter. Therefore, you cannot specify the member resource adapters in the following locations:
    - Resource references of the J2EE applications
    - Mapping definitions of the CMP Entity Bean
- You can stop a member resource adapter when the root resource adapter to which the member resource adapter belongs is stopped. If you stop a member resource adapter when the root resource adapter is running, an error message is displayed on the console or on the screen, and the resource adapter fails to stop.
- When a member connection pool is blocked and suspended, to destroy the connections the unused connections are removed from the connection pool. At this time, if you restart the connection pool when the destruction of connections is not yet complete, the total number of connections in the connection pool and the unused connections removed from the connection pool might temporarily exceed the maximum number of connections in the connection pool.

20.1.4 Connection pool clustering operations

This section describes the available functionality and operations when a connection pool is clustered. You can execute the following functionality with a clustered connection pool:

- Suspending a connection pool
- Restarting a connection pool

This section also describes the connection pool states and how to select a connection pool when a connection request is received.

(1) Suspending a connection pool

You can block and suspend a member connection pool. If you execute suspend, a member connection pool is blocked and suspended.

When a J2EE application sends a connection request to the root resource adapter, no connection request is sent to a blocked or suspended member connection pool.

You suspend the member connection pool in the following cases:

- When an error occurs in a database node
- When maintenance is performed for a database node

Note that the suspension methods include the following two methods:

- Auto-suspension
- Manual suspension

> **Reference note**
>
> With the suspension processing, the connections that the J2EE application has finished using are destroyed, all the connections in the connection pool are destroyed, and then the connection pool is suspended. When

a network error or a database node error occurs, a connection is destroyed after waiting for the network to timeout, so the processing from blocking to suspending the connection pool might take time.

(a) Auto-suspension

You can automatically suspend the member connection pool when a database node error occurs. When an error is detected, the member connection pool is automatically blocked and then suspended.

If the following events occur when a connection is obtained, a database node error is determined, and the member connection pool is automatically suspended:

- When a timeout occurs during the detection of the connection errors while a connection is being obtained
- When a physical connection cannot be obtained
- When the obtaining of a physical connection times out
- When the connection management threads deplete

This functionality is enabled by default. Specify the settings for enabling or disabling this functionality as a root resource adapter property. For details on the resource adapter settings, see *5.4 Defining resource adapter properties* in the *uCosminexus Application Server Application Setup Guide*.

(b) Manual suspension

When you perform operations such as database node maintenance, you execute the `cjsuspendpool` command to manually suspend a connection pool. You can execute manual suspension when the status of a member connection pool is Start, Reserved Start, Auto-suspend, and Reserved Auto-suspend. If you execute the `cjsuspendpool` command when the member connection pool is in an auto-suspension state, the status changes to manual suspension. Due to this, you can execute operations so that the member connection pool is not restarted automatically after auto-suspension. For details on the manual suspension procedure, see *20.1.5(2) Suspending the connection pool*. For command details, see *cjsuspendpool (suspend member connection pool)*.

Important note

A connection pool that is suspended manually with the `cjsuspendpool` command is not restarted automatically. Restart the connection pool manually.

(2) Restarting a connection pool

You can restart a suspended member connection pool. When the J2EE application makes a connection request to the root resource adapter, the connection requests are now sent to the restarted member connection pool again.

Note that to restart a connection pool, the number of unused connection management threads must be equal to or more than the maximum number of connections in the connection pool.

You restart the connection pool in the following cases:

- When the database node error is recovered
- When the maintenance of the database node finishes

The restart methods include the following two methods:

- Auto-restart

- Manual restart

> **Reference note**
>
> - If the number of unused connection management threads becomes equal to the maximum number of connections in the connection pool when the connection pool is stopped, a message is output. When the execution of the `cjresumepool` command fails, check the displayed message and then re-execute the command.
> - When the connection pool warming up functionality is enabled, the member connection pool starts after the minimum number of connections defined in the connection pool settings is pooled. If an attempt to generate a connection fails while the connections are being pooled, the member connection pool returns to the suspended state. Also, if the connection pool warming up functionality is disabled, the member connection pool starts immediately.

(a) Auto-restart

You can automatically restart an auto-suspended member connection pool.

The auto-suspended member connection pool sends a physical connection request at regular intervals to check the state of the database node. At this time, if a connection is obtained successfully, the database node is determined to have recovered, and the restart processing is performed automatically. Also, if you do not want to automatically restart an auto-suspended member connection pool, manually suspend the member connection pool after the pool is suspended automatically. To restart the pool, use manual restart.

Note that if the root resource adapter is stopped, the auto-restart processing is not performed. However, if you stop the root resource adapter while the member connection pool is auto-restarting, the running auto-restart processing is continued.

This functionality is enabled by default. Specify the settings for switching between enabling and disabling and for the interval to check the database node state as the root resource adapter properties. For details on the resource adapter settings, see *5.4 Defining resource adapter properties* in the *uCosminexus Application Server Application Setup Guide*.

(b) Manual restart

You can manually restart an auto-suspended or manually suspended connection pool. To restart manually, use the `cjresumepool` command. You can execute manual restart when the status of the connection pool is as follows:

- Auto-suspension
- Manual suspension
- Reserved auto-suspension
- Reserved manual suspension

For details on the manual restart procedure, see *20.1.5(3) Restarting the connection pool*. For the command details, see *cjresumepool (restart member connection pool)*.

(3) States of the connection pool

A connection pool state only exists for the member connection pools. Note that the connection pool state is maintained even after you restart the J2EE server and resource adapter.

You can check the state of the member connection pool with the following methods. When you execute manual suspension or manual restart, check the connection pool state before you execute the commands.

- `cjlistrar` command

  The `cjlistrar` command outputs the names and states of all the deployed resource adapters to the standard output. If you specify `-clusterpool`, you can also display the states of the member connection pools.

  For the command details, see *cjlistrar (list resource adapters)* in the *uCosminexus Application Server Command Reference Guide*.

- `cjlistpool` command

  The `cjlistpool` command displays the connection pool information. The command can also display the states of the member connection pools for the member resource adapters. For details on examples of displaying the connection information of a member resource adapter, see the manual *uCosminexus Application Server Common Container Functionality Guide*.

  For the command details, see *cjlistpool (list connection pools)* in the *uCosminexus Application Server Command Reference Guide*.

This section describes the states of the member connection pools.

The state of the member connection pool is maintained even if you restart the J2EE server and member resource adapter. For the procedure for checking the connection pool states, see *20.1.5(1) Checking the connection pool state*.

Note that the connection pools other than those for the member resource adapters do not have a state.

(a) Transition of the connection pool state

The following figure shows the transition of the member connection pool state.

Figure 20–4: Transition of the member connection pool state (when the pool is suspended manually)

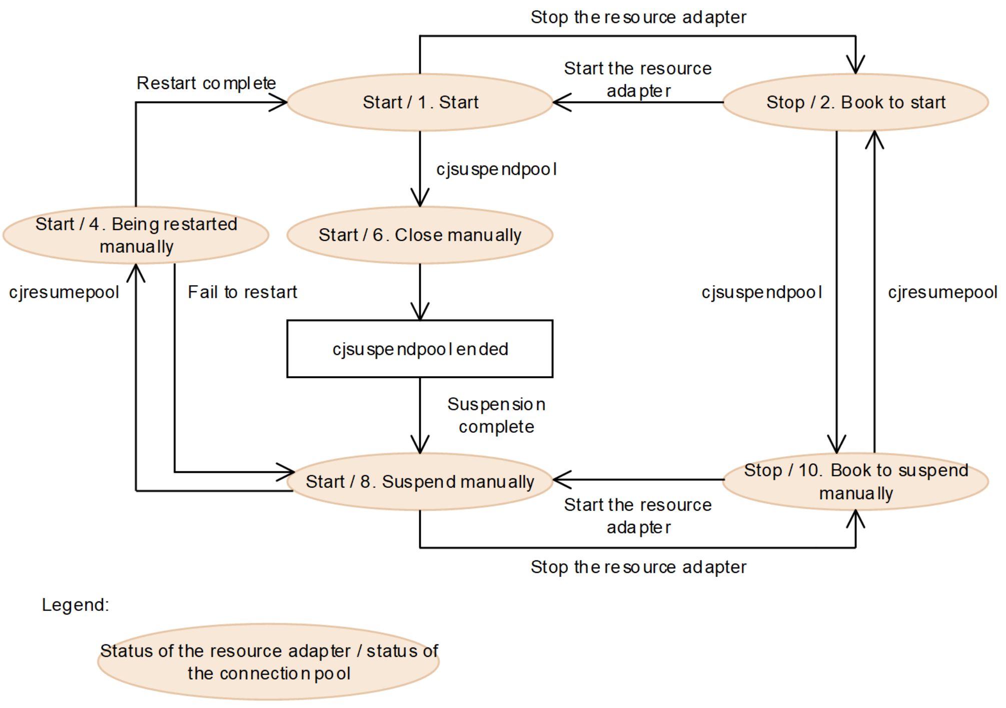
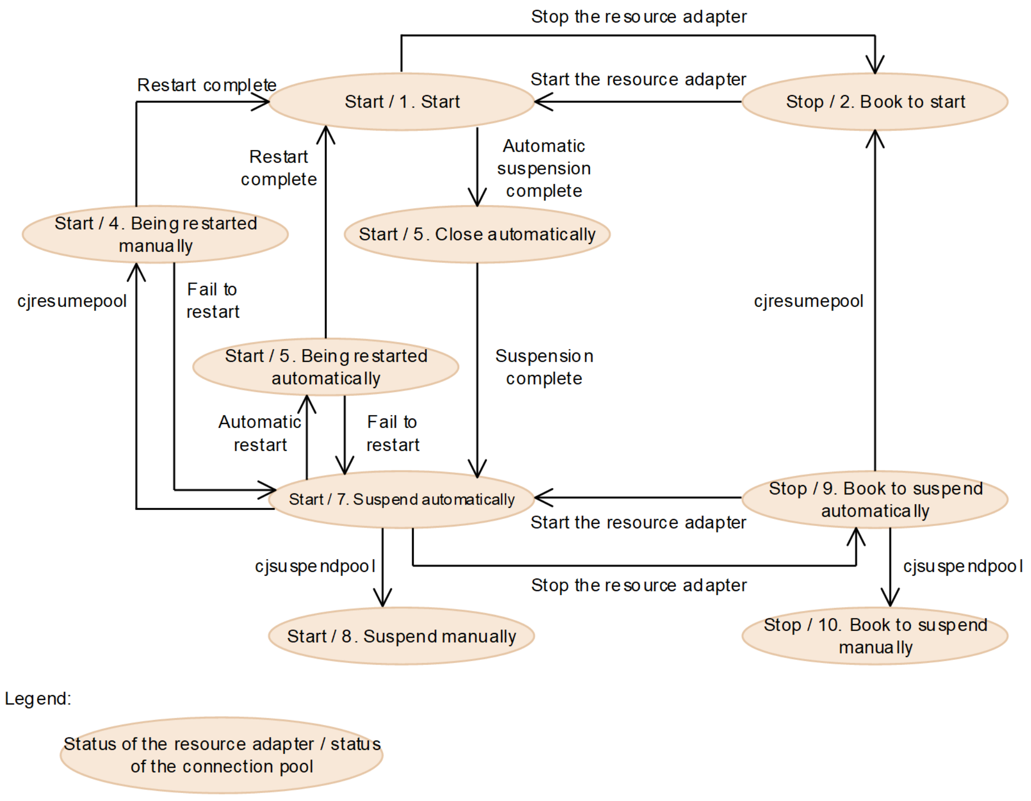

Stop the resource adapter
Restart complete
Start the resource adapter
Start / 1. Start
Stop / 2. Book to start
cjsuspendpool
Start / 4. Being restarted manually
Start / 6. Close manually
cjresumepool
Fail to restart
cjsuspendpool
cjresumepool
cjsuspendpool ended
Suspension complete
Start / 8. Suspend manually
Stop / 10. Book to suspend manually
Start the resource adapter
Stop the resource adapter
Legend:
Status of the resource adapter / status of the connection pool

Figure 20–5: Transition of the member connection pool state (when the pool is suspended automatically)


Stop the resource adapter
Restart complete
Start / 1. Start
Start the resource adapter
Stop / 2. Book to start
Restart complete
Automatic suspension complete
Start / 4. Being restarted manually
Start / 5. Close automatically
cjresumepool
Fail to restart
cjresumepool
Start / 5. Being restarted automatically
Suspension complete
Automatic restart
Fail to restart
Start / 7. Suspend automatically
Stop / 9. Book to suspend automatically
Start the resource adapter
cjsuspendpool
cjsuspendpool
Stop the resource adapter
Start / 8. Suspend manually
Stop / 10. Book to suspend manually
Legend:
Status of the resource adapter / status of the connection pool

Note that when the J2EE server starts, if the status of the connection pool is Resuming or Blocked, the connection pool status transits to Suspended.

The following table describes each status.

Table 20–5: Status of the member connection pools

No.	Status	Status displayed in the message	Explanation
1	Start status	`running`	A state in which the connection pool receives processing. When a connection request is made to the root resource adapter, the processing is only performed for a started connection pool.
2	Reserved Start status	`runningReserved`	A state in which the resource adapter is stopped when the connection pool has the Start status. The status of a connection pool with the Reserved Start status changes to the Start status when you start the resource adapter. This status is reached immediately after the resource adapter is deployed.
3	Auto-restarting status	`resumingAutomatically`	A state in which the connection pool performs the restart processing using the auto-restart functionality. If the restart processing is successful, the status changes to the Start status.

No.	Status	Status displayed in the message	Explanation
			If the restart processing fails, the status returns to the Auto-suspension status. No request is sent to a connection pool with the Auto-restarting status when a connection request is sent to the root resource adapter.
4	Manual Restarting status	• `resumingManuallyFromSuspendedAutomatically` • `resumingManuallyFromSuspendedManually` • `resumingManually`	A state in which the connection pool performs the restart processing by executing the `cjresumepool` command in the Auto-suspension status or Manual Suspension status. If the restart processing is successful, the status changes to the Start status. If the restart processing fails, the status returns to the one before the command was executed. No request is sent to a connection pool with the Manual Restarting status when a connection request is sent to the root resource adapter.
5	Automatically Blocked status	`blockedAutomatically`	A state in which the auto-suspension functionality is used to block a connection pool and perform the suspension processing. When the suspension processing finishes, the status changes to the Auto-suspension status. No request is sent to a connection pool with the Automatically Blocked status when a connection request is sent to the root resource adapter.
6	Manually Blocked status	`blockedManually`	A state in which the `cjsuspendpool` command is used to block a connection pool and perform the suspension processing. When the suspension processing finishes, the status changes to the Manual Suspension status. No request is sent to a connection pool with the Manually Blocked status when a connection request is sent to the root resource adapter.
7	Auto-suspension status	`suspendedAutomatically`	A state in which the auto-suspension functionality is used to suspend the connection pool. There are no connections in the connection pool. No request is sent to a connection pool with the Auto-suspension status when a connection request is sent to the root resource adapter.
8	Manual Suspension status	`suspendedManually`	A state in which the `cjsuspendpool` command is used to suspend the connection

No.	Status	Status displayed in the message	Explanation
			pool. There are no connections in the connection pool. No request is sent to a connection pool with the Manual Suspension status when a connection request is sent to the root resource adapter.
9	Reserved Auto-suspension status	`suspendedAutomaticallyReserved`	A state in which the resource adapter is stopped when the connection pool is in the Auto-suspension status. A connection pool with the Reserved Auto-suspension status changes to the Auto-suspension status when the resource adapter is started. Note that at this time the connection pool warming up functionality is disabled.
10	Reserved Manual Suspension status	`suspendedManuallyReserved`	A state in which the resource adapter is stopped when the connection pool is in the Manual Suspension status. A connection pool with the Reserved Manual Suspension status changes to the Manual Suspension status when the resource adapter is started. Note that at this time the connection pool warming up functionality is disabled.

Note

The numbers indicate the numbers in *Figure 20-4* and *Figure 20-5*.

(b) Executability of commands based on the connection pool status

Depending on the state of the connection pool, some commands can be executed and some cannot be executed. The following table describes whether the commands can be executed for each connection pool state.

Table 20–6: Executability of commands based on the connection pool status

Command	Connection pool status							
	Start	Reserved Start	Auto-restarting / Manual Restarting	Automatically Blocked/ Manually Blocked	Auto-suspension	Manual Suspension	Reserved Auto-suspension	Reserved Manual Suspension
`cjstartrar`	N	Y	N	N	N	N	Y	Y
`cjstoprar`	Y	N	R#1	R#2	Y	Y	N	N
`cjclearpool`	Y	N	N	N	N	N	N	N
`cjlistrar`	Y	Y	Y	Y	Y	Y	Y	Y
`cjsuspendpool`	Y	Y	N	N	Y	N	Y	N
`cjresumepool`	N	N	N	N	Y	Y	Y	Y
`cjstopsv` (for a normal termination)	Y	Y	R#1	R#2	Y	Y	Y	Y

Command	Connection pool status							
	Start	Reserved Start	Auto-restarting / Manual Restarting	Automatically Blocked/ Manually Blocked	Auto-suspension	Manual Suspension	Reserved Auto-suspension	Reserved Manual Suspension
`cjstopsv` (for a forced termination)	Y	Y	Y	Y	Y	Y	Y	Y

Legend:

Y: Can be executed

R: Restricted

N: Cannot be executed

#1

The command is received, but the processing is executed when the status changes to Start or Suspend.

#2

The command is received, but the processing is executed when the status changes to Suspend.

(4) How to select a connection pool

When the J2EE application sends a connection request to the root resource adapter, one member connection pool is selected. The method by which the member connection pool is selected at this time is the round-robin method.

The connection pool that is selected is a member connection pool with the Start status. A member connection pool that has free connections is selected on priority.

However, if only a member connection pool with depleted connections is available, the connection request changes to pending. Furthermore, if a timeout occurs in a pending connection, the connection cannot be obtained. Also, if a connection pool is blocked when a connection request is pending, the connection request is restarted and an attempt is made to obtain a connection from the member connection pool that is next in the priority order. If a connection cannot be obtained from any of the member connection pools, the attempt to obtain the connection fails.

Note that if there is a connection request when a member connection pool with the Start status does not exist, the attempt to obtain the connection fails.

To specify the maximum size of each member connection pool, use the following guideline:

Maximum-size-of -the-member-connection-pool-(number) = *Maximum-number-of-concurrent-connections-permitted-for-the-system* / *number-of-database-nodes*

20.1.5 Procedure for stopping or starting a connection pool manually

This section describes the procedure for connection pool clustering. By manually stopping and restarting some of the connection pools when you handle the errors occurring in a database and when you perform database maintenance, you can perform database maintenance without stopping the entire system. To perform database maintenance by manually stopping and restarting some of the connection pools:

1. Check the connection pool status (see *(1)*)
   Use the server management commands to execute the operation.
2. Suspend the connection pool (see *(2)*)
   Use the server management commands to execute the operation.

3. Restart the connection pool (see *(3)*)
   Use the server management commands to execute the operation.
4. Check the connection pool status (see *(1)*)
   Use the server management commands to execute the operation.

(1) Checking the connection pool state

You check the connection pool state before and after you perform maintenance for a database in a cluster configuration.

For details on the connection pool states, see *20.1.4(3) States of the connection pool*.

You check the connection pool state with the `cjlistrar` command.

The format and example of execution of the `cjlistrar` command are as follows:

Format of execution

```
cjlistrar server-name -clusterpool
```

Example of execution

```
cjlistrar MyServer -clusterpool
```

(2) Suspending the connection pool

This section describes the procedure for suspending the connection pool.

You can suspend a connection pool with the `cjsuspendpool` command. When you execute the `cjsuspendpool` command, the connection pool is blocked and suspended and the connection requests are no longer received.

You can execute the `cjsuspendpool` command when the status of the connection pool is as follows:

- Start
- Reserved Start
- Auto-suspension
- Reserved Auto-suspension

For details on how to check the state of the connection pool, see *(1) Checking the connection pool state*.

The format and example of execution of the `cjsuspendpool` command are as follows:

Format of execution

```
cjsuspendpool server-name -resname display-name-of-member-resource-adapter
-to-be-suspended
```

Example of execution

```
cjsuspendpool MyServer -resname DB_Connector_for_Oracle_ClusterPool_Member
```

(3) Restarting the connection pool

This section describes the procedure for restarting the connection pool.

You can restart the connection pool with the `cjresumepool` command. When you execute the `cjresumepool` command, the connection pool status changes to Manual Restarting and the restart processing is executed. When the restart processing finishes, the connection pool status changes to Start and the connection requests are received.

You can execute the `cjresumepool` command when the state of the connection pool is as follows:

- The number of unused connection management threads is equal to the maximum number of connections in the connection pool or more
- The status of the connection pool is Manual Suspension, Reserved Manual Suspension, Auto-suspension, or Reserved Auto-suspension

For details on how to check the state of the connection pool, see *(1) Checking the connection pool state*.

The format and example of execution of the `cjresumepool` command are as follows:

Format of execution

```
cjresumepool server-name -resname display-name-of-member-resource-adapter
-to-be-restarted
```

Example of execution

```
cjresumepool MyServer -resname DB_Connector_for_Oracle_ClusterPool_Member
```

Note that after you execute the `cjresumepool` command, you must check the status of the connection pool to check whether the restart processing was executed correctly. For details on how to check the state of the connection pool, see *(1) Checking the connection pool state*.

20.1.6 Settings required for clustering a connection pool

To cluster a connection pool, you must set up the DB Connector properties. For details on setting up the DB Connector properties, see *20.3 Settings for connecting to the database (in the case of a cluster connection pool)*.

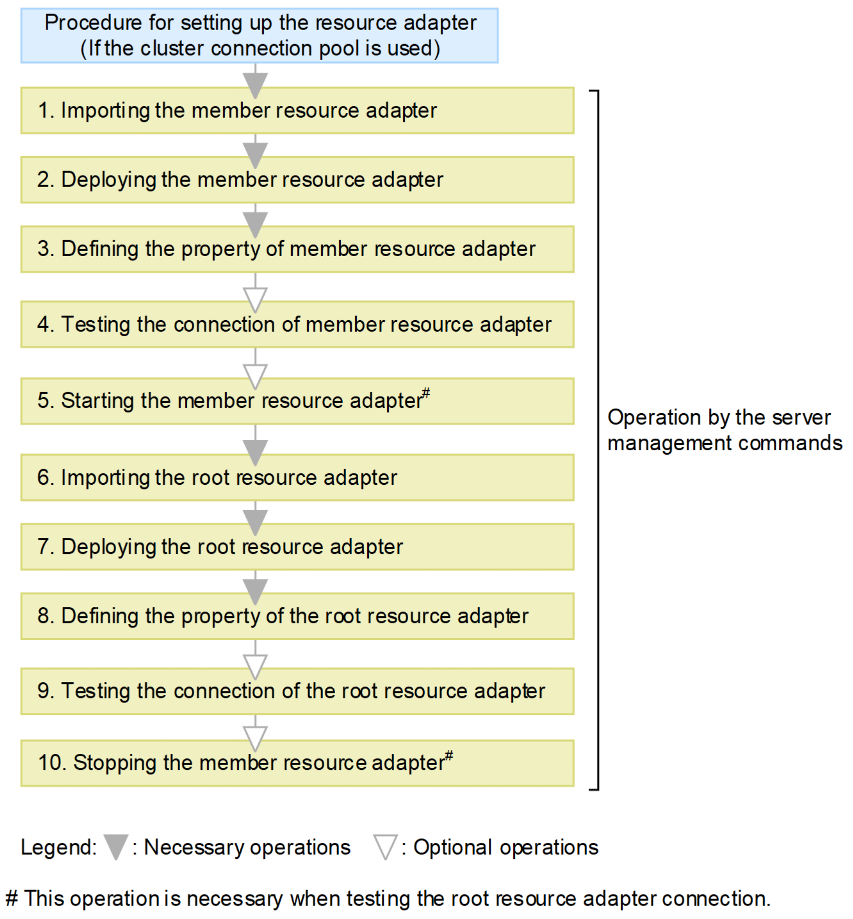
20.2 Resource connections

This section describes the flow for setting the resource adapter for the connection pool clustering functionality when a cluster connection pool is used.

20.2.1 Procedure for resource adapter settings (To use connection pool clustering)

The following figure shows the procedure for resource adapter settings for connecting to a database when the connection pool is clustered.

Figure 20–6: Procedure for resource adapter settings in connection pool clustering


Procedure for setting up the resource adapter (If the cluster connection pool is used)
1. Importing the member resource adapter
2. Deploying the member resource adapter
3. Defining the property of member resource adapter
4. Testing the connection of member resource adapter
5. Starting the member resource adapter#
6. Importing the root resource adapter
7. Deploying the root resource adapter
8. Defining the property of the root resource adapter
9. Testing the connection of the root resource adapter
10. Stopping the member resource adapter#
Operation by the server management commands
Legend: ▼ : Necessary operations ▽ : Optional operations
# This operation is necessary when testing the root resource adapter connection.

A description of points 1 to 10 in the figure is as follows:

1. Use the server management commands to import the member resource adapters.

   Use the `cjimportres` command to import the member resource adapters.

   For details on the resource adapter to import, see the manual *uCosminexus Application Server Common Container Functionality Guide*.

2. Use the server management commands to deploy the member resource adapters.

   Use the `cjdeployrar` command to deploy the member resource adapters.

3. Use the server management commands to define the member resource adapter properties.

Use the `cjgetrarprop` command to obtain the HITACHI Connector Property file, edit the file, and then use the `cjsetrarprop` command to apply the edited content.

The items that you can set up by defining the member resource adapter and root resource adapter properties differ. For details on the items defined in the properties that can be set for the member resource adapters and root resource adapter, see *20.1 Functionality for connection pool clustering*.

For details on the resource adapter properties specified for the functionality to be used, see the following locations:

- Settings for using the resource connection and the transaction management functionality
  uCosminexus Application Server Common Container Functionality Guide
- Functionality for performance tuning
  uCosminexus Application Server Common Container Functionality Guide
- Functionality for fault tolerance
  uCosminexus Application Server Common Container Functionality Guide
- Setting the optional names for J2EE resources
  uCosminexus Application Server Common Container Functionality Guide

4. Use the server management commands to implement the connection test for the member resource adapters.
   Use the `cjtestres` command to implement the connection test for the member resource adapters.
   For details on the contents to verify in connection tests for the member resource adapters, see the manual *uCosminexus Application Server Common Container Functionality Guide*.
   The steps from 1 to 4 are repeated for the number of member resource adapters only.

5. Use the server management commands to start the member resource adapters.
   To implement the connection test of the root resource adapter, start the member resource adapters first. Use the `cjstartrar` command to start the member resource adapters.

6. Use the server management commands to import the root resource adapter.
   Use the `cjimportres` command to import the root resource adapter.
   For details on the resource adapter to import, see the manual *uCosminexus Application Server Common Container Functionality Guide*.

7. Use the server management commands to deploy the root resource adapter.
   Use the `cjdeployrar` command to deploy the root resource adapter.

8. Use the server management commands to define the root resource adapter properties.
   Use the `cjgetrarprop` command to obtain the HITACHI Connector Property file, edit the file, and then use the `cjsetrarprop` command to apply the edited content.

9. Use the server management commands to implement the connection test for the root resource adapter.
   Use the `cjtestres` command to implement the connection test for the root resource adapter.
   For details on the contents to verify in the connection test for the root resource adapter, see the manual *uCosminexus Application Server Common Container Functionality Guide*.

10. Use the server management commands to stop the member resource adapters.
    After you implement the connection test for the root resource adapter, stop the member resource adapter. Use the `cjstoprar` command to stop the member resource adapters.

For details on the operations with the server management commands, see *3. Basic Operations of Server Management Commands* in the *uCosminexus Application Server Application Setup Guide*. Also, for details on the commands, see *2.4 Resource operation commands used with J2EE servers* in the *uCosminexus Application Server Command Reference*

Guide. For details on the property files, see *4. Property Files Used for Setting Resources* in the *uCosminexus Application Server Application and Resource Definition Reference Guide*.

Important note

To cluster a connection pool, you must resolve the references from the J2EE application to the root resource adapter. When you use the root resource adapter to define the J2EE application properties, resolve the references from the J2EE application to the root resource adapter.

20.3 Settings for connecting to the database (in the case of a cluster connection pool)

When you connect to a clustered database of Oracle RAC using DB Connector, you can cluster and use the connection pool. For details on the cluster connection pools, see *20.1 Functionality for connection pool clustering*.

This section describes the settings for clustering the DB Connector connection pool.

20.3.1 Overview of a cluster connection pool

A connection pool that is clustered is called a *cluster connection pool*. A cluster connection pool consists of a root resource adapter and a member resource adapter. The connection pool of a member resource adapter is called the *member connection pool*.

The following sections describe how to control the operations and state for using the cluster connection pool of DB Connector:

(1) Setting a cluster connection pool

To use a cluster connection pool, you need to setup a resource adapter that can use the cluster connection pool.

Set up the DB Connector for the cluster connection pool with the following procedure:

Note that you repeat steps 1 and 2 for the required number of member resource adapters.

1. Setup a DB Connector for the member resource adapter.
   Use the following procedures to setup a DB Connector for member resource adapters:
   - Import the RAR file of the DB Connector for the member resource adapter.
   - Define properties.
   - Deploy the DB Connector.
   - Check the connectivity.
     Use the connectivity test to verify if the connection is correct.
2. Start the DB Connector for the member resource adapter.
3. Set up the DB Connector for the root resource adapter.
   Use the following procedures to set up the DB Connector for the root resource adapter:
   - Import the RAR file of DB Connector for the root resource adapter.
   - Define properties.
   - Deploy DB Connector.
   - Check the connectivity.
     Use the connectivity test to verify if the connection is correct.
4. Start the DB Connector for the root resource adapter.

The root resource adapter is directly accessed from a J2EE application, so for the deployed root resource adapter, you need to resolve the references by using the J2EE application property settings. For details, see *9.3.3 Defining resource adapter references* in the manual *uCosminexus Application Server Application Setup Guide*.

Reference note

To set up new DB Connector properties, you can use the template files provided in Cosminexus Component Container.

The template files of the HITACHI Connector Property file are saved in the following directory:

- In Windows:
  Cosminexus-installation-directory`\CC\admin\templates\`
- In UNIX:
  `/opt/Cosminexus/CC/admin/templates/`

You can use this template file to edit the HITACHI Connector property file before importing the DB Connector. Copy and use the template file.

For details on the template file names of the HITACHI Connector Property file, see *4.1.13 Template files of the HITACHI Connector Property File* in the *uCosminexus Application Server Application and Resource Definition Reference Guide*.

You do not use the template file to change the properties of DB Connector for which properties are already specified. You obtain the Connector properties of an imported DB Connector, and then edit the HITACHI Connector Property file.

(2) State of cluster connection pool and operations that can be executed

You can suspend a member connection pool manually when there is a database failure or for maintenance. If the member connection pool is suspended, the processing is not performed in the case of a connection acquisition request to the root resource adapter.

You can manually restart the suspended member connection pool. In the case of a connection acquisition request to the root resource adapter, only the process for a running connection pool is performed.

For details on controlling the connection pool state, see *20.1.4 Connection pool clustering operations*.

(3) Functional differences based on the types of resource adapters

The following table lists the resource adapter types and the functions that can be used:

Table 20–7: List of functions based on the resource adapter types

Functions	Types of resource adapters	
	Root resource adapter	Member resource adapter
Connection pooling	N	D
Warming up of connection pool	N	Y
Connection sharing and association	N	Y
Statement pooling	N	Y
Cache of DataSource object	Y	N
Optimization of sign-on in the container management of DB Connector	N	Y

Functions	Types of resource adapters	
	Root resource adapter	Member resource adapter
Connection failure detection	N	D
Timeout in connection failure detection	N	D
Connection acquisition request when connections are used up	N	D
Retry in acquiring connection	N	N
Display of connection pool information	N	Y
Clear connection pool	N	Y
Connection auto close	N	Y
Connection sweeper	N	Y
Test the connectivity	Y	Y
Connection pool count adjustment function	N	Y
Display of connection pool information	N	Y
Suspend connection pool#	N	Y
Restart connection pool#	N	Y
User-specified name space function of J2EE resources	Y	N

Legend:

D: Definitely enabled

Y: Can be used

N: Cannot be used

#

If you do not use a clustered connection pool, you cannot suspend and restart the connection pool.

The following table lists the resource adapter types that set the DB Connector attributes:

Table 20–8: Types of resource adapters and attribute settings

Settings	Types of resource adapters	
	Root resource adapter	Member resource adapter
Transaction support level	N	Y
Feasibility of acquiring log	Y	Y
Database connection information	--	Y
DB Connector specific settings (such as statement pool)	--	Y
Security information (user name and password)	N	Y
Connection pool size	N	Y
Cluster connection pool specific settings	Y	--

Legend:

Y: Settings are required

N: Settings are not required

--: No settings

20.3.2 Setting the DB Connector for a member resource adapter

Use the following procedure to setup a DB Connector for a member resource adapter:

1. Import the DB Connector for the member resource adapter.
2. Define the properties.
3. Deploy the DB Connector for the member resource adapter.
4. Check the connectivity.

(1) Importing the DB Connector for member resource adapter

Execute the following command to import the DB Connector for the member resource adapter:

(a) Execution format

```
cjimportres [server-name] [-nameserver provider-URL] -type rar -f file-path
```

Specify the RAR file in *file-path*.

The RAR file is stored in the following directory:

- In Windows
  Cosminexus-installation-directory`\CC\DBConnector\ClusterPool\`
- In UNIX
  `/opt/Cosminexus/CC/DBConnector/ClusterPool/`

The RAR files to be imported as member resource adapters are explained below:

`DBConnector_Oracle_CP_ClusterPool_Member.rar`

The DBConnector_Oracle_CP_ClusterPool_Member.rar is a member resource adapter of the cluster connection pool. Use this RAR file when you use a local transaction or when you do not use a transaction (when LocalTransaction or NoTransaction is specified in the transaction support level). Connect to Oracle by using ConnectionPoolDataSource of Oracle JDBC Thin Driver.

You cannot set this RAR file in the resource references of the J2EE applications.

(b) Example of execution

```
cjimportres MyServer -type rar -f "c:\Program Files\Hitachi\Cosminexus\CC\DB
Connector\ClusterPool\DBConnector_Oracle_CP_ClusterPool_Member.rar"
```

For details on the `cjimportres` command, see *cjimportres (import resource)* in the *uCosminexus Application Server Command Reference Guide*.

(2) Defining DB Connector properties for member resource adapter

You define the DB Connector properties for the member resource adapter. For details on the procedure for defining the properties, see *4.2.2 Defining the DB Connector properties* in the manual *uCosminexus Application Server Application Setup Guide*. The DB Connector property settings for the member resource adapters will be explained here.

(a) General information regarding the DB Connector for member resource adapters

The following table describes the settings for general information attributes (`<outbound-resourceadapter>` tag) of the DB Connector that can be set:

Items	Mandatory	Corresponding tags
Transaction support level#	Y	`<transaction-support>`
Scope of re-authentication support	Y	`<reauthentication-support>`

Legend:

Y: Mandatory

#

Specify the same transaction support level for the member resource adapters that form one cluster connection pool.

For details on the property settings, see *4.1.1 Specifications of the HITACHI Connector Property file* in the *uCosminexus Application Server Application and Resource Definition Reference Guide*.

(b) Configuration properties for member resource adapters

The configuration properties (`<config-property>` tag) and settings of DB Connector for member resource adapters are the same as that of the corresponding resource adapter (`DBConnector_Oracle_CP.rar`). For details on the configuration properties of the corresponding resource adapter, see *4.2.2 Defining the DB Connector properties* in the manual *uCosminexus Application Server Application Setup Guide*.

(c) Runtime properties

The following table describes the settings for runtime properties (`<outbound-resourceadapter>` - `<connection-definition>` - `<connector-runtime>` tag) of the DB Connector for the member resource adapter:

Items	Corresponding tags
Property name	`<property-name>`
Property data type	`<property-type>`
Property value	`<property-value>`

Repeat the above settings for all the properties that you want to define.

Setup the following items in the property name (`<property-name>`):

Property items	Settings of property name (<property-name>)
User name#	`User`
Password#	`Password`
Minimum number of connections to be pooled in the connection pool	`MinPoolSize`
Maximum number of connections to be pooled in the connection pool	`MaxPoolSize`
Choose whether to output the log	`LogEnabled`

Property items	Settings of property name (<property-name>)
Period from the last usage of connection until you determine whether to cancel the connection automatically (connection sweeper)	`ConnectionTimeout`
Interval for canceling the connection automatically (connection sweeper)	`SweeperInterval`
Maximum waiting time when the connection acquisition requests (when the connections are used up) are managed in a queue	`RequestQueueTimeout`
Choose whether to enable alert output for monitoring the connection pool	`WatchEnabled`
Interval for monitoring the connection pool	`WatchInterval`
Threshold value for monitoring the usage state of the connection pool	`WatchThreshold`
Choose whether to output the file with the connection pool monitoring results	`WatchWriteFileEnabled`
Interval for operating the connection count adjustment function	`ConnectionPoolAdjustmentInterval`
Choose whether to enable the warming up function of the connection pool	`Warmup`

#

Specify the same user name for the member resource adapters that form one cluster connection pool.

Note:

The following items are always `Enabled` regardless of the scope of settings for the member resource adapters:

Property items	Valid values	Settings of property name (<property-name>)
Minimum number of connections to be pooled in connection pool	Connection pooling function is enabled. Default value `10` is assumed even if `0` is specified in `MinPoolSize` and `MaxPoolSize`.	`MinPoolSize`
Maximum number of connections to be pooled in connection pool		`MaxPoolSize`
Choose a method to check whether a failure has occurred in the connections inside the pool	Always `1` (Failure detected when the connection is acquired)	`ValidationType`
Choose whether to manage the connection acquisition requests in a queue when the connections are used up	Always `true`	`RequestQueueEnable`
Choose whether to enable the timeout of the network failure detection function	Always `true`	`NetworkFailureTimeout`

The retry function for acquiring connections is always disabled regardless of the settings for connection retry count (`RetryCount`) and connection retry waiting time (`RetryInterval`) of runtime properties.

(3) Deploying the DB Connector for member resource adapters

If you deploy the DB Connector for the member resource adapter, you can use it as a J2EE resource adapter. Note that you can define the properties after deploying the DB Connector. In such a case, however, define the properties when the root resource adapter to which the applicable DB Connector for the member resource adapter belongs and the DB Connector for the member resource adapter are not running. For the method of defining properties, see *(2) Defining DB Connector properties for member resource adapter*.

Execute the following command to deploy the DB Connector for the member resource adapter:

Execution format

```
cjdeployrar [server-name] [-nameserver provider-URL] -resname display-name
-of-DB-Connector-for-the-member-resource-adapter
```

Example of execution

```
cjdeployrar MyServer -resname DB_Connector_for_Oracle_ClusterPool_Member
```

For details on the `cjdeployrar` command, see *cjdeployrar (deploy resource adapter)* in the ***uCosminexus** Application Server Command Reference Guide*.

(4) Testing the connectivity of the DB Connector for a member resource adapter

Use the connectivity test to verify whether the information set in the DB Connector for the member resource adapter is correct.

Execute the following command to perform connectivity testing of the DB Connector for the member resource adapter:

Execution format

```
cjtestres [server-name] [-nameserver provider-URL] -type rar -resname Disp
lay-name-of-DB-Connector-for-the-member-resource-adapter
```

Example of execution

```
cjtestres -type rar -resname DB_Connector_for_Oracle_ClusterPool_Member
```

For details on the `cjtestres` command, see *cjtestres (execute resource connection test)* in the ***uCosminexus** Application Server Command Reference Guide*.

Note:

You cannot delete the DB Connector for the member resource adapter whose connection is once tested until you restart the J2EE server. To delete the DB Connector for the member resource adapter, stop the root resource adapter to which the applicable DB Connector for the member resource adapter belongs and the DB Connector for the member resource adapter, and then restart the J2EE server.

20.3.3 Setting the DB Connector for a root resource adapter

Set up the DB Connector for the root resource adapter with the following procedure:

1. Import the DB Connector for the root resource adapter.
2. Define the properties.
3. Deploy the DB Connector for the root resource adapter.
4. Check the connectivity.

(1) Importing the DB Connector for a root resource adapter

Execute the following command to import the DB Connector for the root resource adapter:

(a) Execution format

```
cjimportres [server-name] [-nameserver provider-URL] -type rar -f file-path
```

Specify the RAR file in *file-path*.

The RAR file is stored in the following directory:

- In Windows
  Cosminexus-installation-directory`\CC\DBConnector\ClusterPool\`
- In UNIX
  `/opt/Cosminexus/CC/DBConnector/ClusterPool/`

The RAR files to be imported as root resource adapters are explained below:

`DBConnector_CP_ClusterPool_Root.rar`

DBConnector_CP_ClusterPool_Root.rar is a root resource adapter of a cluster connection pool. Use this RAR file when a member resource adapter that belongs to the root resource adapter uses a local transaction or no transaction (when LocalTransaction or NoTransaction is specified in the transaction support level) for connecting to the database.

Specify the RAR file in the resource reference of the J2EE applications.

(b) Example of execution

```
cjimportres MyServer -type rar -f "c:\Program Files\Hitachi\Cosminexus\CC\DB
Connector\ClusterPool\DBConnector_CP_ClusterPool_Root.rar"
```

For details on the `cjimportres` command, see *cjimportres (import resource)* in the *uCosminexus Application Server Command Reference Guide*.

(2) Defining DB Connector properties for a root resource adapter

You define the properties of the DB Connector for the root resource adapter. For details on the procedure for defining the properties, see *4.2.2 Defining the DB Connector properties* in the manual *uCosminexus Application Server Application Setup Guide*. This section explains the settings for the properties of a DB Connector for the root resource adapter.

The valid properties for the DB Connector for the root resource adapter are described below. The items other than those described below are ignored even if specified.

- Description of the DB Connector for the root resource adapter (`<description>`)
- Name of the DB Connector for the root resource adapter (`<display-name>`)
- Configuration properties (`<outbound-resourceadapter>` - `<connection-definition>` - `<config-property>`)
- Choosing whether to output the log of runtime properties (`<outbound-resourceadapter>` - `<connection-definition>` - `<connector-runtime>`) (`<LogEnabled>`)
- Optional name information (`<outbound-resourceadapter>` - `<connection-definition>` - `<connector-runtime>` - `<resource-external-property>`)

The following table describes the settings for configuration properties:

Items	Corresponding tags
Configuration property name	`<config-property-name>`
Configuration property data type	`<config-property-type>`
Configuration property value	`<config-property-value>`

Repeat the above settings for all the configuration properties that you want to define.

The following table describes the settings and an example of configuration property settings when using `DBConnector_CP_ClusterPool_Root.rar`:

Table 20–9: Example of configuration property settings when using DBConnector_CP_ClusterPool_Root.rar

Item names	Example of settings
`Algorithm`	`RoundRobin`
`enableAutoPoolSuspend`	`True`
`enableAutoPoolResume`	`True`
`dbCheckInterval`	`30`
`memberResourceAdapterName1`	`DB_Connector_for_Oracle_ClusterPool_Member1`
`memberResourceAdapterName2`	`DB_Connector_for_Oracle_ClusterPool_Member2`
`logLevel`	`ERROR`

Set up the display name of the member resource adapter for the `memberResourceAdapterName[`*n*`]` property of configuration property name (`<config-property-name>`). By default, `memberResourceAdapterName[`*n*`]` is defined up to priority 2. When you specify a member resource adapter, add the properties. Specify priority *n* in the range of 1 to 100. n need not be consecutive.

(3) Deploying the DB Connector for a root resource adapter

If you deploy the DB Connector for the root resource adapter, you can use it as a J2EE resource adapter. Note that you can define the properties after deploying the DB Connector. In such a case, however, define the properties when the applicable DB Connector for the root resource adapter is not running. For details on how to define the properties, see *(2) Defining DB Connector properties for a root resource adapter*.

Execute the following command to deploy the DB Connector for the root resource adapter:

Execution format

```
cjdeployrar [server-name] [-nameserver provider-URL] -resname Display-name
-of-DB-Connector-for-the-root-resource-adapter
```

Example of execution

```
cjdeployrar MyServer -resname DB_Connector_for_ClusterPool_Root
```

For details on the `cjdeployrar` command, see *cjdeployrar (deploy resource adapter)* in the *uCosminexus Application Server Command Reference Guide*.

(4) Testing the connectivity of DB Connector for root resource adapter

Use the connectivity test to verify whether the information set in the DB Connector for the root resource adapter is correct.

Execute the following command to perform the connectivity testing of the DB Connector for the root resource adapter:

Execution format

```
cjtestres [server-name] [-nameserver provider-URL] -type rar -resname Disp
lay-name-of-DB-Connector-for-the-root-resource-adapter
```

Example of execution

```
cjtestres -type rar -resname DB_Connector_for_ClusterPool_Root
```

For details on the `cjtestres` command, see *cjtestres (execute resource connection test)* in the *uCosminexus Application Server Command Reference Guide*.

Note:

You cannot delete the DB Connector for the root resource adapter whose connection is once tested until you restart the J2EE server. To delete the DB Connector for the root resource adapter, stop the DB Connector for the root resource adapter and then restart the J2EE server.

20.3.4 Starting and stopping the DB Connector for a member resource adapter

(1) Starting the DB Connector for a member resource adapter

Execute the following command to start the DB Connector for the member resource adapter:

Execution format

```
cjstartrar [server-name] [-nameserver provider-URL] -resname Display-name
-of-DB-Connector-for-the-member-resource-adapter
```

Example of execution

```
cjstartrar MyServer -resname DB_Connector_for_Oracle_ClusterPool_Member
```

For details on the `cjstartrar` command, see *cjstartrar (start resource adapter)* in the *uCosminexus Application Server Command Reference Guide*.

(2) Stopping the DB Connector for a member resource adapter

Execute the following command to stop the DB Connector for the member resource adapter:

Execution format

```
cjstoprar [server-name] [-nameserver provider-URL] -resname Display-name-o
f-DB-Connector-for-the-member-resource-adapter
```

Example of execution

```
cjstoprar MyServer -resname DB_Connector_for_Oracle_ClusterPool_Member
```

For details on the `cjstoprar` command, see *cjstoprar (stop resource adapter)* in the *uCosminexus Application Server Command Reference Guide*.

Note:

Stop the root resource adapter to which the member resource adapter belongs and then stop the member resource adapter.

20.3.5 Starting and stopping the DB Connector for a root resource adapter

(1) Starting the DB Connector for a root resource adapter

Execute the following command to start the DB Connector for the root resource adapter:

Execution format

```
cjstartrar [server-name] [-nameserver provider-URL] -resname Display-name
-of-DB-Connector-for-the-root-resource-adapter
```

Example of execution

```
cjstartrar MyServer -resname DB_Connector_for_ClusterPool_Root
```

For details on the `cjstartrar` command, see *cjstartrar (start resource adapter)* in the *uCosminexus Application Server Command Reference Guide*.

Note:

Start the member resource adapter belonging to the root resource adapter and then start the root resource adapter.

(2) Stopping the DB Connector for a root resource adapter

Execute the following command to stop the DB Connector for the root resource adapter:

Execution format

```
cjstoprar [server-name] [-nameserver provider-URL] -resname Display-name-o
f-DB-Connector-for-the-root-resource-adapter
```

Example of execution

```
cjstoprar MyServer -resname DB_Connector_for_ClusterPool_Root
```

For details on the `cjstoprar` command, see *cjstoprar (stop resource adapter)* in the *uCosminexus Application Server Command Reference Guide*.

Note:

If a J2EE resource in the J2EE application references the DB Connector for the root resource adapter, stop the J2EE application and then stop the DB Connector for the root resource adapter.

20.3.6 Checking the state of the connection pool

You can reference the states of the connection pools that are used in a cluster connection pool with the following two methods:

- Reference the resource adapter names and the state of the member connection pools for all the deployed resource adapters using the `cjlistrar` command.
- Reference the member connection pool information using the `cjlistpool` command.

  Reference the member connection pool information and lock/ suspend/ restart the member connection pool as and when required. For details on defining the state of the connection pool, see *20.1.4 Connection pool clustering operations*.

(1) Checking the state of the connection pool

The resource adapter names and the state of the resource adapters are displayed for all the deployed resource adapters. The state of the connection pool is also displayed if the resource adapter is a member resource adapter of a cluster connection pool.

Execute the following command to reference the state of a connection pool:

Execution format

```
cjlistrar [server-name] [-nameserver provider-URL] -clusterpool
```

Example of execution

```
cjlistrar MyServer -clusterpool
```

For details on the `cjlistrar` command, see *cjlistrar (list resource adapters)* in the *uCosminexus Application Server Command Reference Guide*.

(2) Referencing the member connection pool information

The connection pool information is displayed. The information about the member connection pool is also displayed if the resource adapter is a member resource adapter of the cluster connection pool.

Execute the following command to reference the connection pool information for all the resource adapters:

Execution format

```
cjlistpool [server-name] [-nameserver provider-URL] -resall
```

Example of execution

```
cjlistpool MyServer -resall
```

Execute the following command for displaying the connection pool information for a specific resource adapter:

Execution format

```
cjlistpool [server-name] -resname resource-adapter-display-name
```

Example of execution

```
cjlistpool MyServer -resname DB_Connector_for_Oracle_ClusterPool_Member
```

The connection pool information for the resource adapter is displayed.

For details on the `cjlistpool` command, see *cjlistpool (list connection pools)* in the *uCosminexus Application Server Command Reference Guide*.

Note:

You cannot execute the `cjlistpool` command for a root resource adapter.

20.3.7 Suspending the connection pool

You can suspend a member connection pool manually in the case of a database failure or during maintenance. A suspended connection pool does not receive the connection acquisition request.

Execute the following command to suspend the member connection pool of the cluster connection pool:

Execution format

```
cjsuspendpool [server-name] [-nameserver provider-URL] -resname display-na
me-of-the-member-resource-adapter-to-be-suspended
```

Example of execution

```
cjsuspendpool MyServer -resname DB_Connector_for_Oracle_ClusterPool_Member
```

For details on the `cjsuspendpool` command, see *cjsuspendpool (suspend member connection pool)*.

Note:

You cannot automatically resume a connection pool that was suspended manually by using the `cjsuspendpool` command. Use the `cjresumepool` command to restart the connection pool manually.

20.3.8 Resuming the connection pool

You cannot resume a suspended member connection pool manually. When the J2EE application makes a connection acquisition request to the root resource adapter, the resumed connection pool is able to receive the connection acquisition request again.

Execute the following command to restart the member connection pool of a cluster connection pool:

Execution format

```
cjrestartpool [server-name] [-nameserver provider-URL] -resname display-na
me-of-the-member-resource-adapter-to-be-resumed
```

Example of execution

```
cjresumepool MyServer -resname DB_Connector_for_Oracle_ClusterPool_Member
```

For details on the `cjresumepool` command, see *cjresumepool (restart member connection pool)*.

20.4 Overview of settings and operations

This section gives an overview for the following resource adapter settings:

- Settings for connecting to the database (in the case of cluster connection pool)

20.4.1 Settings for connecting to the database (in the case of cluster connection pool)

This operation is required for using the clustered connection pool of the DB Connector.

Table 20–10: Overview of settings required for connecting to the database (in the case of cluster connection pool)

Settings	Contents	Reference
Setting the DB Connector for a member resource adapter	Execute all the operations up to importing, setting and testing the connectivity of the DB Connector for the member resource adapter.	*20.3.2*
Setting the DB Connector for a root resource adapter	Execute all the operations up to importing, setting and testing the connectivity of the DB Connector for the root resource adapter.	*20.3.3*
Starting and stopping the DB Connector for a member resource adapter	Start the DB Connector for the member resource adapter. Also, stop the DB Connector for a running member resource adapter.	*20.3.4*
Starting and stopping the DB Connector for a root resource adapter	Start the DB Connector for the root resource adapter. Also, stop the DB Connector for a running root resource adapter.	*20.3.5*
Checking the state of the connection pool	Reference the member connection pool state.	*20.3.6*
Suspending the connection pool	Suspend the member connection pool.	*20.3.7*
Resuming the connection pool	Resume the member connection pool that was suspended.	*20.3.8*

20.5 Resource operation commands to be used on a J2EE server

This section describes `cjresumepool` (restarts a member connection pool) and `cjsuspendpool` (suspends a member connection pool) in the resource operation commands to be used on a J2EE server.

cjresumepool (restart member connection pool)

Format

```
cjresumepool [server-name] [-nameserver provider-URL]
       -resname resource-adapter-display-name
       [-resname resource-adapter-display-name ...]
```

Function

This command restarts a specified member connection pool of the cluster connection pool. This command can restart multiple member connection pools concurrently. The command attempts to restart all specified member connection pools even if there are member connection pools for which restart fails. If even one member connection pool cannot be restarted, the command terminates abnormally.

You must execute this command in the 1.4 mode; if it is executed in the basic mode, it terminates abnormally.

When this command is executed for a resource adapter that is running, the command sets the member connection pool to the "restarting manually" status. If the warming up functionality for member connection pools is enabled, the command next pools the connections in the member connection pool and then places the member connection pool in running status.

When this command is executed for a resource adapter that is stopped, the command sets the member connection pool in "running reserved" status.

Arguments

server-name

Specifies the name of a connected J2EE server. If the server name is omitted, the host name is assumed.

`-nameserver` *provider-URL*

Specifies the access protocol for the CORBA Naming Service, the name of the host running the CORBA Naming Service, and the port number being used by the host. These items are specified in the following format:

```
protocol-name::host-name:port-number
```

For details on specifications, see *2.1.2 Provider URL* in the manual *uCosminexus Application Server Command Reference Guide*.

`-resname` *resource-adapter-display-name*

Specifies the display name of a member resource adapter that is to be restarted.

If the specified resource adapter is not a member resource adapter of the cluster connection pool, the command terminates abnormally.

This argument enables you to execute the command regardless of whether the specified resource adapter is running or stopped. However, whether or not the command can be executed depends on the status of the member connection pool, as shown in the following table:

Member connection pool status	Command executability
Suspended automatically status Suspended manually status Suspended automatically reserved status Suspended manually reserved status	Can be executed
Running status Running reserved status Restarting automatically status Restarting manually status Blocked automatically status Blocked manually status	Cannot be executed

Input examples

```
cjresumepool MyServer -resname DB_Connector_for_Oracle_ClusterPool_Member
```

Return values

`0`:

The command terminated normally.

`1`:

The command terminated abnormally.

`2`:

The command could not be executed because of an exclusion error.

`3`:

A timeout error occurred.

`9`:

The command could not be executed because there are no administrator privileges (in Windows Server 2012, Windows Server 2008, Windows 8, Windows 7, or Windows Vista).

Notes

- When you specify a server name in the command arguments, you must specify it immediately after the command name. You can specify the other arguments in any order as long as they are subsequent to the server name (or subsequent to the command name if the server name is omitted). However, you cannot switch the sequence of an option name and its corresponding value (for example, you cannot specify *application-name* `-name`); also, you cannot specify an option name in conjunction with a non-corresponding value (for example, you cannot specify `-nameserver` *application-name* `-name` *provider-URL*).
- When you specify a server name for the command argument, you must specify a character string that is consistent with the letter case of the server name specified for the `cjsetup` command. For details on the `cjsetup` command, see *cjsetup (set up or unsetup J2EE server)* in the manual *uCosminexus Application Server Command Reference Guide*.

cjsuspendpool (suspend member connection pool)

Format

```
cjsuspendpool [server-name] [-nameserver provider-URL]
       -resname resource-adapter-display-name
       [-resname resource-adapter-display-name ...]
```

Function

This command suspends a member connection pool of the cluster connection pool. The command can suspend multiple member connection pools concurrently. The command attempts to suspend all specified member connection pools even if there are member connection pools for which suspension fails. If even one member connection pool cannot be suspended, the command terminates abnormally.

This command is effective only when it is executed in the 1.4 mode. If the command is executed in the basic mode, it terminates abnormally.

When this command is executed for a resource adapter that is running, the command sets the member connection pool to "blocked manually" status and terminates. Next, suspension processing is executed in the J2EE server. After suspension processing is completed, the member connection pool is set to "suspended manually" status.

When this command is executed for a resource adapter that is stopped, the command sets the member connection pool to "suspended manually reserved" status.

Arguments

server-name

Specifies the name of a connected J2EE server. If the server name is omitted, the host name is assumed.

`-nameserver` *provider-URL*

Specifies the access protocol for the CORBA Naming Service, the name of the host running the CORBA Naming Service, and the port number being used by the host. These items are specified in the following format:

```
protocol-name::host-name:port-number
```

For details on specifications, see *2.1.2 Provider URL* in the manual *uCosminexus Application Server Command Reference Guide*.

`-resname` *resource-adapter-display-name*

Specifies the display name of a member resource adapter that is to be suspended.

If the specified resource adapter is not a member resource adapter of the cluster connection pool, the command terminates abnormally.

This argument enables you to execute the command regardless of whether the specified resource adapter is running or stopped. However, whether or not the command can be executed depends on the status of the member connection pool, as shown in the following table:

Member connection pool status	Command executability
Running status Running reserved status Suspended automatically status Suspended automatically reserved status	Can be executed
Suspended manually status	Cannot be executed

Member connection pool status	Command executability
Suspended manually reserved status Restarting automatically status Restarting manually status Blocked automatically status Blocked manually status	

Input examples

```
cjsuspendpool MyServer -resname DB_Connector_for_Oracle_ClusterPool_Member
```

Return values

`0`:

The command terminated normally.

`1`:

The command terminated abnormally.

`2`:

The command could not be executed because of an exclusion error.

`3`:

A timeout error occurred.

`9`:

The command could not be executed because there are no administrator privileges (in Windows Server 2012, Windows Server 2008, Windows 8, Windows 7, or Windows Vista).

Notes

- When you specify a server name in the command arguments, you must specify it immediately after the command name. You can specify the other arguments in any order as long as they are subsequent to the server name (or subsequent to the command name if the server name is omitted). However, you cannot switch the sequence of an option name and its corresponding value (for example, you cannot specify *data-source-display-name* `-resname`); also, you cannot specify an option name in conjunction with a non-corresponding value (for example, you cannot specify `-nameserver` *data-source-display-name* `-resname` *provider-URL*).
- When you specify a server name for the command argument, you must specify a character string that is consistent with the letter case of the server name specified for the `cjsetup` command. For details on the `cjsetup` command, see *cjsetup (set up or unsetup J2EE server)* in the manual *uCosminexus Application Server Command Reference Guide*.

20.6 HITACHI Connector Property file

The HITACHI Connector Property file is used to acquire and edit the resource adapter properties.

20.6.1 Properties that you can specify in the <config-property> tag set up for DB Connector

This section describes the values that you can specify in the `<config-property>` tag which is to be set for the following DB Connectors:

- When connecting to Oracle by using a cluster connection pool (root resource adapter)
- When connecting to Oracle by using a cluster connection pool (member resource adapter)

(1) When connecting to Oracle by using cluster connection pool (root resource adapter)

- `DBConnector_CP_ClusterPool_Root.rar`
  Use the above when the transaction is not managed or when a local transaction is used.
  For details on the properties that you can specify, see the following table.

Table 20–11: Properties that you can specify when using DBConnector_CP_ClusterPool_Root.rar

config-property-name	config-property-type	config-property-value
`algorithm`	`java.lang.String`	Specifies how to select the connection pool of the connection pool clustering functionality. • `RoundRobin` An attempt is made to acquire the connections as per the priority order from the connection pool that is next in the priority list below the last chosen connection pool. When the connection pool with the lowest priority is reached, the connection pool with the highest priority is chosen. An exception is thrown when you cannot acquire a connection from the entire connection pool. Specified when the cluster database is active/active configuration and aims at load distribution in each instance. The default value is <u>`RoundRobin`</u>.
`dbCheckInterval`	`java.lang.Integer`	Specifies an integer value from 2 to 2147483647 (unit: seconds) for the interval to check the DB node status in the suspended connection pool. If you specify a value outside the valid range, the default value is assumed. The default value is <u>`30`</u>. When the Oracle JDBC Thin Driver is used in the member resource adapter, specify a time period longer than the `loginTimeout` property value of each member resource adapter.
`enableAutoPoolResume`	`java.lang.Boolean`	Specifies whether to enable or disable the auto-resume functionality of the connection pool. • If you specify `true` The auto-resume functionality of the connection pool is enabled. • If you specify `false` The auto-resume functionality of the connection pool is disabled. The default value is <u>`true`</u>.

config-property-name	config-property-type	config-property-value
`enableAutoPoolSuspend`	`java.lang.Boolean`	Specifies whether to enable or disable the auto-suspend functionality of the connection pool. • If you specify `true` The auto-suspend functionality of the connection pool is enabled. • If you specify `false` The auto-suspend functionality of the connection pool is disabled. The default value is <u>`true`</u>.
`logLevel`	`java.lang.String`	Specifies the log trace level output by the DB Connector. You can specify the following values: • `0` or `ERROR` • `10` or `WARNING` • `20` or `INFORMATION` The default value is <u>`0 or ERROR`</u>.
`memberResourceAdapterName[`*n*`]`	`java.lang.String`	Specifies the display name of the member resource adapter of priority *n*. This property is not defined by default, so add this property as and when required. For *n*, specify the value from 1 to 100.
`memberResourceAdapterName1`	`java.lang.String`	Specifies the display name of the member resource adapter of priority 1. Smaller the specified value, higher the priority order.
`memberResourceAdapterName2`	`java.lang.String`	Specifies the display name of the member resource adapter of priority 2.

(2) When connecting to Oracle by using the cluster connection pool (member resource adapter)

- `DBConnector_Oracle_CP_ClusterPool_Member.rar`

  Use the above when the transaction is not managed or when a local transaction is used.

  For details on the properties that you can specify, see the following table.

Table 20–12: Properties that you can specify when using DBConnector_Oracle_CP_ClusterPool_Member.rar

config-property-name	config-property-type	config-property-value
`CallableStatementPoolSize` *uCosminexus Application Server Common Container Functionality Guide*	`java.lang.Integer`	Specifies the pool size of `CallableStatement` for each connection allocated to the connection pool. The default value is <u>`10`</u>. When `0` is specified, statements are not pooled.
`CancelStatement`	`java.lang.Boolean`	Specifies whether to cancel the SQL statement being executed in the `Statement` class, `CallableStatement` class, and `PreparedStatement` class, in the case of a transaction timeout or a forced termination of an application. • If you specify `true` The SQL statement being executed is cancelled. • If you specify `false` The SQL statement being executed is not cancelled. The default value is <u>`true`</u>. Specify `false` when connecting to an exclusive server.
`ConnectionIDUpdate`	`java.lang.Boolean`	Specifies whether to update the connection ID for each `DataSource#getConnection` method.

config-property-name	config-property-type	config-property-value
		• If you specify `true` The connection ID is generated for each `DataSource#getConnection` method. • If you specify `false` The connection ID is generated for the first `DataSource#getConnection` method and is not updated thereafter. The default value is `false`.
`databaseName`	`java.lang.String`	Specifies a specific database name (SID) on the Oracle server. The set value is passed to the `setDatabaseName` method of the DataSource node interface of Oracle JDBC Thin Driver.
`loginTimeout`	`java.lang.Integer`	Specifies an integer value (unit: milliseconds) from 1 to 2147483647 when trying to connect to the database. If you specify a value outside the valid range, the default value is assumed. The default value is `8000`. The specified value is rounded up in increments of 1 second, and is passed to the `setLoginTimeout` method of the DataSource system interface of Oracle JDBC Thin Driver.
`logLevel`	`java.lang.String`	Specifies the log trace level output by the DB Connector. You can specify the following values: • `0` or `ERROR` • `10` or `WARNING` • `20` or `INFORMATION` The default value is `0 or ERROR`.
`portNumber`	`java.lang.Integer`	Specifies the port number by which the Oracle server listens to the request. The default port number is `1521`. The set value is passed to the `setPortNumber` method of the DataSource node interface of Oracle JDBC Thin Driver.
`PreparedStatementPool Size` *uCosminexus Application Server Common Container Functionality Guide*	`java.lang.Integer`	Specifies the pool size of `PreparedStatement` for each connection allocated to the connection pool. The default value is `10`. When `0` is specified, statements are not pooled.
`serverName`	`java.lang.String`	Specifies the host name or IP address of the Oracle server. The set value is passed to the `setServerName` method of the DataSource node interface of Oracle JDBC Thin Driver.
`url`	`java.lang.String`	Specifies the JDBC URL required by the Oracle JDBC Thin Driver to connect to the database. The value you specify is passed to the `setURL` method of Oracle JDBC Thin Driver. When a value is set in this property, the value specified in `databaseName`, `portNumber`, and `serverName` is ignored. Specifies a `thin` driver in JDBC URL, when the user specifies a url. (Example) `jdbc:oracle:thin:@ServerA:1521:service1`

21 Asynchronous Parallel Processing of Threads

This chapter describes the asynchronous parallel processing of threads from `TimerManager` and `WorkManager` based on *Timer and Work Manager for Application Servers*.

21.1 Organization of this chapter

The following table describes the functionality and reference locations of the asynchronous parallel processing of threads.

Table 21–1: Functionality of the asynchronous parallel processing of threads

Functionality	Reference location
Overview of asynchronous parallel processing of threads	*21.2*
Asynchronous timer processing by using `TimerManager`	*21.3*
Asynchronous thread processing by using `WorkManager`	*21.4*

21.2 Overview of the asynchronous parallel processing of threads

With Application Server, you can execute the asynchronous parallel processing of threads such as the asynchronous timer processing or the asynchronous thread processing in a Java EE environment.

With the standard specifications of Java EE, a new thread cannot be generated from a servlet, or EJB cannot manage threads. We basically do not recommend the asynchronous parallel processing of threads. Therefore, with Application Server, APIs are provided based on *Timer and Work Manager for Application Servers* specifications defined by `CommonJ`, to implement the asynchronous parallel processing of threads in the Java EE environment.

The following subsections give an overview of APIs used for implementing the asynchronous parallel processing of threads:

- `TimerManager`

  `TimerManager` is an API based on the *Timer for Application Servers* specifications. With this API, you can schedule the asynchronous processing of threads by specifying an execution interval. This functionality is called *asynchronous timer processing*.
- `WorkManager`

  `WorkManager` is an API based on the *Work Manager for Application Servers* specifications. With this API, you can perform the asynchronous processing of threads. This functionality is called *asynchronous thread processing*.

You can use `TimerManager` and `WorkManager` from EJBs or servlets.

For details on the compatibility with *Timer and Work Manager for Application Servers* on Application Server, see *21.2.3 Compatibility with Timer and Work Manager for Application Servers*.

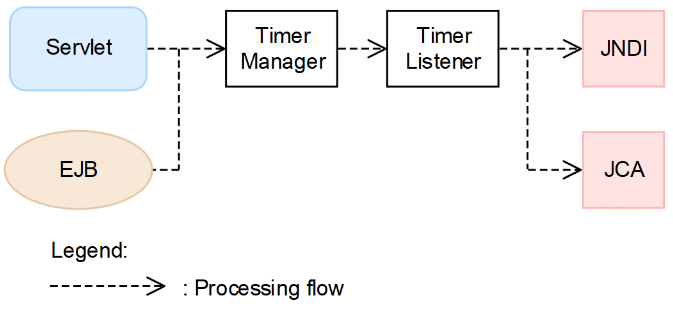
21.2.1 Procedure for the asynchronous parallel processing of threads

To execute the asynchronous parallel processing of threads, perform the lookup of `TimerManager` or `WorkManager` from EJBs and servlets. This section describes the flow of the asynchronous timer processing by using `TimerManager` and the flow of the asynchronous thread processing by using `WorkManager`.

Flow of the asynchronous timer processing by using TimerManager

The following figure shows the flow of the asynchronous timer processing by using `TimerManager`.

Figure 21–1: Flow of the asynchronous timer processing by using TimerManager

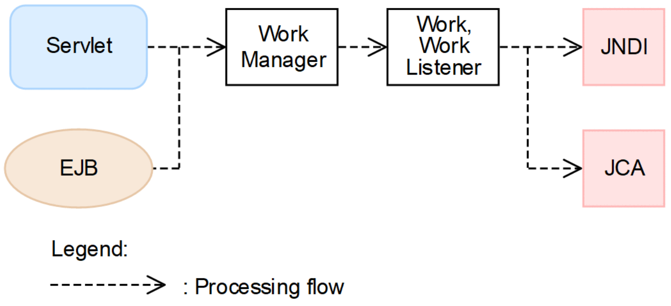


EJBs and servlets are the sources of the schedule for invoking the asynchronous parallel processing to be executed. `TimerManager` is created when JNDI performs a lookup. You implement an entity of the processing to be executed, in `TimerListener`, which is a listener provided by `TimerManager`. `TimerListener` executes the processing by accessing JNDI or JCA, as and when required.

Flow of the asynchronous thread processing by using WorkManager

The following figure shows the flow of the asynchronous thread processing by using `WorkManager`.

Figure 21–2: Flow of the asynchronous thread processing by using WorkManager



EJBs and servlets are the sources of the schedule for invoking the asynchronous parallel processing to be executed. `WorkManager` is created when an application starts. If a lookup is performed by JNDI, the `WorkManager` created when the application starts is returned. You implement the entity of the processing to be executed, in `Work` or `WorkListener` provided by `WorkManager`. `Work` or `WorkListener` executes the processing by accessing JNDI or JCA, as and when required.

21.2.2 Java EE functionality that you can use in the asynchronous parallel processing of threads

You can use the Java EE functionality in processes that are executed as the asynchronous parallel processing. APIs of `TimerManager` and `WorkManager`, which can use the Java EE functionality, are as follows:

`TimerManager`

- `TimerListener.timerExpired`

  This method is executed when reaching the set up time.

- `StopTimerListener.timerStop`

  This method is executed when the `TimerManager.stop` method is executed or when the application stops.

- `CancelTimerListener.timerCancel`

  The method is executed when the `TimerManager.cancel` method is executed.

`WorkManager`

- `Work.run`

  This is a processing method, which is asynchronously executed on `WorkManager`.

- `WorkListener.workAccepted`

  This method is executed when `WorkManager` receives the scheduled `Work`.

- `WorkListener.workCompleted`

  This method is executed immediately after completing the `run` method of the scheduled `Work`.

- `WorkListener.workRejected`

  This method is executed when you cannot continue the schedule processing, after `WorkManager` receives the scheduled `Work`.

- `WorkListener.workStarted`

  This method is executed immediately before executing the `run` method of the scheduled `Work`.

For details on APIs, see *API specifications* for *Timer and Work Manager for Application Servers*.

The following table describes the Java EE functionality that you can use in `TimerManager` and `WorkManager`.

Table 21–2: Java EE functionality that you can use in TimerManager and WorkManager

Functionality	Usage status	Reference location
Invoking Enterprise Bean	N	--
Naming Service	Y#	(1)
Transaction service and resource connections	Y#	(2)
Log and trace output	Y	(3)
Using container extension library	Y	(4)
Method cancellation	N	--

Legend:

Y: Can be used

N: Cannot be used

--: Not applicable

#

However, you cannot use a part of the functionality. For details on the functionality that you can use, see the information given in the *Reference location* column.

The following sections classify and describe the functionality that you can use with `TimerManager` and `WorkManager`. The sections also describe the points to be considered when using the functionality.

(1) Naming Service

The following table describes whether the functionality provided as Naming Service can be used with `TimerManager` and `WorkManager`.

Table 21–3: Usage status of Naming Service functionality

Functionality	Usage status
Lookup of DB Connector by using JNDI	Y
Lookup of Java Mail by using JNDI	N
Lookup of JavaBeans resource by using JNDI	N
Lookup of `EntityManager` by using JNDI	N
Lookup of `EntityManagerFactory` by using JNDI	N
Lookup of `TimerManager` by using JNDI	N#1
Lookup of `WorkManager` by using JNDI	N#1
Lookup of user transaction by using JNDI	Y#2

Legend:

Y: Can be used

N: Cannot be used

#1

You cannot invoke `TimerManager` or `WorkManager` by extending the schedule of `TimerManager` or `WorkManager`.

#2

If the schedule source is an EJB that manages transactions in CMT, you cannot perform a lookup with `java:comp/UserTransaction`. Make sure to perform a lookup by using `HITACHI_EJB/SERVERS/`*J2EE-server-name*`/SERVICES/UserTransaction`.

Important note

In `WorkManager` or `TimerManager`, do not use a DB Connector or user transaction acquired at a schedule source. Make sure to acquire the executed processes in the implemented `Timer Listener` or `Work`.

(2) Transaction service and resource connections

You can only use DB Connectors for resource adapters. The following table describes the DB Connectors that you can use with `TimerManager` and `WorkManager`.

Table 21–4: Usage status of DB Connectors

DB Connector name	Usage status
`DBConnector_HiRDB_Type4_CP.rar`	Y
`DBConnector_HiRDB_Type4_XA.rar`	Y
`DBConnector_Oracle_CP.rar`	Y
`DBConnector_Oracle_XA.rar`	Y
`DBConnector_HiRDB_Type4_CP_Cosminexus_RM.rar`	N
`DBConnector_HiRDB_Type4_XA_Cosminexus_RM.rar`	N
`DBConnector_Oracle_CP_Cosminexus_RM.rar`	N
`DBConnector_Oracle_XA_Cosminexus_RM.rar`	N
`DBConnector_CP_ClusterPool_Root.rar`	N
`DBConnector_Oracle_CP_ClusterPool_Member.rar`	N

Legend:

Y: Can be used

N: Cannot be used

When using a DB Connector, specify `NoTransaction`, `LocalTransaction`, or `XATransaction` for the transaction support level. You must specify an optional name of a DB Connector to acquire the connection of the DB Connector. With a lookup by JNDI, use the specified optional name and acquire the connection of the DB Connector. For details on how to acquire the connection by using the optional name of the DB Connector, see *2.6 Assigning an optional name to Enterprise Bean or the J2EE Server (user-specified name space functionality)* in the *uCosminexus Application Server Common Container Functionality Guide*.

The following table describes whether the functionality provided as a resource connection and transaction service can be used in `TimerManager` and `WorkManager`.

Table 21–5: Usage status of the transaction service functionality

Functionality		Usage status
Transaction (user transaction)	Local transaction	Y
	Global transaction	Y

Functionality		Usage status
Automatic conclusion of transaction[#1]		Δ
Transaction timeout		Y
Connection pooling	Connection pooling by using DB Connector	Y
	Warm-up of connection pool	Y
	Adjusting number of connections	Y
Connection sharing[#2]		Δ
Connection association		N
Statement pooling of DB Connector		Y
Detecting connection faults		Y
Waiting for acquiring connections when connections are exhausted		Y
Retrying acquisition of connections		Y
Automatically closing connections		N
Connection sweeper		Y
Output of SQL for examining faults		Y

Legend:

Y: Can be used

Δ: Some part of the functionality cannot be used

N: Cannot be used

#1

You must conclude the user transaction before returning from the Listener processing method. Otherwise, the transaction is rolled back even when an exception does not occur, and the message (KDJE43179-W) is output.

#2

The range of connections that you can share is only the *same transaction*, which is set by default.

Important note

Connections of the acquired DB Connector are not automatically closed, so make sure to set the closure of connections inside a method.

(3) Log and trace output

The following table describes whether the functionality that outputs logs and traces can be used in `TimerManager` and `WorkManager`.

Table 21–6: Usage status of the log and trace functionality

Functionality	Usage status
User log	Y
Performance analysis trace	Y

Legend:

Y: Can be used

Reference note

About the operation name of a performance analysis trace

With the performance analysis trace of `TimerManager` and `WorkManager`, you can acquire unique numbers for schedules. This information is output to the operation name of the trace information. For details on the trace information that you can acquire, see *8. Trace Collection Points and PRF Trace Collection Levels of the Trace Based Performance Analysis* in the *uCosminexus Application Server Maintenance and Migration Guide*.

(4) Using container extension library

You can use the same container extension library as in the case when `TimerManager` and `WorkManager` are not used.

21.2.3 Compatibility with Timer and Work Manager for Application Servers

The specifications stated as vendor dependent in *Timer and Work Manager for Application Servers Specifications* are not supported by Application Server. Also, the specifications of APIs provided by `CommonJ` and APIs provided by Application Server are different. This section describes both the *Timer for Application Server* specifications and *Work Manager for Application Server* specifications not supported by Application Server, and the APIs that operate differently with `CommonJ` and Application Server.

(1) Timer for Application Servers Specifications not supported by Application Server

The following table describes the Timer for Application Servers specifications not supported by Application Server.

Table 21–7: Timer for Application Server specification not supported by Application Server (Vendor dependent functionality)

Specifications not supported by Application Server	Remarks
Customization of the number of maximum scheduling	The number of maximum scheduling is `50`. You cannot change this number.
• The class that implements the listener of TimerManager • Objects (EJB or servlets) other than the general Java objects, which do not inherit Java EE components	An error occurs if you schedule a class that inherits `javax.ejb.EnterpriseBean`. The error check is not performed for other cases.
Inherited items of the transaction context to execution threads	No transaction is performed regardless of the transaction status of the schedule source. This corresponds to `NOT_SUPPORTED` of CMT.
Customization of the inherited items of the J2EE context to execution threads	The items that are inherited are fixed.
The Java EE functionality that can be used in an execution thread	For details on the functionality that you can use, see *21.2.2 Java EE functionality that you can use in the asynchronous parallel processing of threads*.

The components that you can use with J2EE applications are not defined in the Timer for Application Servers. For details on the components that you can use with the `TimerManager` for Application Server, see *21.3.5 Developing applications by using TimerManager*.

(2) Work Manager for Application Servers specifications not supported by Application Server

The following table describes the `Work Manager` for Application Servers specifications not supported by Application Server.

Table 21–8: Work Manager for Application Servers specifications not supported by Application Server (Vendor dependent functionality)

Specifications not supported by Application Server	Remarks
A remote execution of asynchronous thread processing by using `WorkManager`	If you remotely execute `WorkItem`, returns the dummy `RemoteWorkItem` that is executed locally.
Customizing the number of maximum scheduling	There is no limit for the number of maximum scheduling.
• The class that implements the listener of TimerManager • Objects (EJB or servlets) other than general Java objects that do not inherit Java EE components	An error occurs, if you schedule a class that inherits `javax.ejb.EnterpriseBean`. The error check is not performed in other cases.
Creating `WorkManager` at a timing other than the start of an application	`WorkManager` is created only when an application starts.
Inherited items of the transaction context to execution threads	No transaction is performed regardless of the transaction status of the source of schedule. This corresponds to `NOT_SUPPORTED of CMT`.
Customizing the inherited items of the J2EE context to execution threads	The inherited items are fixed.
The Java EE functionality that can be used in execution threads	For details on the functionality that you can use, see *21.2.2 Java EE functionality that you can use in the asynchronous parallel processing of threads*.

The components that you can use with a J2EE application are not defined with Work Manager for Application Servers. For details on the components that you can use with Work Manager for Application Server, see *21.4.4 Developing applications by using WorkManager*.

(3) APIs that operate differently with CommonJ and Application Server

The following table describes the APIs that operate differently with `CommonJ` and Application Server.

Table 21–9: APIs that operate differently with CommonJ and Application Server

Class	Method	Operation on Application Server
`commonj.timers.TimerManager`	`schedule(TimerListener listener,Date time)`	If the listener inherits `javax.ejb.EnterpriseBean`, `IllegalArgumentException` is returned.
	`schedule(TimerListener listener,long delay)`	If the listener inherits `javax.ejb.EnterpriseBean`, `IllegalArgumentException` is returned.
	`schedule(TimerListener listener,Date firstTime,long period)`	If the listener inherits `javax.ejb.EnterpriseBean`, `IllegalArgumentException` is returned.
	`schedule(TimerListener listener,long delay,long period)`	If the listener inherits `javax.ejb.EnterpriseBean`, `IllegalArgumentException` is returned.

Class	Method	Operation on Application Server
	`scheduleAtFixedRate(TimerListener listener,Date firstTime,long period)`	If the listener inherits `javax.ejb.EnterpriseBean`, `IllegalArgumentException` is returned.
	`scheduleAtFixedRate (TimerListener listener,long delay,long period)`	If the listener inherits `javax.ejb.EnterpriseBean`, `IllegalArgumentException` is returned.
`commonj.work.WorkManager`	`schedule(Work work)`	If work is null, `WorkException` is returned.
	`schedule(Work work,WorkListener wl)`	If work is null, `WorkException` is returned. If `WorkListener` inherits `javax.ejb.EnterpriseBean`, `IllegalArgumentException` is returned.

21.3 Asynchronous timer processing by using TimerManager

This section describes the asynchronous timer processing by using `TimerManager`.

The following table describes the organization of this section.

Table 21–10: Organization of this section (Asynchronous timer processing by using TimerManager)

Category	Title	Reference location
Description	Threads scheduling method by using `TimerManager`	*21.3.1*
	The life cycle of `TimerManager`	*21.3.2*
	The state transition of `TimerManager`	*21.3.3*
	Multiple schedules of `TimerManager`	*21.3.4*
Implementation	Developing applications by using `TimerManager`	*21.3.5*

Note:
There is no specific description of *Settings, Operations*, and *Notes* for this functionality.

With the asynchronous timer processing performed by using `TimerManager`, you can schedule the asynchronous processing of threads in a Java EE environment by specifying an execution interval. Threads managed by a container are used in the background, and hence you can safely execute tasks.

In `TimerListener`, you implement the process that performs scheduling. The processing implemented in `TimerListener` is scheduled by executing the method of `TimerManager` in EJBs or servlets, which are the schedule sources. You can respond to the schedule or cancel the schedule by using `Timer`, which is returned from the `schedule` method of `TimerManager`.

To use `TimerManager`, you define the information related to `TimerManager` in the `<resource-ref>` tag of an EJB attribute or a servlet attribute. An EJB or a servlet uses `TimerManager` by performing a lookup with the name defined in the `<res-ref-name>` tag at the time of deployment.

21.3.1 Methods of scheduling threads by using TimerManager

You use the following two methods for scheduling threads, by using `TimerManager`:

- Executing the process only once
- Executing and repeating the process at regular intervals

This section describes an overview of each scheduling method.

(1) Executing the process only once

This is a method that executes the processing only once at a specified time. After the processing is executed, `TimerManager` is destroyed.

(2) Executing and repeating the process at regular intervals

You use the following two methods to repeatedly execute the process at regular intervals:

- `fixed-rate` (Specify the interval for starting the process, and then repeatedly execute the process)
- `fixed-delay` (Specify the interval from the end of process to start of the next process, and then repeatedly execute the process)

The scheduled process continues to execute until you stop `TimerManager` or until the `cancel` method of the corresponding `Timer` is executed.

The following subsection gives an overview of each method:

fixed-rate (Specify interval for starting the process, and then repeatedly execute the process)

This is a method which repeatedly starts the process at regular intervals. Specify the following contents in `fixed-rate`:

Timing to start the first process

You use one of the following methods to specify the settings:

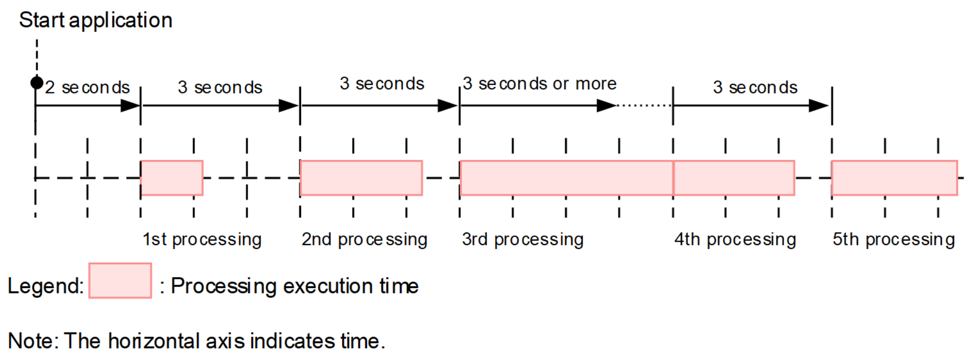
- When specifying the start time,
  Use the `Date` type format to specify the `firstTime` argument of the `scheduleAtFixedRate` method.
- When specifying the elapsed time from the start of an application until execution of the process
  Use the `long` type format to specify the `delay` argument of the `scheduleAtFixedRate` method. The unit is milliseconds.

Interval from the start of the previous process to the start of the next processing

Use the `long` type format to specify the specify `period` argument of the `scheduleAtFixedRate` method. The unit is milliseconds.

The following figure shows an image of `fixed-rate` process. In this figure, the time from the start of an application to the start of the first process is two seconds, and the time from the start of the previous process to the start of the next process is three seconds.

Figure 21–3: Image of fixed-rate processing



With the `fixed-rate` process, if the process time executed previously is longer than the time specified in the `period`, the next process is started immediately after the previously executed process ends. In this figure, the time for the third process is longer than three seconds as specified in `period`, and hence, the fourth process is started immediately after the third process ends.

fixed-delay (Specify an interval from the end of a process to the start of the next process, and then repeatedly execute the process)

This method repeatedly starts the process at regular intervals after the previous process ends. You specify the following contents in `fixed-delay`.

Timing to start the first process

Use one of the following methods to specify the timing:

- When specifying the start time

  Use the `Date` type format to specify `firstTime`, an argument of the `schedule` method.
- When specifying elapsed time from the start of application until execution of the processing

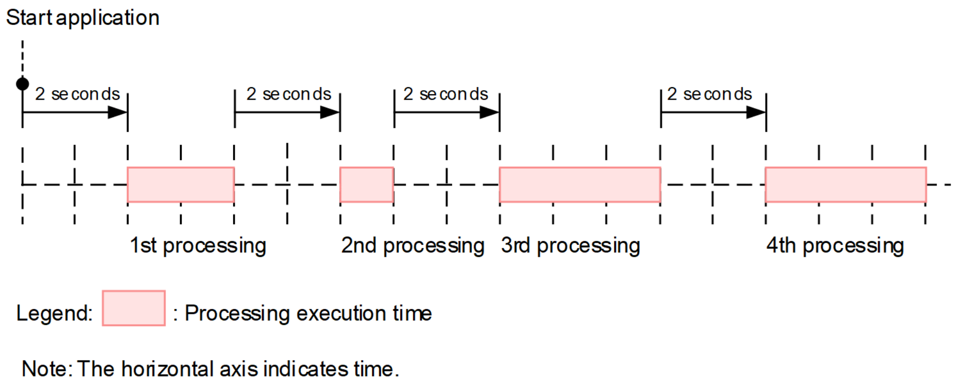
  Use the `long` type to specify `delay`, an argument of the `schedule` method. The unit is milliseconds.

Interval from completion of previous processing to start of next processing

Use the `long` type to specify `period`, an argument of the `schedule` method. The unit is milliseconds.

The following figure shows the image of the `fixed-delay` process. In this figure, the time from the start of an application to the start of the first process and the time from the end of the previous process to the start of the next process is considered as two seconds.

Figure 21–4: Image of the fixed-delay process



21.3.2 Life cycle of TimerManager

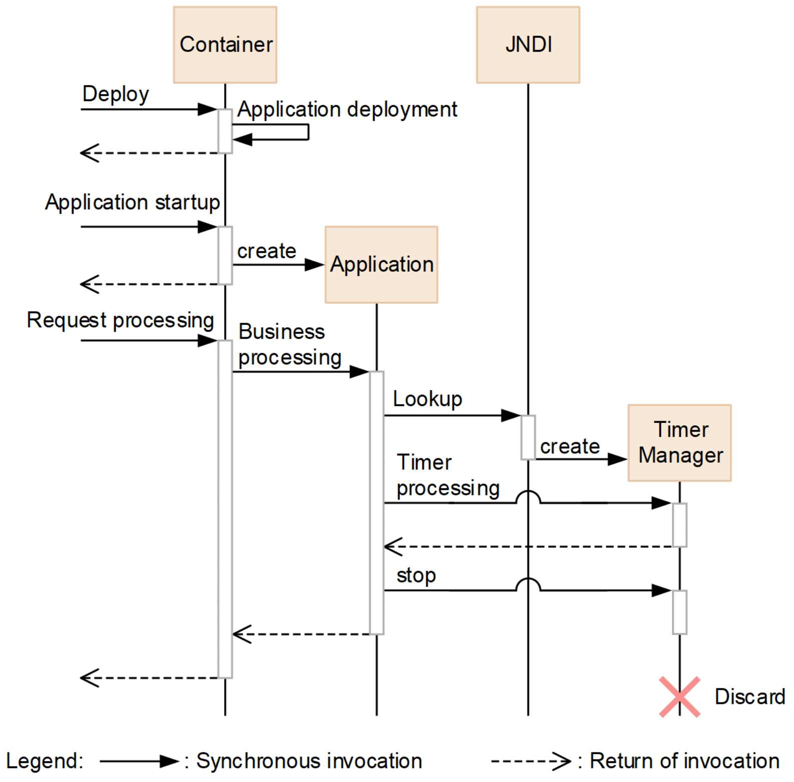
This section describes a life cycle of `TimerManager`.

`TimerManager` is created when lookup is performed by JNDI in an application. `TimerManager` is created for each lookup. We recommend that you execute the `stop` method in the application and explicitly stop the created `TimerManager`. `TimerManager` can be automatically stopped without executing the `stop` method. However, in that case, the application does not stop until `TimerManager` stops. Therefore, stopping the application might take a longer time, depending on the stop process of `TimerManager`.

`TimerManager` is not persisted. As a result, when JavaVM ends, the created `TimerManager` and scheduled timer are destroyed.

The following figure shows the life cycle of `TimerManager`.

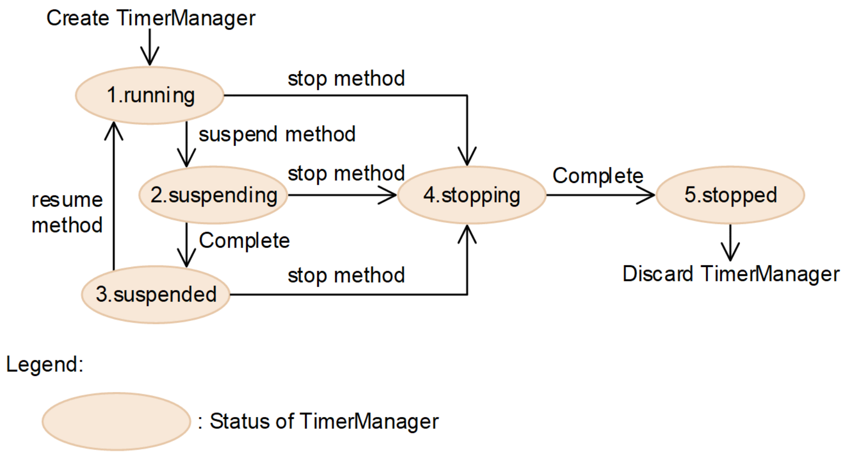
Figure 21–5: Life cycle of TimerManager



21.3.3 State transition of TimerManager

The status of `TimerManager` changes depending on the lock or stop process, by suspend and resume. You can check the status of `TimerManager` at the respective time, by using the `isStopped`, `isStopping`, and `isSuspended` method. The following figure shows the status transition of `TimerManager`.

Figure 21–6: Status transition of TimerManager



The following table describes the details of each status.

Table 21–11: Status of TimerManager

Number in figure#	Status	Explanation
1	running	It is a status indicating that TimerManager is running. You can receive and execute a new schedule.
2	suspending	It is a status indicating that suspension is in process. This status shows that a task is being executed when suspension is executed. If no task is being executed, the status transits to suspended status.
3	suspended	It is a status indicating that all the tasks are suspended. When the status is suspended, all the scheduled tasks are not executed. The tasks with suspended status are executed when those tasks are resumed.
4	stopping	It is a status indicating that the stopping TimerManager is being executed. This status shows that a task is being executed when stopping is TimerManager is being executed. If no task is being executed, the status transits to stopped status.
5	stopped	It is a status indicating that TimerManager is stopped. This status shows that all the tasks are stopped and no process is executed after that. You cannot resume TimerManager, once it is stopped.

#
Number indicating the number in *Figure 21-6*.

21.3.4 Multiple schedules of TimerManager

With the timer process scheduled by TimerManager, use the threads managed in the thread pool. When the timer processing is scheduled, one thread, from among the threads managed in the thread pool, is assigned. If there is a blank thread in the thread pool, the blank thread is used. If there is no blank thread in the thread pool, a thread is generated and used. A thread generated in the thread pool is pooled until TimerManager stops.

The maximum number of threads that you can concurrently use in an instance is 50. A thread is assigned even if the scheduled timer processing is in standby status. Therefore, the maximum number of processes that can be concurrently scheduled is 50, irrespective of the status of timer processing.

If the number of already generated threads reaches the maximum number, the scheduled timer process is stored in a queue and process waits until a blank thread is available. The timer process, stored in the queue, is executed as soon as a blank thread is available.

Use multiple TimerManager if you want to concurrently schedule 51 or more threads.

21.3.5 Developing applications by using TimerManager

This section describes development of applications by using TimerManager.

The following table describes the usage status of components, which configure the application, when using TimerManager.

Table 21–12: Usage status of components, which configure the application, when using TimerManager

Component	Usage status
EJB client	N
Resource adapter	N

Component				Usage status
JavaBeans resources				N
Servlet/JSP#				Y
EJB	Stateless Session Bean	EJB2.1 or earlier versions	CMT	Y
			BMT	Y
		EJB3.0		N
	Stateful Session Bean			N
	Entity Bean			N
	Message-driven Bean			N

Legend:

Y: Can be used

N: Cannot be used

#

You can use the components also with the servlet listener or the filter.

The flow of developing an application by using `TimerManager` is as follows:

1. Defining the properties of EJBs or servlets, which are the schedule sources
2. Implementing the processing to be executed in the listener of `TimerManager`
3. Creating EJBs or servlets, which are the schedule sources
4. Compiling J2EE applications which use `TimerManager`

Details of each task are as follows:

(1) Defining the properties of EJBs or servlets, which are the schedule sources

Define properties of EJBs or servlets, which use `TimerManager`, in the DD. You cannot implement the definition for using `TimerManager`, in annotation.

The following table describes the properties, which you must define to use TimerManager.

Table 21–13: Properties, which you must define to use TimerManager

Tag name		Explanation
<Root-tag>		--
	`<description>`	Set optionally.
	`<res-ref-name>`	Specify the JNDI ENC name (name to be used for the JNDI lookup).
	`<res-type>`	Set the following value. `commonj.timers.TimerManager`
	`<res-auth>`	The set value is ignored.
	`<res-sharing-scope>`	Set `Unshareable`. However, even if you set `Shareable`, the same operation as for `Unshareable` is executed (new `TimerManager` is created whenever you perform lookup).

Tag name		Explanation
	`<mapped-name>`	The set value is ignored.
	`<injection-target>`	The set value is ignored.
	`<linked-to>`	The set value is ignored.

The definition example of `web.xml` when you use `TimerManager` in servlet is as follows.

```
<web-app>
  <display-name>TimerManagerSample</display-name>
  <servlet>
    <servlet-name>SampleServlet</servlet-name>
    <display-name>SampleServlet</display-name>
    <servlet-class>SampleServlet</servlet-class>
  </servlet>
  ...
  <resource-ref>
    <res-ref-name>timer/MyTimer</res-ref-name>
    <res-type>commonj.timers.TimerManager</res-type>
    <res-auth>Container</res-auth>
    <res-sharing-scope>Unshareable</res-sharing-scope>
  </resource-ref>
</web-app>
```

`TimerManager` is created whenever a lookup by JNDI is performed in an application. Created `TimerManager` is destroyed when you execute the `stop` method or end the application. You can define multiple `TimerManager`, as and when required.

(2) Implementing the processing to be executed in the listener of TimerManager

To use `TimerManager`, you must create a listener, with which the process to be executed is implemented. Listener interfaces consist of- an interface that must be implemented and interfaces to be implemented as and when required. The interface that must be implemented and the interfaces to be implemented if required are as follows:

Interface that must be implemented

- `TimerListener`

Interfaces to be implemented as and when required

- `StopTimerListener`
- `CancelTimerListener`

For details on APIs, see *API specifications* in *Timer and Work Manager for Application Servers*.

The example of a class where `TimerListener`, `StopTimerListener`, and `CancelTimerListener` are implemented is as follows.

```
public class MyTimerListener
        implements TimerListener,StopTimerListener, CancelTimerListener {
    private int count = 0;

        public MyTimerListener() {
        }
        public void timerStop(Timer timer) {
                System.out.println("Timer stopped:  " + timer);
        }

        public void timerCancel(Timer timer) {
                System.out.println("Timer cancelled:  " + timer);
        }

        public void timerExpired(Timer timer) {
                System.out.println("Timer expired !");
                    if(count++ > 10) {
                //Canceled because the set count is reached
                timer.cancel();
            } else {
             System.out.println("The next timer will fire at : " +
                                   timer.getScheduledExecutionTime());
            }
        }
}
```

(3) Creating EJBs or servlets, which is the schedule source

To use `TimerManager`, implement lookup of the JNDI name of `TimerManager`, which is defined in properties, and the process scheduling of `TimerManager`, in EJBs or servlets, which are the schedule sources.

Lookup by using JNDI of TimerManager, which is defined in properties

Perform lookup for the JNDI name of `TimerManager`, which is defined in properties, to acquire `TimerManager`. Use `java:comp/env` for lookup. The example of acquiring `TimerManager` is as follows:

```
InitialContext ic = new InitialContext();
TimerManager tm = (TimerManager)ic.lookup
                     ("java:comp/env/timer/MyTimer");
```

Scheduling the TimerManager processes

Schedule the `TimerManager` processes by invoking the `schedule` method of `TimerManager`. The example of scheduling the `TimerManager` processes is as follows:

```
InitialContext ctx = new InitialContext();
TimerManager mgr = (TimerManager)
              ctx.lookup("java:comp/env/timer/MyTimer");
TimerListener listener = new MyTimerListener();
mgr.schedule(listener, 1000*60,1000*10);
mgr.stop();
```

(4) Compiling J2EE applications, which use TimerManager

Include the following JAR file for compiling J2EE applications, which use `TimerManager`.

Cosminexus-installation-directory`\CC\lib\ejbserver.jar`

21.4 Asynchronous thread processing by using WorkManager

This section describes the asynchronous thread processing performed by using `WorkManager`.

The following table describes the organization of this section.

Table 21–14: Organization of this section (asynchronous thread processing by using WorkManager)

Category	Title	Reference location
Description	The daemon `Work` and non-daemon `Work`	*21.4.1*
	The thread pool and queues used in the non-daemon `Work`	*21.4.2*
	The life cycle of `WorkManager`, daemon `Work` and the non-daemon `Work`	*21.4.3*
Implementation	Developing applications by using `WorkManager`	*21.4.4*
Settings	Settings in the execution environment	*21.4.5*

Note:

There is no specific description of *Operations* and *Notes* for this functionality.

With the asynchronous thread processing performed by using `WorkManager`, you can execute the asynchronous processing of threads in the Java EE environment. Because the threads managed by a container are used in the background, you can execute tasks safely.

Implement the process to be executed asynchronously, with `Work`. The process implemented with `Work` is scheduled when you execute the `schedule` method of `WorkManager` in EJBs or servlets, which are the schedule sources. You can check the schedule status by using `WorkItem`, which is returned by the `schedule` method of `WorkManager`.

To use `WorkManager`, define the information related to `WorkManager`, in `<resource-ref>` tag of EJB properties or servlet properties. The EJB or servlet uses `WorkManager` by performing lookup with the name defined in `<res-ref-name>` tag at the time of deployment.

21.4.1 Daemon Work and non-daemon Work

In `WorkManager`, you can create two types of `Work` such as the daemon `Work` (long-life `Work`) and the non-daemon `Work` (short-life `Work`). An overview of each `Work` is as follow:

- **Daemon Work (long-life Work)**
  The daemon `Work` is created when you execute the `schedule` method and `Work` continues even if the request processing of a servlet or EJB ends. The daemon `Work` is destroyed when `WorkManager` ends. The daemon `Work` is always executed with a newly created thread and not with threads in the thread pool.
- **Non-daemon Work (short-life Work)**
  The non-daemon `Work` is created when you execute the `schedule` method and `Work` is destroyed when processing of the `run` method ends. For the non-daemon `Work`, use threads and queues that are managed in the thread pool.

21.4.2 Thread pool and queues used in non-daemon Work

The non-daemon `Work` is processed using the thread pool and queues. The thread pool and queues used for the process are created in the unit of `WorkManager`, which is defined in the DD. Set the maximum size of threads that can be pooled in a thread pool. The following section describes the maximum size of threads that can be pooled and relation and operation of the number of threads in a pool, when the non-daemon `Work` is scheduled.

- If threads in a pool are less than the maximum number of threads in a thread pool

  Create a thread and execute the non-daemon `Work`. The thread is generated irrespective of whether any blank thread exists in thread pool.
- If a pool contains threads of the same number as the number of the maximum threads in a thread pool

  Use blank threads in the thread pool and execute the non-daemon `Work`. If no blank thread exists, the scheduled non-daemon `Work` is stored in queue. The non-daemon `Work`, which is stored in the queue, is executed when a blank thread is available.

The maximum number of threads in a thread pool is `10` by default. To change the maximum number of threads, see *21.4.5 Settings in the execution environment*. There is no limit for a queue size.

Tip

When you attempt to stop `WorkManager`, the stop process starts after `WorkManager` being executed and all the `WorkManager` processes stored in the queue end. `WorkManager`, which is stored in the queue is executed even if `WorkManager` is stopped when storing in a queue.

21.4.3 Life cycle of WorkManager, daemon Work and non-daemon Work

This section describes the life cycle of `WorkManager`, the daemon `Work` and the non-daemon `Work`.

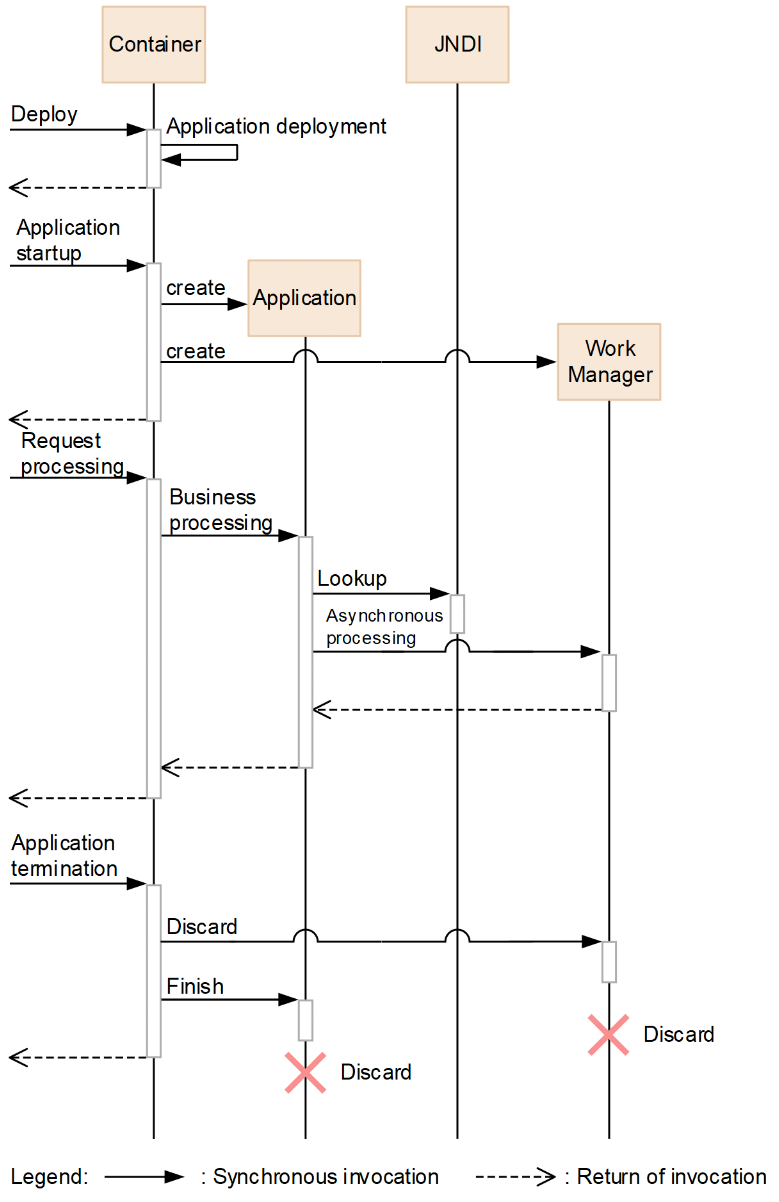
(1) Life cycle of WorkManager

`WorkManager` is created when an application starts. When lookup is performed in an application, `WorkManager`, created when the application starts, is returned. The same `WorkManager`, created when the application starts, is invoked even if lookup is performed for multiple times. `WorkManager` is destroyed when the application stops.

`WorkManager` is not persisted. As a result, when JavaVM ends, created `WorkManager` and the scheduled asynchronous process are destroyed.

The following figure shows the life cycle of `WorkManager`.

Figure 21–7: Life cycle of WorkManager

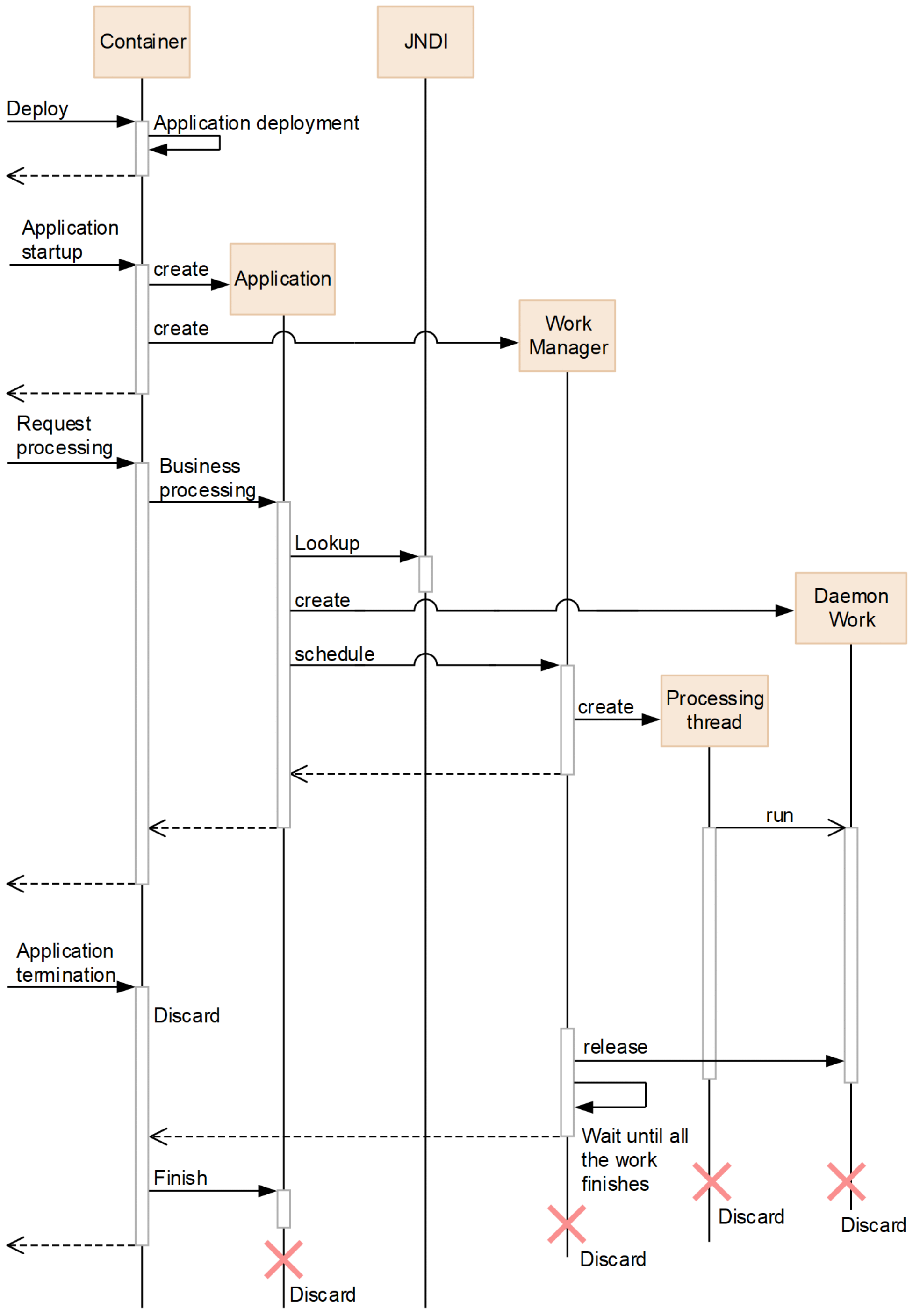


(2) Life cycle of daemon Work

A daemon `Work` is created when you execute the `schedule` method. The daemon `Work` is destroyed when you stop `WorkManager` (when you stop the applications corresponding to `WorkManager`). When you stop `WorkManager`, `WorkManager` waits until all daemon `Work` end after executing the `release` method of the daemon `Work`.

The following figure shows the life cycle of the daemon `Work`.

Figure 21–8: Life cycle of the daemon Work



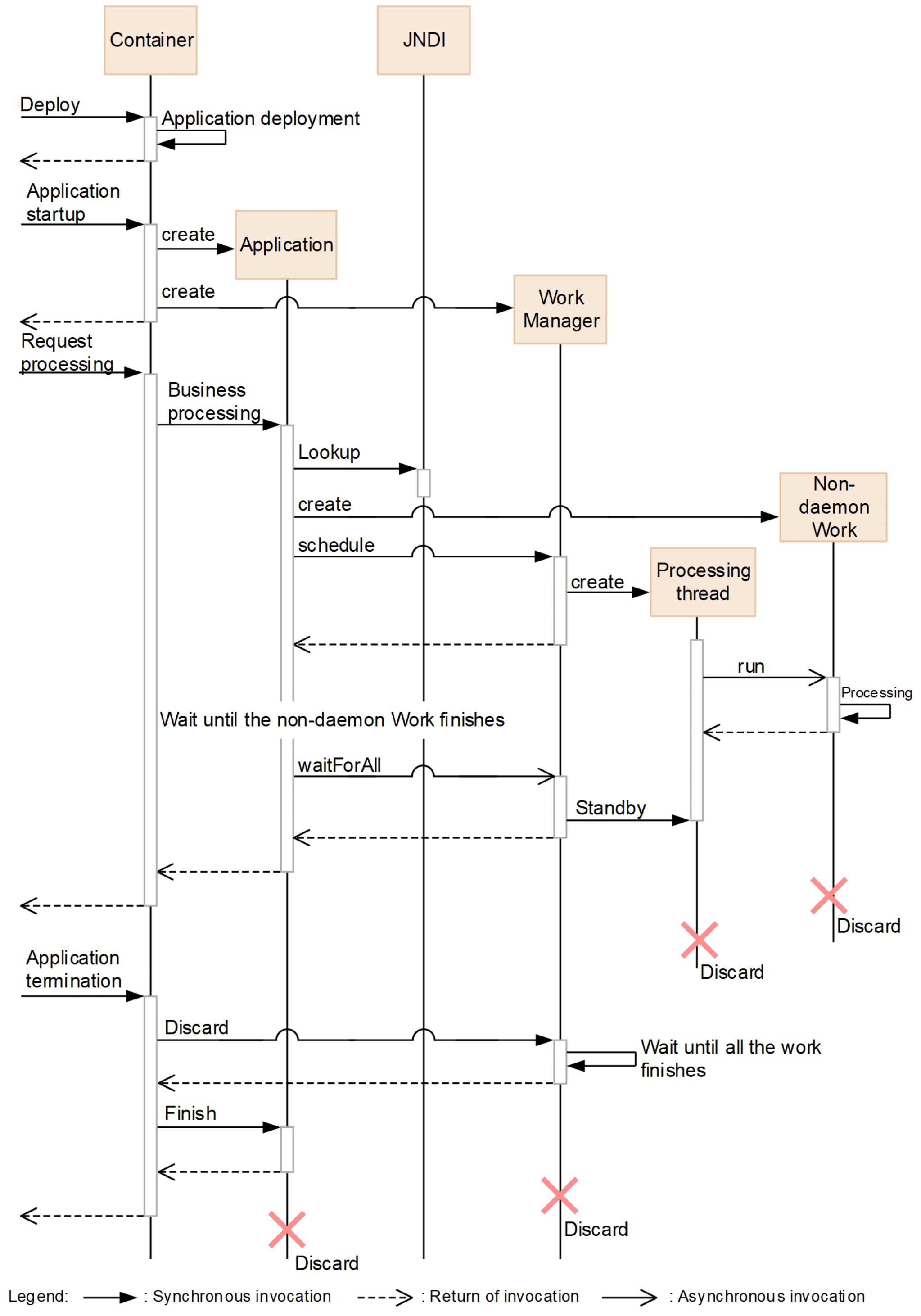
Legend:
→ : Synchronous invocation ---> : Return of invocation → : Asynchronous invocation

(3) Life cycle of the non-daemon Work

A non-daemon `Work` is created when you execute the `schedule` method. The non-daemon `Work` ends when processing of the `run` method ends. If you want to stop `WorkManager` (stop the corresponding applications) when non-daemon `Work` is being executed or is pending in a queue, wait until the non-daemon `Work` stops and then end `WorkManager`.

The following figure shows the life cycle of the non-daemon `Work`.

Figure 21–9: Life cycle of non-daemon Work



21.4.4 Developing applications by using WorkManager

This section describes development of applications by using `WorkManager`.

The following table describes the usage status of components, which configure an application, when using `WorkManager`.

Table 21–15: Usage status of components, which configure an application, when using WorkManager

Component				Usage status
EJB client				N
Resource adapter				N
JavaBeans resources				N
Servlet/JSP[#]				Y
EJB	Stateless Session Bean	EJB2.1 or earlier versions	CMT	Y
			BMT	Y
		EJB3.0		N
	Stateful Session Bean			N
	Entity Bean			N
	Message-driven Bean			N

Legend:

Y: Can be used

N: Cannot be used

#

You can use the components also for the servlet listener or the filter.

The procedure for developing an application by using `WorkManager` is as follows:

1. Defining the properties of EJBs or servlets, which are the schedule sources
2. Implementing the processes to be executed in `Work` and `Listener`
3. Creating EJBs or servlets, which are the schedule sources
4. Compiling J2EE applications which uses `WorkManager`

Details of each task are as follows.

(1) Defining the properties of EJBs or servlets, which are the schedule sources

Define EJB or servlet properties, which use `WorkManager`, in the DD. Define the properties in property definition file of EJBs or servlets. You cannot define the properties in an annotation.

The following table describes the properties, which you must define to use `WorkManager`.

Table 21–16: Properties, which you must define to use WorkManager

Tag name		Explanation
<Root-tag>		--
	`<description>`	Set optionally.
	`<res-ref-name>`	Specify the JNDI ENC name (name to be used for JNDI lookup).
	`<res-type>`	Set the following value. `commonj.work.WorkManager`

Tag name		Explanation
	`<res-auth>`	The set value is ignored.
	`<res-sharing-scope>`	Set `Shareable`. However, even if you set `Unshareable`, the same operation as for `Shareable` is executed (WorkManager is created when application starts and the same `WorkManager` is returned when lookup is performed).
	`<mapped-name>`	The set value is ignored.
	`<injection-target>`	The set value is ignored.
	`<linked-to>`	The set value is ignored.

The definition example of `web.xml` when you use `WorkManager` in the servlet is as follows:

```
<web-app>
  <display-name>WorkManagerSample</display-name>
  <servlet>
    <servlet-name>SampleServlet</servlet-name>
    <display-name>SampleServlet</display-name>
    <servlet-class>SampleServlet</servlet-class>
  </servlet>
  <resource-ref>
    <res-ref-name>wm/MyWorkManager</res-ref-name>
    <res-type>commonj.work.WorkManager</res-type>
    <res-auth>Container</res-auth>
    <res-sharing-scope>Shareable</res-sharing-scope>
  </resource-ref>
</web-app>
```

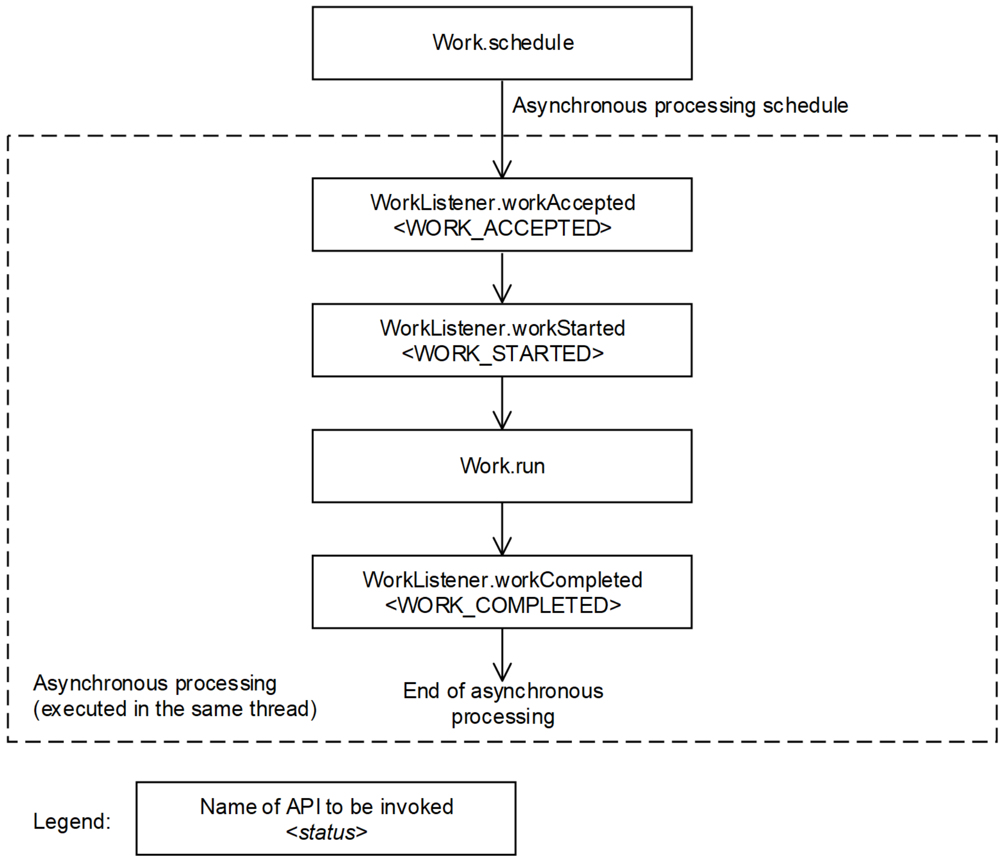
`WorkManager` is automatically created depending on property definitions when you start the application. The number of `WorkManager`, defined in the property, is created.

(2) Implementing the processes to be executed in Work and Listener

To use `WorkManager`, you must create `Work` and the listener, with which the processing to be executed is implemented. There are two types of interfaces - `Work` interface having the `run` method, which is a process entity, and `WorkListener` interface used to execute the process at the times such as process reception, start and end. Among these interfaces, make sure to implement `Work`. For details on APIs, see the *API specifications* in *Timer and Work Manager for Application Servers*.

The following figure shows the procedure which is invoked by API of the `WorkListener` interface and the status transition.

Figure 21–10: Procedure invoked by API of the WorkListener interface and the state transition


Work.schedule
Asynchronous processing schedule
WorkListener.workAccepted <WORK_ACCEPTED>
WorkListener.workStarted <WORK_STARTED>
Work.run
WorkListener.workCompleted <WORK_COMPLETED>
Asynchronous processing (executed in the same thread)
End of asynchronous processing
Legend:
Name of API to be invoked <*status*>

The `WorkListener` method and the `Work.run` method are invoked in the same thread. As a result, you can use the common Java EE context for each method.

The following section describes the flow and implementation example of the processes executed in the daemon `Work` and the non-daemon `Work`, and also the implementation example of `WorkListener`.

The flow and implementation example of the processes executed in the daemon Work

To use the daemon `Work`, implement `Work` in such a way that the `isDaemon` method returns `true`.

When `WorkManager` ends, the container executes the `release` method to stop the daemon `Work`. Therefore, implement in such a way that process of the `run` method ends when the `release` method is executed. Note that if the method is not implemented properly, the daemon `Work` might not stop when you stop `WorkManager` and continues to wait endlessly.

The implementation example of the daemon `Work` is as follows:

```
public class MyWork implements Work {
  private String name;
  private boolean isLoopContinue = true;
  public MyWork() {}

  public void release() {
    isLoopContinue = false;
  }

  public boolean isDaemon() {
    return true;
  }

  public void run() {
    while (isLoopContinue) {
      System.out.println("DaemonWork is executed");
   try {
     Thread.sleep(10000);
   } catch(InterruptedException e) {}
 }
  }
  public String toString() {
    return name;
  }
}
```

The flow and the implementation example of processes executed in the non-daemon Work

To use the non-daemon `Work`, implement `Work` in such a way that the `isDaemon` method returns `false`.

The processes of non-daemon `Work` must end during scheduled processes of EJBs or servlets. Therefore, implement the process in such a way that the EJB or servlet process ends after waiting for the scheduled work to end. To wait for the end of the scheduled `Work`, use the `waitForAll` or `waitForAny` method. If the process of EJBs or servlets ends before the end of the scheduled `Work`, the `Work` process is executed beyond the life cycle of the scheduled EJB or servlet. Make sure to end the process in a user program by using methods such as the `waitForAll` method, so that the non-daemon `Work` is not executed beyond the life cycle of the scheduled request.

The implementation example of the non-daemon `Work` is as follows:

```
public class MyWork implements Work {
         private String name;
         private String data;
         public MyWork(String name) {
                  this.name = name;
}

         public void release() {}

         public boolean isDaemon() {
                  return false;
         }

         public void run() {
                  data = "Hello, World. name=" + name;
         }

         public String getData() {
                  return data;
         }

         public String toString() {
                  return name;
         }
}
```

The implementation example of WorkListener

The implementation example of `WorkListener` is as follows:

```
public class ExampleListener implements WorkListener {
    public void workAccepted(WorkEvent we) {
        System.out.println("Work Accepted");
    }
    public void workRejected(WorkEvent we) {
        System.out.println("Work Rejected");
    }

    public void workStarted(WorkEvent we) {
        System.out.println("Work Started");
    }

    public void workCompleted(WorkEvent we) {
        System.out.println("Work Completed");
    }
}
```

(3) Creating EJBs or servlets, which are the schedule sources

To use `WorkManager`, implement lookup of the JNDI name of `WorkManager`, which is defined in properties, and the process scheduling of `WorkManager`, in EJBs or servlets, which are the schedule sources.

The JNDI name of WorkManager defined in properties

Perform lookup for the JNDI name of `WorkManager`, which is defined in properties, to acquire `WorkManager`. Use `java:comp/env` for lookup. The example of acquiring `WorkManager` is as follows.

```
   InitialContext ic = new InitialContext();
   WorkManager tm = (WorkManager)ic.lookup
                        ("java:comp/env/wm/MyWorkManager");
```

Scheduling the WorkManager process

Execute the scheduling of the `WorkManager` process by invoking the `schedule` method of `WorkManager`.

An example of a program which waits for of all `Work` to end, after scheduling multiple non-daemon `Work` is as follows.

```
MyWork work1 = new MyWork();
MyWork work2 = new MyWork();
InitialContext ctx = new InitialContext();
WorkManager mgr = (WorkManager) ctx.lookup("java:comp/env/wm/MyWorkManager
");
WorkItem wi1 = mgr.schedule(work1, new ExampleListener());
WorkItem wi2 = mgr.schedule(work2);
Collection coll = new ArrayList();
coll.add(wi1);
coll.add(wi2);
mgr.waitForAll(coll, WorkManager.INDEFINITE);

System.out.println("work1 data:  " + work1.getData());
System.out.println("work2 data:  " + work2.getData());
```

(4) Compiling the J2EE application, which uses WorkManager

Include the following JAR file when you compile the J2EE application, which uses `WorkManager`.

Cosminexus-installation-directory`\CC\lib\ejbserver.jar`

21.4.5 Settings in the execution environment

If you want to change the maximum number of threads in the thread pool, which is used in non-daemon `Work`, from the default value `10`, you must perform J2EE server settings.

Perform J2EE server settings in the Easy Setup definition file. Specify the definition of the maximum number of threads in a thread pool, in the `<configuration>` tag of the logical J2EE Server (j2ee-server) in the Easy Setup definition file. The following table describes the settings in the Easy Setup definition file.

Table 21–17: Definition for changing the maximum number of threads in a thread pool, defined in the Easy Setup definition file

Parameter to be specified	Setting details
`ejbserver.commonj.WorkManager.non_daemon_work_threads`	Set the maximum number of threads in a thread pool, which are used in the non-daemon `Work`. Set the value in the range of 1 through 65535#. The default value is `10`.

#

If you specify a number, which is out of range, the KDJE34510-W message is displayed and the default value is used.

For details on the Easy Setup definition file and parameters, see *4.3 Easy Setup definition file* in the *uCosminexus Application Server Definition Reference Guide.*

Appendixes

A. Installing the redirector functionality

The redirector functionality installation methods are as follows:

- **New installation**
  With this method, you install the redirector functionality in an environment where the redirector functionality is not installed. With a new installation, the installer creates the default user definition for the user definition of the redirector.
- **Update installation**
  With this method, you install the redirector functionality (same product) in an environment where the redirector functionality is already installed. With an update installation, the user definition of the redirector is inherited.

A.1 Installing the redirector functionality (In Windows)

Connect to Application Server to install the redirector functionality. Use the product media, installer, for installation. For details on how to use an installer, see the documentation for the product media.

This appendix describes the installation procedure. The installation operation requires Administrator permissions. For details on precautions when installing the functionality, see *Appendix I.1 Notes on the installation and un-installation of Application Server* in the manual *uCosminexus Application Server System Setup and Operation Guide.*

Procedure

1. Set the product media in the CD-ROM drive.
   In the Hitachi Integrated Installer dialog box, **The selected software will be installed.** appears.
   If the Hitachi Integrated Installer dialog box does not appear, use the explorer, and then double click **HCD_INST.EXE** in the CD-ROM directory.

   Important note

   If you copy the product media to the hard disk and then install the product, make sure the copy destination path name does not include semicolon (`;`), period (`.`), and double-byte characters.

2. Select Application Server, and then click the **Install** button.
   In the Confirming Start of Installation - Hitachi Integrated Installer dialog box, **Installation will start now. Do you want to continue?** appears.
3. Click the **OK** button.
   The Welcome to uCosminexus Application Server Setup Program dialog box appears.
4. Click the **Next** button.
   The Select Install Destination dialog box appears.
5. As and when required, select the **Install destination folder**, and then click the **Next** button.
   The Select Functionality dialog box appears.
6. Click the button on the left side of one of the following:
   - For a new installation: **Redirector - This setup is for installing Redirector. All specifiable options can be customized.**

- For an update installation: **Redirector - Installs Redirector, and you can select the program components to be reinstalled.**

The Select Program dialog box appears.

7. Select the component software (program) you want to install, and then click the **Next** button.
   Select from the following component software:
   - Component Container - Redirector (redirector functionality)
   - HTTP Server (Web server)
   - Performance Tracer

   The User Information dialog box appears.
8. Enter the User name and Company name, and then click the **Next** button.
   The Select Program Folder dialog box appears.
9. As and when required, change the **Program folder**, and then click the **Next** button.
   The Start Installation dialog box appears.
10. Check the specified contents, and if the contents are correct, click the **Next** button.
    The installation starts. When the installation finishes, the Setup Complete dialog box appears.
11. Click the **Finish** button.
    A window asking whether you want to restart the OS appears.
12. Click the **Yes** button.
    The OS restarts and the installation of the redirector functionality finishes.

A.2 Installing the redirector functionality (In UNIX)

Connect to Application Server to install the redirector functionality.

Use the product media, PP installer, for installation.

This appendix describes the installation procedure. The installation operation requires the root permission. For details on precautions when installing the functionality, see *Appendix I.1 Notes on the installation and un-installation of Application Server* in the manual *uCosminexus Application Server System Setup and Operation Guide*.

Procedure

1. Log in to Application Server with the root permission (superuser).
2. Check whether the language type for executing the PP installer matches with the language on the terminal where you want to execute the installer, and make sure that the languages match.
3. Set the product media in the CD-ROM drive.
4. If the product media is a CD-ROM, mount the CD-ROM file system.
   An example of command execution is as follows. The underlined part specifies the device special file name, and the mount directory name of the CD-ROM file system. Note that these names vary depending on the OS, hardware, and environment.

(Example of execution in AIX)

```
mount -r -v cdrfs /dev/cd0 /cdrom
```

(Example of execution in HP-UX)

```
mount -r -F cdfs /dev/dsk/c0t2d0 /cdrom
```

(Example of execution in Linux)

```
mount -r -o mode=0544 /dev/cdrom /mnt/cdrom
```

5. Start the setup program.

   An example of command execution is as follows. The underlined part specifies the mount directory name of the CD-ROM file system.

   (Example of execution in AIX)

   ```
   /cdrom/aix/setup /cdrom
   ```

   (Example of execution in HP-UX)

   ```
   /cdrom/IPFHPUX/SETUP /cdrom
   ```

   (Example of execution in Linux)

   ```
   /mnt/cdrom/x64lin/setup /mnt/cdrom
   ```

   The CD-ROM setup program installs the PP installer and the resident process auto-start program on the hard disk, and the PP installer starts automatically.

   Important note

   The coded contents and how to display the contents for the CD-ROM directory and file names might differ depending on the computer environment. Check with the `ls` command, and enter the displayed file name as is.

6. In the main menu of the PP installer, press the **I** key.

   The Install PP window appears.

7. Move the cursor to the program, and then press the **Space** key.

   Select the required items from the common modules of the products, and the following component software:

   - Component Container - Redirector (redirector functionality)
   - HTTP Server (Web server)
   - Performance Tracer

   <`@`> appears on the left side of the selected program. Note that you can also select and install a program.

   Important note

   The program names displayed by the products vary for the common modules of the products.

8. Make sure that <`@`> is displayed on the left side of all the programs, and then press the **I** key.

   The message **Install PP? (y: install, n: cancel)==>** appears at the bottom of the window.

9. Press the **y** or **Y** key.
   The installation process starts.
   If you press the **n** or **N** key, the installation process is cancelled and you return to the Install PP window.
10. When the message indicating the end of installation is output, press the **Q** key.
   You return to the main menu of the PP installer.
11. As and when required, press the **L** key in the main menu of the PP installer to check the installed programs.
   The Display PP List window appears. If you press the **P** key, the list of installed programs is output to `/tmp/hitachi_PPLIST`. Press the **Q** key to return to the main menu of the PP installer.
12. In the main menu of the PP installer, press the **Q** key.
   The installation of the redirector functionality finishes.

B. Tuning Parameters for Performing the Performance Tuning with Methods other than the Recommended Procedures

This section describes the tuning parameters when tuning performance by using a method other than the recommended procedure.

B.1 Tuning parameters for specifying the timeout (methods other than the recommended procedures)

This appendix describes the locations to set up the tuning parameters used for specifying the timeout.

(1) Timeout specified in the Web server for receiving requests from the client and sending data to the client

For the Web server integration, set up the tuning parameter for each Web server. As for the in-process HTTP server, set up the parameters in every J2EE server.

Table B–1: Tuning parameters for the timeout to be specified in the Web server for receiving requests from the client and sending data to the client (method other than the recommended procedures)

Web server to be used	Method of setup	Location of setup
Web server integration	Management portal (For Cosminexus HTTP Server)	`Timeout` directive of "**Additional directives**" of "**Perform the setup for each item**" in "**Web Server Setup**" window.
		`Timeout` directive of "**Setup file details**" of "**Directly set the details of setup file**" in "**Web Server Setup**" window.
	Edit file#	• For Cosminexus HTTP Server The `Timeout` directive in `httpsd.conf` • For Microsoft IIS `receive_client_timeout` key in `isapi_redirect.conf`
In-process HTTP server	Edit file	`webserver.connector.inprocess_http.receive_timeout` key in `usrconf.properties`
		`webserver.connector.inprocess_http.send_timeout` key in `usrconf.properties`

#
Set up the tuning parameters by editing `httpd.conf`, a definition file of Cosminexus HTTP Server.

(2) Timeout specified in the redirector for sending data to the Web container

The following table describes the tuning parameters for the timeout to be specified in the redirector. You can specify these tuning parameters only for the Web server integration.

Table B–2: Tuning parameters for the timeout to be specified in the redirector (methods other than the recommended procedures)

Web server type	Method of setup	Location of setup
Cosminexus HTTP Server	Edit file	`JkConnectTimeout` parameter of `mod_jk.conf`
Microsoft IIS	Edit file	`connect_timeout` key in `isapi_redirect.conf`
Cosminexus HTTP Server	Edit file	`JkSendTimeout` parameter of `mod_jk.conf`
Microsoft IIS	Edit file	`send_timeout` key in `isapi_redirect.conf`

(3) Timeout specified in the redirector for receiving data from the Web container

Set up the tuning parameters for each worker definition of the redirector. The following table describes the methods and locations to set up the tuning parameters for the timeout to be specified in the redirector:

Table B–3: Tuning parameters for the timeout to be specified in the redirector (methods other than the recommended procedures)

Method of setup	Location of setup
Edit file	`worker.`*worker-name*`.receive_timeout` key in `workers.properties`

You can specify this tuning parameter only for the Web server integration.

(4) Timeout specified in the Web container for receiving data from the redirector

Set up the tuning parameter for each J2EE server. The following table describes the tuning parameters for the timeout to be specified in the Web container:

Table B–4: Tuning parameters for the timeout to be specified in the Web container (method other than the recommended procedures)

Method of setup	Location of setup
Edit file	`webserver.connector.ajp13.receive_timeout` key in `usrconf.properties`

You can specify this tuning parameter only for the Web server integration.

(5) Timeout specified in the Web container for receiving data from the redirector

Set up the tuning parameter for each J2EE server. The following table describes the tuning parameters for the timeout to be specified in the Web container:

Table B–5: Tuning parameters for the timeout specified in the Web container (methods other than the recommended procedures)

Method of setup	Location of setup
Edit file	`webserver.connector.ajp13.send_timeout` key in `usrconf.properties`

You can specify this tuning parameter only for Web server integration.

(6) Timeout specified in the EJB client for remotely invoking the Enterprise Bean (RMI-IIOP communication) and for invoking the Naming Service by JNDI

Set up the tuning parameter for each J2EE server, EJB client application, or invocation by API.

The following table describes the tuning parameters (remote invocation by RMI-IIOP communication) for the timeout to be specified in the EJB client:

Table B–6: Tuning parameters for the timeout to be specified in the EJB client (remote invocation by RMI-IIOP communication) (methods other than the recommended procedures)

Unit	Method of setup	Location of setup
Each J2EE server	Edit file	`ejbserver.rmi.request.timeout` key in `usrconf.properties`

The setup of the tuning parameter for each EJB client application and for each API is the same as specified for the recommended procedures.

The following table describes the tuning parameters (Naming Service invocation) for the timeout to be specified in the EJB client:

Table B–7: Tuning parameters for the timeout to be specified in the EJB client (invoking the Naming Service) (method other than the recommended procedures)

Unit	Method of setup	Location of setup
Each J2EE server	Edit file	`ejbserver.jndi.request.timeout` key in `usrconf.properties`

The setup of the tuning parameters for each EJB client application is the same as specified in the recommended procedures.

(7) Timeout set up in the EJB client for invoking the Enterprise Bean from CTM

Set up the tuning parameter in J2EE server unit, EJB client application unit, or invocation API unit.

The same value as that specified in *(6) Timeout specified in the EJB client for remotely invoking the Enterprise Bean (RMI-IIOP communication) and for invoking the Naming Service by JNDI* is inherited as the setup value of this timeout.

(8) Timeout specified in the EJB container for the database transaction (when DB Connector is used)

Set up the tuning parameter for each J2EE server, Enterprise Bean, interface, method (for CMT), or each invocation by API (for BMT).

The following table describes the tuning parameters for the transaction timeout:

Table B–8: Tuning parameters for the transaction timeout (method other than the recommended procedures)

Unit	Method of setup	Location of setup
Each J2EE server	Edit file	`ejbserver.jta.TransactionManager.defaultTimeOut` key in `usrconf.properties`

The setup of the tuning parameters for the Enterprise Bean, interface, each method (for case of CMT), and API (for BMT) are the same as specified in the recommended procedures.

(9) Database timeout

The setup of the tuning parameter specified for the database timeout are the same as that specified in the recommended procedures.

B.2 Tuning parameters for optimizing the operations of the Web application (methods other than the recommended procedures)

This appendix describes the locations to set up the tuning parameters used for optimizing the operations of the Web application.

(1) Tuning parameters for separating the deployment of static contents and Web application

Specify the separation of the deployment of static contents and Web application as parameter of the file that defines the operations of the Web server. The setup locations, files and parameters differ according to the type of the Web server used.

The following table describes the methods and locations of setup for each type of Web server:

Table B–9: Tuning parameters for separating the deployment of static contents and Web application (methods other than the recommended procedures)

Web server used	Type of Web server	Method of setup	Location of setup
Web server integration (separation using the redirector module)	Cosminexus HTTP Server	Edit file	Mapping definition of `mod_jk.conf` (`JkMount` parameter)
	Microsoft IIS	Edit file	`uriworkermap.properties`
In-process HTTP server (separation using the reverse proxy module)	Cosminexus HTTP Server	Management portal	`ProxyPass` directive of "**Additional directives**" of "**Perform setup for each item**" in "**Web Server Setup**" window.
			`ProxyPass` directive of "**Setup file details**" of "**Directly set the details**

Web server used	Type of Web server	Method of setup	Location of setup
			of setup file" in "**Web Server Setup**" window.
		Edit file	`ProxyPass` directive# of `httpsd.conf`

#

For details about `httpsd.conf`, see the *uCosminexus Application Server HTTP Server User Guide*.

(2) Tuning parameters for caching static contents

The tuning parameters for caching static contents are explained in this appendix. You set up these tuning parameters for each Web container or Web application.

The following table describes the methods and locations to set up the tuning parameters specified in each Web container:

Table B–10: Tuning parameters for caching static contents (items to be set up for each Web container) (methods other than the recommended procedures)

Setup items	Method of setup	Location of setup
Select whether static contents cache is to be used	Edit file	`webserver.static_content.cache.enabled` key in `usrconf.properties`
Setup of maximum memory size for each Web application	Edit file	`webserver.static_content.cache.size` key in `usrconf.properties`
Setup of maximum file size of the static contents for cache	Edit file	`webserver.static_content.cache.filesize.threshold` key in `usrconf.properties`

The setup of the tuning parameters for each Web application is the same as that specified in the recommended procedures.

(3) Tuning parameters for distributing the requests using a redirector

Specify the tuning parameters for distributing the requests using a redirector as parameters of the file that defines the operations of the Web server. The location of setup, files, and parameters differ according to the type of the Web server used.

You can define these tuning parameters only for Web server integration. You cannot define these tuning parameters if you are using an in-process HTTP server.

The following table describes the methods and locations of setup for each Web server:

Table B–11: Tuning parameters for distributing the requests using a redirector (methods other than the recommended procedures)

Type of Web server	Method of setup	Location of setup
Cosminexus HTTP Server	Edit file	Mapping definition of `mod_jk.conf` (`JkMount` parameter)
Microsoft IIS	Edit file	`uriworkermap.properties`

B.3 Tuning parameters for a persistent connection (methods other than the recommended procedures)

This appendix describes the tuning parameters for a persistent connection.

Determine the tuning of this item when you use in-process HTTP servers in Web front-end systems.

Table B–12: Tuning parameters to be set up for a Persistent Connection (methods other than the recommended procedures)

Setup item	Method of setup	Location of setup
Upper-limit value of the number of Persistent Connections	`usrconf.properties`	`webserver.connector.inprocess_http.persistent_connection.max_connections` key
Upper-limit value of the request-processing frequency	`usrconf.properties`	`webserver.connector.inprocess_http.persistent_connection.max_requests` key
Timeout	`usrconf.properties`	`webserver.connector.inprocess_http.persistent_connection.timeout` key

C. Error Status Code

This appendix describes the error status codes returned by the Web container, redirector, and in-process HTTP server.

The location where the error occurs depends on the used Web server. Reference the error status code according to the location where the error has occurred. The following table describes the used Web servers and the corresponding locations where the error has occurred.

Table C–1: Used Web server and the corresponding location where the error occurred

Cosminexus HTTP Server to be used	Location where the error occurred		
	Web container	Redirector	HTTP In-process HTTP server
Hitachi Web Server or Microsoft IIS	Y	Y	N
In-process HTTP server	Y	N	Y

Legend:

Y: Error occurs

N: Error does not occur

C.1 Error status codes returned by the Web container

When the client accesses a non-existent resource or a servlet in which an exception occurred, the Web container returns an error status code. The following table describes the error status codes returned by the Web container, and the conditions for returning the error status codes.

Table C–2: Error status codes returned by the Web container and conditions for returning the error status codes

Error status codes	Conditions for returning the error status code
400 Bad Request	Error status code `400` is returned when any of the following conditions are met: • When a client directly sends a request to a resource that is specified as the login page used for Form authentication, and the user is successfully authenticated from the login page that is displayed as a result of the request • When the access satisfies all the following three conditions: 1. The version of HTTP is "HTTP/1.0". 2. The servlet to be accessed inherits `javax.servlet.http.HttpServlet`. 3. The HTTP method at the access is not overwritten by the servlet. • When an access is made by a request header with a Content-Length header value of greater than 2147483647 or smaller than 0 • When an access is made by a request header with a non-numeric Content-Length header value • When an access is made by a request header containing multiple Content-Length headers • When request URIs cannot be normalized
401 Unauthorized	The error status code 401 is returned when a resource that requires Basic authentication is accessed as follows: • The access uses an invalid user ID or password. • The access does not include authentication information (the Authorization header).
403 Forbidden	Error status code `403` is returned when any of the following conditions are met: • When a resource that requires the Basic or Form authentication is accessed using an unauthenticated user name

Error status codes	Conditions for returning the error status code
	• When a resource that does not permit any access is accessed without any `role-name` element for the `auth-constraint` element being specified in `web.xml`#1 • When static contents are accessed by using the `PUT` or `DELETE` method • When a resource whose `<transport-guarantee>` element in `web.xml` is set to `INTEGRAL` or `CONFIDENTIAL` is accessed via http#2
404 Not Found	Error status code 404 is returned when any of the following is accessed: • When a non-existent resource is accessed • When a servlet or a JSP file in which `javax.servlet.UnavailableException` occurs, is accessed#3
405 Method Not Allowed	Error status code 405 is returned in the case of an access that satisfies all of the following three conditions: • When the HTTP version is "HTTP/1.1" • When the servlet to be accessed inherits `javax.servlet.http.HttpServlet` • When the HTTP method during access does not get overridden by the corresponding servlet
412 Precondition Failed	Error status code 412 is returned, when the static contents that do not match the conditions specified in If-Match header or If-Unmodified-Since header, are accessed.
413 Request Entity Too Large	Error status code 413 is returned when the size of the request body exceeds the upper-limit value.
416 Requested Range Not Satisfiable	Error status code 416 is returned, when the static contents that use the value of an invalid Range header applicable to any of the following cases, are accessed: • The value of Range header does not begin with "`byte`" • A numeric character and "`-`" is not used in range definition • The specified range is not appropriate
500 Internal Server Error	Error status code `500` is returned when any of the following conditions are met: • When a servlet or a JSP file in which an exception occurs, is accessed#4 • When a JSP file whose compilation failed is accessed • When deleted static contents are accessed#5 • When I/O error occurs when accessing the static contents • When a resource protected by `<auth-constraint>` element is accessed, when the definition of `web.xml` is invalid#6
501 Not Implemented	Error status code 501 is returned, when the static contents or the servlet that inherits `javax.servlet.http.HttpServlet` is accessed by an HTTP method other than the `GET`, `HEAD`, `POST`, `PUT`, `DELETE`, `OPTIONS`, and `TRACE` method.
503 Service Unavailable	Error status code `503` is returned when any of the following conditions are met: • When there is no space in the pending queue of requests#7 • When a servlet or JSP file in which `javax.servlet.UnavailableException` occurs, is accessed#8 • When a Web container being shut down is accessed • When a Web application that is in abnormal state due to an unexpected error or exception, is accessed

#1

Applicable when the Web application is version 2.4 or later.

#2

This applies to the case when the port number of the https, used by the URL an access is forwarded to, is not set for the `webserver.connector.redirect_https.port` key in `usrconf.properties`.

#3

Applicable when the version of the Web application is 2.4 or later, and `javax.servlet.UnavailableException` indicating the permanent unavailability occurs, and the exception is not caught in the servlet and JSP file.

#4

Applicable in the following cases:

- When the version of the Web application is 2.4 or later

  When exception is not caught in servlets or JSPs
- When the version of the Web application is 2.3

  When the error page is not specified in the `<error-page>` tag of `web.xml`, or in the page directive of a JSP file, and the exception is not caught in the servlet or JSP file

#5

Applicable when the reload functionality of the Web application, re-compilation functionality of the JSP file, or the reload functionality of J2EE application is not used.

#6

Applicable when the `<role-name>` element is defined in the `<auth-constraint>` element of `web.xml`, and the `<login-config>` element is not defined. If the application is started in this state, warning message KDJE39150-W is output to the console window, and in the message log.

#7

Applicable when the settings to control the number of concurrently executing threads in the Web application, or in the URL group are specified.

#8

Applicable in the following cases:

- When the version of Web application is 2.4 or later

  When `javax.servlet.UnavailableException` indicating the temporary unavailability occurs, and the exception is not caught in the servlets or JSPs
- When the version of Web application is 2.3

  When the error page is not specified in `<error-page>` tag of `web.xml`, or in the page directive of JSP file, and the exception is not caught in the servlet or JSP file

C.2 Error status codes returned by the Redirector

When a timeout occurs during a data transaction with the Web container, and when the coding of the definition file contains an error, the redirector returns an error status code. The following tables describe the error status codes returned by the redirector and the conditions for returning the error status code, for each type of Web server.

Table C–3: Error status codes returned by the redirector and the occurrence conditions (for Cosminexus HTTP Server)

Error status codes	Conditions for returning the error status code
400 Bad Request	Error status code `400` is returned when any of the following conditions are met: • When the port number of Host header of the request is invalid • When the request method is not `POST`#1 • When the request does not have a Content-Length header (for a POST request, the body is a chunk format)#1 • When the Content-Length header value of the request exceeds the upper limit of the POST data size set in the POST request sending destination worker #1
500 Internal Server Error	Error status code `500` is returned when any of the following conditions are met: • When there is a coding error in the contents of `mod_jk.conf`#2 • When there is a failure in reading, or a coding error in the contents of `workers.properties`#1 • When the request header exceeds 16 KB#3 • When failed to establish connection with the Web container • When timeout occurs during establishment of a connection to the Web container • When an error occurs during sending data to the Web container

Error status codes	Conditions for returning the error status code
	• When timeout occurs during sending data to the Web container • When an error occurs during receiving data from the Web container • When timeout occurs during receiving data from the Web container • When timeout occurs in reading the POST data from the client • When an unsupported HTTP method#4 is specified in the request

#1

Applicable when the default worker is not specified in the distribution by the POST data size.

#2

Applicable only in Windows. The web server fails to start in UNIX.

#3

Might be applicable when a request header of a total size of 16 KB or more can be received according to the settings for limitations of request headers in Cosminexus HTTP Server.

#4

For details on whether the HTTP method is supported or not, see *Table C-5*.

Table C–4: Error status codes returned by the redirector and the occurrence conditions (for Microsoft IIS)

Error status codes	Conditions for returning the error status code
400 Bad Request	Error status code `400` is returned when any of the following conditions are met: • When the request URL contains a (percent sign (`%`) and the two characters after the percent sign (`%`) do not express hexadecimals (characters other than A-F, a-f, or 0-9) • When the port number of Host header of the request is invalid
403 Forbidden	Error status code `403` is returned when any of the following conditions are met: • When the request URL begins with "`hitachi_ccfj`"#1 • When the request URL contains "`%2F`"#1
500 Internal Server Error	Error status code `500` is returned when any of the following conditions are met: • When there is a coding error in the contents of `isapi_redirect.conf` • When there is failure in reading, or a coding error in the contents of `workers.properties` • When the request header exceeds 16 KB#2 • When failed to establish connection with the Web container • When timeout occurs during establishment of a connection to the Web container • When an error occurs during sending data to the Web container • When timeout occurs during sending data to the Web container • When an error occurs during receiving data from the Web container • When timeout occurs during receiving data from the Web container • When timeout occurs in reading the POST data from the client • When an unsupported HTTP method#3 is specified in the request

#1

Not case-sensitive.

#2

Might be applicable when a request header of a total size of 16 KB or more can be received according to the settings for limitations of request headers in Microsoft IIS.

#3

For details on whether the HTTP method is supported or not, see *Table C-5*.

Support for the request HTTP methods in the redirector

The following table lists the request HTTP methods that are supported in the redirector.

Table C–5: Support for the request HTTP methods in the redirector

HTTP method	Supported or not
`OPTIONS`	Y
`GET`	Y
`HEAD`	Y
`POST`	Y
`PUT`	Y
`DELETE`	Y
`TRACE`	Y
`CONNECT`	--#
`PROPFIND`	Y
`PROPPATCH`	Y
`MKCOL`	Y
`COPY`	Y
`MOVE`	Y
`LOCK`	Y
`UNLOCK`	Y
`ACL`	Y
`REPORT`	Y
`VERSION-CONTROL`	Y
`CHECKIN`	Y
`CHECKOUT`	Y
`UNCHECKOUT`	Y
`SEARCH`	Y
Methods available in HTTP1.1 other than above-mentioned methods	--#

Legend:

Y: Supported

--: Not supported

#

The redirector returns a status 500 error to a request that specifies an unsupported HTTP method. Furthermore, the message KDJE41001-E is output.

C.3 Error status codes returned by the in-process HTTP server

When the size of the request from the client exceeds the upper limit and if the value is invalid, the in-process HTTP server returns the error status code. The following table describes the error status codes returned by the in-process HTTP server, and the conditions for returning the error status codes.

Table C–6: Error status codes returned by the in-process HTTP server and the occurrence conditions

Error status codes	Conditions for returning the error status code
400 Bad Request	Error status code `400` is returned when the request status meets any of the following conditions: • When the request HTTP version is 1.1 and the Host header does not exist • When the port number of Host header of the request is invalid • When the size of the request header exceeds the upper limit • When the number of request headers exceeds the upper limit • When the request URI is invalid • When an attempt to decode the request URI has failed • When the request URI cannot be normalized • When the Content-Length header value of the request is greater than 2147483647 or smaller than or 0 • When the Content-Length header value of the request is a non-numeric value • When multiple Content-Length headers are specified for the request • When the HTTP version of the request line is not supported
403 Forbidden	The error status code 403 is returned when a resource whose `<transport-guarantee>` element in `web.xml` is set to `INTEGRAL` or `CONFIDENTIAL` is accessed via http.
405 Method Not Allowed	Error status code 405 is returned when access is made from an HTTP method that is not permitted
413 Request entity too large	Error status code 413 is returned when the request body size exceeds the upper limit.
414 Request-URI too large	Error status code 414 is returned when the length of the request line exceeds the upper limit.
500 Internal Server Error	Error status code 500 is returned when an attempt to read the file fails and the file is returned with the status code 200 by the redirect functionality.
501 Not Implemented	Error status code 501 is returned when the transfer-encoding header value of the request is not supported.
503 Service Unavailable	Error status code 503 is returned when an attempt is made to process the requests exceeding the upper limit of flow control.

D. Precautions related to Cosminexus HTTP Server Settings

This appendix describes the precautions related to the Cosminexus HTTP Server settings.

D.1 Precautions for restarting Cosminexus HTTP Server

If the cause of a failure that occurred while restarting Cosminexus HTTP Server exists in the Easy Setup definition file (if the Smart Composer functionality is not used, the redirector definition file (`mod_jk.conf`) or the workers file (`workers.properties`)), the message is output to one of the following:

- Start command execution window or event log of Cosminexus HTTP Server
  - When Cosminexus HTTP Server is started, stopped, or restarted from the Command Prompt, the message is output to the Start command execution window.
  - When Cosminexus HTTP Server is started, stopped, or restarted from the service, the message is output in the event log. This is applicable only in Windows.
- Error log files for Cosminexus HTTP Server
  By default, error log files are output to the following locations:
  - In Windows
    Cosminexus-HTTP-Server-installation-directory`\logs\error.log`
  - In UNIX
    `/opt/hitachi/httpsd/logs/error.log`

The following table describes each of the output messages:

Table D–1: Messages output in the Start command execution window or event log of Cosminexus HTTP Server

Message	Cause and Action
JkWorkersFile file_name invalid#	The file name specified in the `JkWorkersFile` key is invalid. Correct the value specified in the `JkWorkers`File key, and then restart the Web server.
Can't find the workers file specified#	The specified workers file cannot be found. Check whether the file specified in the `JkWorkersFile` key exists, and then restart the Web server.
Content should start with /#	The first character in the URL pattern is not forward slash (`/`). Correct the first character of the URL pattern specified in the `JkMount` key to forward slash (`/`), and then restart the Web server.
JkOptions: Illegal option '*a...a*'	The value (*a...a*) specified in the `JkOption` key is invalid. Correct the value specified in the `JkOption` key, and then restart the Web server.
a...a takes *b...b* arguments,	The number of values that can be specified in the key shown in *a...a* is *b...b*. Correct the number of values that are specified in the key shown in *a...a*, and then restart the Web server.
a...a must be On or Off	The value that can be specified in the key shown in *a...a* is either `On` or `Off`. Change the value specified in the key shown in *a...a* to `On` or `Off`, and then restart the Web server.
JkModulePriority: Invalid value specified.*a...a*	The value *a...a* specified in the `JkModulePriority` key is invalid. Specify the correct value in the `JkModulePriority` key, and then restart the Web server.

#
This message is output only in UNIX.

Table D–2: Error log files of Cosminexus HTTP Server

Message	Description and action
[Time] [emerg] Memory error	The memory is insufficient. Secure the memory that the system can use, and then restart the Web server.
[Time] [emerg] Error while opening the workers#	One of the following problems occurred: • Opening of the workers file failed. Check whether read permission is included in the access permission of the file specified in the `JkWorkersFile` key, and then restart the Web server. • The contents of the workers file are invalid. Correct the contents of the workers file, and then restart the Web server.
[Time] [emerg] Error while checking the mod_jk.conf#	The description in the redirector definition file is inappropriate. Check the message output in the redirector, and then restart the Web server.

#
This message is output only in UNIX.

D.2 Precautions related to the redirector log

- If there is no permission to write the Cosminexus HTTP Server execution account in the log output destination directory, the request is processed normally, but the log is not output.
- In Windows, by default the log output destination directory (*Cosminexus-installation-directory*`\CC\web\redirector\logs`) of the redirector does not exist. Therefore, when Cosminexus HTTP Server starts, the access permission of the directory that is one level higher (`redirector` directory) is inherited, the `logs` directory is generated, and an attempt is made to output the log. At this time, if there is no write permission for the Cosminexus HTTP Server execution account, the log is not output. In this case, generate the logs directory and set the access permission or set the access permission for the directory that is one level higher (`redirector` directory).

  Furthermore, when the log output destination directory of the redirector is changed and the specified path exists only halfway, set the access permission for the existing lowest directory or create the directory corresponding to the specified path completely and set the access permission for the lowest directory.

D.3 Precautions for upgrading Cosminexus HTTP Server

When Cosminexus HTTP Server is upgraded, you must restart Cosminexus HTTP Server to reflect the changes in the redirector for Cosminexus HTTP Server. For details on how to restart Cosminexus HTTP Server, see the *uCosminexus Application Server HTTP Server User Guide*.

E. Microsoft IIS Settings

This appendix describes how to specify the settings when integrating with a Web server using Microsoft IIS.

E.1 Microsoft IIS 10.0 settings

This appendix describes how to specify the settings for Microsoft IIS 10.0 when integrating with a Web server using Microsoft IIS.

To integrate a J2EE server with Microsoft IIS:

1. Install Microsoft IIS and role services
2. Add the ISAPI and CGI limitations
3. Add an ISAPI filter
4. Set the handler mapping
5. Add a virtual directory
6. Disable the authentication functionality of the Web server
7. Set an application pool in the **Server** node
8. Set an application pool in the **Site** node
9. Set the access permissions for the log output destination directory of the redirector
10. Start the Web site

Reference note

(INTENTIONALLY DELETED)

(1) Installing Microsoft IIS and role services

To use Microsoft IIS, you need to install Microsoft IIS and role services in accordance with your goals. The following shows the procedure for installing IIS and role services:

If Microsoft IIS is already installed and additional role services are necessary, install the role services. (The required role services are described later.)

1. Start **Server Manager**.
   (If Server Manager does not start automatically, start it from the startup window.)

2. From **Server Manager**, click **Add Roles and Features**.

3. After the wizard starts, click **Next**.

4. Select **Role-based or feature -based installation**, and then click **Next**.

5. Select the desired server, and then click **Next**.

6. From **Server Roles**, select **Web Server (IIS)**, and then click **Next**.
   If the message `Do you want to add required features to Web Server (IIS)?` pops up, select the **Include management tools (if applicable)** check box, and then click **Add Features**.

7. From **Select Features**, click **Next**.

8. From **Web Server Role (IIS)**, click **Next**, and from **Select Role Services**, select the minimum role services that you need, as follows, and then click **Next**.
   - HTTP error
   - Reference directory
   - Existing documents
   - Static contents
   - ISAPI extension
   - ISAPI filter
   - IIS management console

   Add other role services as necessary. In particular, **HTTP log** and **Trace** are useful when troubleshooting failures.

9. Click **Install**.

10. Click **Close**.
    Installation is complete.

(2) Adding the ISAPI and CGI limitations

To add the limitations of ISAPI and CGI:

1. Start **Internet Information Service (IIS) Manager**.
   You can start it by selecting **Tools** and then **Server Manager**.

2. In the **Features View** server, from the **Home** page, double-click **ISAPI and CGI Limitations**.

3. In the **ISAPI and CGI Limitations** page, under the Actions window, click **Add**.

4. In the Add ISAPI or CGI Limitations dialog box, perform the following operations:
   - In **ISAPI or CGI Path**, specify the DLL (*Cosminexus-installation-directory*`\CC\web\redirector\isapi_redirect.dll`) of the redirector.
   - In **Description**, enter `ISAPI`.
   - Check the **Allow Execution of Extension Path** checkbox.

5. Click the **OK** button.
   Close the **Add ISAPI or CGI limitations** dialog box to apply the settings.

(3) Adding an ISAPI filter

To add an ISAPI filter:

1. In the **Features View** site, from the **Home** page, double-click **ISAPI Filter**.

2. In the **ISAPI Filter** page, under the Actions window, click **Add**.

3. In the Add ISAPI Filter dialog box, perform the following operations:
   - In **Filter Name**, enter `hitachi_ccfj`.

- In **Executable File**, specify the DLL (*Cosminexus-installation-directory*`\CC\web\redirector\isapi_redirect.dll`) of the redirector.

4. Click the **OK** button.
   The Add ISAPI Filter dialog box closes, and the settings are applied.

(4) Setting the handler mapping

To set the handler mapping:

1. In the **Features View** site, from the **Home** page, double-click **Handler Mapping**.
2. In the **Handler Mapping** page, select **ISAPI-dll**, and then in the Actions window, click **Edit**.
3. In the Edit Module Map dialog box, perform the following operations:
   - In the **Requested Path**, enter `*.dll`.
   - In **Executable File**, specify the DLL (*Cosminexus-installation-directory*`\CC\web\redirector\isapi_redirect.dll`) of the redirector.

   If the handler mapping is already set, the Edit Script Map dialog box will run; however, the operation contents are the same.
4. Click the **OK** button.
5. In the message dialog box for confirming whether or not to enable the editing of the module map, click the **Yes** button to enable the ISAPI extension functionality.
6. In the **Handler Mapping** page, select **ISAPI-dll**, and then in the Actions window, click **Edit Access Permission of Functionality**.
7. In the Edit Access Permission of Functionality dialog box, check all access permissions including **Read**, **Script**, and **Execute**.
8. Click the **OK** button.
   The Edit Access Permission of Functionality dialog box closes, and the settings are applied.

(5) Adding a virtual directory

Add a virtual directory named `hitachi_ccfj`. To add the virtual directory:

1. In the Connections window, expand the **Site** node, and click the site for adding the virtual directory.
2. In the Actions window, click **Display Virtual Directory**.
   The **Virtual Directory** page appears.
3. In the **Virtual Directory** page, under the Actions window, click **Add Virtual Directory**.
4. In the Add Virtual Directory dialog box, perform the following operations:
   - In **Alias**, enter `hitachi_ccfj`.
   - In **Physical path**, specify the directory in which the DLL (*Cosminexus-installation-directory*`\CC\web\redirector\isapi_redirect.dll`) of the redirector is saved.
5. Click the **OK** button.

The Add Virtual Directory dialog box closes, and the settings are applied.

(6) Disabling the Web server authentication functionality

To use the authentication functionality of the Web container, you need to disable the authentication functionality of the Web server, as follows:

- Active Directory client certificate authentication
  To access the resources managed by the Web container, disable this functionality.
- Digest authentication
  Be sure to disable this functionality regardless of whether the authentication functionality of the Web container is used.
- Windows authentication
  To use the authentication functionality (Basic authentication) of the Web container, disable this functionality.
- APS.NET impersonation
  To access the resources managed by the Web container, disable this functionality.

The following shows how to disable Active Directory client certificate authentication, digest authentication, Windows authentication, forms authentication, and APS.NET impersonation:

1. In the Connections window, select **Home** page, and then double-click **Authentication**.
2. In the Authentication window, select **Active Directory Client Certificate Authentication**. In the Action window, click **Disable.** (Optional)
3. In the Connections window, unroll the **Site** node, and then click **hitachi_ccfj**.
4. On the **Home** page, double-click **Authentication**.
5. In the Authentication window, select **Forms Authentication**. In the Action window, click **Disable**. (Optional)
   To access the resources managed by the Web container, disable this functionality.
6. In the Authentication window, select **Digest Authentication**. In the Action window, click **Disable**. (Required)
   Be sure to disable digest authentication, regardless of whether the authentication functionality of the Web container is used.
7. In the Authentication window, select **Windows Authentication**. In the Action window, click **Disable**. (Optional)
   Be sure to disable Windows authentication when Basic authentication of the Web container is set.
8. In the Authentication window, select **APS.NET Impersonation**. In the Action window, click **Disable**. (Optional)
   To access the resources managed by the Web container, disable this functionality.
9. Click **OK** to close each window.

(7) Setting an application pool in the Server node

To set an application pool in the **Server** node:

1. In the Connections window, expand the **Server** node, and then click **Application Pool**.
2. In the **Application Pool** page, select the application pool to be used, and then in the Actions window, click **Detailed Settings**.

3. In the Detailed Settings dialog box, perform the following operation:

   In **Maximum Number of Worker Processes**#, specify the maximum number of processes for processing requests in Microsoft IIS.

   #

   In this manual, the execution processes of the Web container are also referred to as *Worker processes*; however, the *Worker processes* set here are the processes used to process requests in Microsoft IIS.

4. Click the **OK** button.

   The Detailed Settings dialog box closes, and the settings are applied.

(8) Setting an application pool in the Site node

To set an application pool in the **Site** node:

1. In the Connections window, expand the **Site** node, and then click the site in which to specify the application pool.
2. In the Actions window, click **Detailed Settings**.
3. In the Detailed Settings dialog box, enter **Application Pool**.

   In Application Pool, specify the name of the application pool set in *(7) Setting an application pool in the Server node*.
4. Click the **OK** button.

   The Detailed Settings dialog box closes, and the settings are applied.

(9) Setting access permissions for the log output destination directory of the redirector

In the log output destination directory of the redirector, you must add the permission for writing to the execution account of the application pool of Microsoft IIS. Set the access permissions for the log output destination directory of the redirector from **Explorer**.

Specify the execution account of the application pool in the Detailed Settings dialog box of the application pool, under **ID**. If the default ID is specified, add the write permission for the **IIS_IUSRS** group.

The default IDs are as follows:

- Microsoft IIS 10.0: `ApplicationPoolIdentity`

Note that during new installation, by default, the log output destination directory of the redirector (*Cosminexus-installation-directory*`\CC\web\redirector\logs`) does not exist. Therefore, either create the logs directory and set the access permission, or set the access permission for the directory that is one level higher (`redirector` directory).

Also, when the log output destination directory of the redirector is changed and the specified path exists up to only halfway, either set the access permission for the existing lowermost directory, or create all directories corresponding to the specified path and set the access permission for the lowermost directory.

(10) Starting the Web site

To start the Web site of Microsoft IIS:

1. In the Connections window, expand the **Site** node, and then click the site to be started.
2. In the Actions window, click **Start**. If the site is already started, click **Restart**.

The Web site either starts or is restarted.

(11) Notes

This appendix describes the notes on setting Microsoft IIS.

(a) Notes on replication of configuration settings in multiple environments

In Microsoft IIS, you can save the configuration settings in the `web.config` file. Also, based on the saved `web.config` file, you can replicate the configuration settings in multiple environments using `xcopy`.

However, for a configuration environment in which the redirector is used, you cannot set the redirector when replicating the settings in multiple environments using `xcopy`. Even when you specify the same configuration settings for multiple environments using `xcopy`, set the redirector manually in each environment.

(b) Notes on customizing the error page

If you have specified the settings for returning a custom error page in the error page of Microsoft IIS, the customization of the error page by the `<error-page>` tag of `web.xml` might be disabled. If you want to enable the customization of the error page by the `<error-page>` tag of `web.xml`, disable the settings of custom error page in the error page of Microsoft IIS.

(c) Points to be noted when using along with other filters

If Microsoft IIS receives requests that are to be forwarded to the Web container, the Redirector for Microsoft IIS changes the request URL information to be used in the ISAPI filter. Therefore, the request URL received by Microsoft IIS cannot be retrieved in the ISAPI filter if the ISAPI filter is executed after the Redirector for Microsoft IIS. Accordingly, you must make the settings in such a way that the filter is executed before the Redirector for Microsoft IIS, and the filter can retrieve the URL requests received by Microsoft IIS. To change the execution order, you can set the execution priority of the Redirector for Microsoft IIS by setting the value of the `filter_priority` key in the `isapi_redirect.conf` file (Operation Definition file of the Redirector for Microsoft IIS) to "Medium" or "Low". For details on the `filter_priority` key of the `isapi_redirect.conf` file (Operation Definition file of the Redirector for Microsoft IIS) see *14.2.1 isapi_redirect.conf (Redirector action definition file for Microsoft IIS)*.

F. Contract Between the JPA Provider and EJB Container

This appendix describes the contract between the JPA provider and EJB container.

F.1 Runtime-related contract

The runtime-related contract includes the responsibilities of the container and the responsibilities of the JPA provider.

(1) Responsibilities of the container

- **Contract related to the transaction scope persistence context**

  If the transaction scope persistence context is used, the container executes the following processing if the entity manager is not associated with a JTA transaction:

  - The container invokes `EntityManagerFactory.createEntityManager` in the following cases and creates a new entity manager:

    If the entity manager method that uses the transaction scope persistence context is invoked for the first time in a business method within the scope of the JTA transaction
  - After the JTA transaction is concluded (committed or rolled back), the container invokes `EntityManager.close` to close the entity manager.

  Also, if the container satisfies all the following conditions, `TransactionRequiredException` is thrown:

  - When the transaction scope persistence context is used
  - When the transaction is not active
  - When the application invokes the `persist`, `remove`, `merge`, and `refresh` methods of `EntityManager`
- **Contract related to the extended persistence context**

  If the extended persistence context is used, the container executes the following processing:

  - The container invokes `EntityManagerFactory.createEntityManager` in the following cases and creates a new entity manager:

    If the references to the entity manager using the extended persistence context are defined when the Stateful Session Bean instance is created
  - The container invokes `EntityManager.close` at the following timing and closes the entity manager:

    When the Stateful Session Bean that created the entity manager and the Stateful Session Bean that inherited the same persistence context are deleted
  - When a business method of the Stateful Session Bean that uses a container-managed transaction is invoked, if the entity manager is not associated with a JTA transaction, the container associates the entity manager with the JTA transaction and invokes `EntityManager.joinTransaction`. If another entity manager is already associated with the JTA transaction, the container throws `EJBException`.
  - When `UserTransaction.begin` is invoked in a business method of the Stateful Session Bean that uses a bean-managed transaction, the container associates the entity manager with the JTA transaction and invokes `EntityManager.joinTransaction`. For details on the bean-managed transactions, see *2.7.2 BMT* in the *uCosminexus Application Server EJB Container Functionality Guide*.
- **Contract related to the container-managed entity manager**

  If the application invokes `EntityManager.close` when the container-managed entity manager is being used, the container throws `IllegalStateException`.

If the property is specified in `@PersistenceContext` or in the `<persistence-context-ref>` tag of the DD, the container uses the `EntityManagerFactory.createEntityManager(Map map)` method, creates the entity manager, includes the specified property in the `map` argument, and passes the property to the JPA provider.

- **Automatic connection closing functionality**

  In Application Server, the connection obtained by the JPA provider with the extension of the Session Bean and Web components sometimes closes automatically by the automatic closing functionality of the container. The automatic connection closing functionality is used to prevent a connection leak.

  The connection is subject to being closed automatically if the following conditions are satisfied:

  - A connection obtained by a Stateless Session Bean is closed automatically when returned from a business method.
  - A connection obtained by a Stateful Session Bean is closed automatically when the Stateful Session Bean is destroyed.
  - A connection obtained by a Web component is closed automatically when returned from a service method.

  However, even if the above conditions are applicable, if the connection is participating in a JTA transaction, automatic close is reserved until the JTA transaction is committed.

(2) Responsibilities of the JPA provider

Whether the entity manager to be used with an application is defined to use the transaction scope persistence context or the extended persistence context, is not transmitted to the JPA provider. The responsibilities of the JPA provider include creating the entity manager when requested by the container and registering Synchronization in the transaction in order to receive the notification about transaction conclusion from the transaction.

- When the container invokes `EntityManagerFactory.createEntityManager`, the JPA provider must create a new entity manager and return the created entity manager to the container. If a JTA transaction is active, the JPA provider must register Synchronization in the JTA transaction.
- When the container invokes `EntityManager.joinTransaction`, the JPA provider must register Synchronization in the JTA transaction. However, if `joinTransaction` was invoked previously and Synchronization is already registered in the JTA transaction, nothing need be done.
- When a JTA transaction is committed, the JPA provider must flush all the changed entity statuses in the database.
- When a JTA transaction is rolled back, the JPA provider must detach all the `managed` entities.
- When the JPA provider throws an exception that causes transaction rollback, the JPA provider must mark the transaction for rollback.
- When the container invokes `EntityManager.close`, the JPA provider must release all the allocated resources after all the unresolved transactions related to that entity manager are concluded. If the entity manager is already closed, the JPA provider must throw `IllegalStateException`.
- When the container invokes `EntityManager.clear`, the JPA provider must detach all the `managed` entities.

(3) javax.transaction.TransactionSynchronizationRegistry interface

The JPA provider can use the `TransactionSynchronizationRegistry` interface to register Synchronization in a transaction and to mark a transaction for rollback. The `TransactionSynchronizationRegistry` instance can be looked up with the name `java:comp/TransactionSynchronizationRegistry` using the JNDI.

The interface definition is as follows:

```
package javax.transaction;

/**
 * This interface is used from system level components
 * of Application Server such as the
 * JPA provider and resource adapters.
 * Using this interface, you can
 * register synchronization invoked in a particular order,
 * register the resource object in the current transaction,
 * obtain the current transaction context,
 * obtain the current transaction status,
 * and mark the current transaction for rollback.
 *
 * This interface is implemented by Application Server
 * as a stateless service object.
 * The same object can be used from multiple components
 * in a multi-thread safe manner.
 *
 * With default Application Server, the instance implementing this
 * interface can be looked up with the default name using the JNDI.
 * The default name is
 * java:comp/TransactionSynchronizationRegistry.
 */
public interface TransactionSynchronizationRegistry {

 /**
 * When this method is invoked, a unique object expressing the
 * transaction associated with the current thread is returned.
 * The hashCode and equals method of this object is overridden
 * and can be used as the hashmap key.
 * If the transaction does not exist, null is returned.
 *
 * All the objects returned by invoking this method in the same
 * transaction context of same Application Server have the same
 * hashCode and the comparison results in the equal method are
 * true.
 *
 * The toString method returns the transaction context information
 * as a string in an easy-to-read format.
 * However, the string format returned by toString is not defined.
 * Also, the compatibility of the toString results between versions
 * is not guaranteed.
 *
 * There is no guarantee that the obtained object can be serialized
 * and the operations when the object is sent outside JavaVM are
 * not defined.
 *
 * @return Object that uniquely expresses the transaction
 * associated with the thread when this method is invoked.
 */
 Object getTransactionKey();

 /**
 * The object is added or replaced in the resource map
 * of the transaction associated with the thread used when this
 * method is invoked.
 * The map key must be a class defined on the method invocation
 * side so that conflict does not occur.
 * The class used as the key must have the appropriate hashCode
 * and equals method as the map key.
 * The map key and value is not evaluated and used by this class.
 * The general contract of this method is the same as the put
 * method of Map and the key must be other than null, but the
 * value can be set as null.
 * If a value associated with the key already exists, the value
 * is replaced.
 *
 * @param key Entry key of map
 * @param value Entry value of map
 * @exception IllegalStateException When an active transaction
 * does not exist
 * @exception NullPointerException When the argument key is null
 */
 void putResource(Object key, Object value);

 /**
 * The object is extracted from the resource map of the transaction
 * associated with the thread used when this method is invoked.
 * The key must be the same as the object specified in the
 * putResource method beforehand, in the same transaction.
 * If the specified key does not exist in the current resource
 * map, null is returned.
 * The general contract of this method is the same as the put
 * method of Map and the key must be other than null, but the
 * value can be set as null.
 * If the key is not stored in the map or if a null value is stored
 * for the key, the return value is null.
 * @param key Entry key of map
 * @return Value associated with the key
 * @exception IllegalStateException When an active transaction
 * does not exist
 * @exception NullPointerException When the argument key is null
 */
 Object getResource(Object key);

 /**
 * Registers the synchronization instances invoked in a particular
 * order.
 * beforeCompletion of Synchronization registered with this
 * method is invoked after
 * SessionSynchronization.beforeCompletion, and
 * Synchronization.beforeCompletion, which is directly
 * registered in a transaction, are invoked, and before the 2-phase
 * commit processing starts.
 * Similarly, afterCompletion of Synchronization registered with
 * this method is invoked after the completion of 2-phase commit
 * processing and before SessionSynchronization.afterCompletion
 * and Synchronization.afterCompletion, which is directly
 * registered in the transaction, are invoked.
 *
 * beforeCompletion is invoked by the transaction context
 * associated with the thread when this method is invoked.
 * With beforeCompletion, access to resources such as connector
 * is permitted, but access is not permitted to user components
 * such as timer service and bean methods.
```

```
 * This is because the data managed on the invocation side and
 * the data that is already flushed by other Synchronization
 * registered with registerInterposedSynchronization might be
 * changed.
 * The general context becomes the context of the component
 * that invokes registerInterposedSynchronization.
 *
 * The context when afterCompletion is invoked is not defined.
 * Note that access to user components is not allowed.
 * Also, the resource can be closed, but
 * transactional operations cannot be performed for the resource.
 *
 * If this method is invoked when the transaction is not active,
 * IllegalStateException is thrown.
 *
 * If this method is invoked after the 2-phase commit processing
 * starts, IllegalStateException is thrown.
 *
 * @param sync Instance of Synchronization to be registered
 * @exception IllegalStateException When an active transaction
 * does not exist
 */
 void registerInterposedSynchronization(Synchronization sync);

 /**
 * When this method is invoked, the status of the transaction
 * associated with the thread is returned.
 * The return value of this method is the same as the result of  * Transacti
onManager.getStatus().
 *
 * @return Status of the transaction associated with
 * the thread when this method is invoked.
 */
 int getTransactionStatus();

 /**
 * When this method is invoked, the transaction associated with
 * the thread is marked for rollback.
 *
 * @exception IllegalStateException When an active transaction
 * does not exist
 */
 void setRollbackOnly();

 /**
 * When this method is invoked, returns information about whether
 * the transaction associated with the thread is marked for
 * rollback.
 *
 * @return true if the transaction is marked for rollback
 * @exception IllegalStateException When an active transaction
 * does not exist
 */
 boolean getRollbackOnly();
}
```

F.2 Deployment-related contract

The deployment-related contract includes the responsibilities of the container and the responsibilities of the JPA provider.

(1) Responsibilities of the container

During deployment, the container searches `persistence.xml` packaged at a decided location within the application. If `persistence.xml` exists in the application, the container processes the definition of the persistence unit defined in `persistence.xml`. For details on the locations searched by the container, see *8.8 Definitions in persistence.xml*.

The container verifies `persistence.xml` file using `persistence_1_0.xsd`. If the verification results in an error, the container reports to the user. If the provider and data source information is not specified in `persistence.xml`, the default value is used. For details on the default values used, see *8.8 Definitions in persistence.xml*. When creating the entity manager factory of the persistence unit, the container passes the properties to the JPA provider.

The container creates the implementation class instance of `javax.persistence.spi.PersistenceProvider` defined for each persistence unit in `persistence.xml`, invokes the `createContainerEntityManagerFactory` method, and obtains `EntityManagerFactory` for creating the container-managed entity manager. The Meta data of the persistence unit is passed as the `PersistenceUnitInfo` object to the JPA provider using the argument of the `createContainerEntityManagerFactory` method. The container creates only one `EntityManagerFactory` for one persistence unit definition and creates multiple `EntityManagers` from that `EntityManagerFactory`.

When the persistence unit is re-deployed, the container invokes the `close` method of `EntityManagerFactory` that is already obtained and then invokes `createContainerEntityManagerFactory` along with the new `PersistenceUnitInfo`.

(2) Responsibilities of the JPA provider

The JPA provider must implement `PersistenceProvider SPI`, and when the `createContainerEntityManagerFactory` method of `PersistenceProvider` is invoked, the JPA provider must use the Meta data (`PersistenceUnitInfo`) of the persistence unit passed in the argument to create `EntityManagerFactory`, and return the created `EntityManagerFactory` to the container.

The JPA provider processes the Meta data annotation of the managed class (such as an entity class) included in the persistence unit. Also, when the O/R mapping file is used in the persistence unit, the JPA provider must interpret the file. At this time, the JPA provider verifies the O/R mapping file by using `orm_1_0.xsd` and must notify the user if an error occurs.

(3) javax.persistence.spi.PersistenceProvider interface

The JPA provider must implement the `javax.persistence.spi.PersistenceProvider` interface. This interface is invoked by the container and is not invoked from the application. The `PersistenceProvider` implementation class must be public and must have a constructor without argument.

The `javax.persistence.spi.PersistenceProvider` interface is as follows:

```
package javax.persistence.spi;

/**
```

```
* Interface implemented by the JPA provider.
* This interface is used for creating EntityManagerFactory.
* In the Java EE environment, the interface is invoked by the container
* and in the JavaSE environment, the interface is invoked by the
* persistence class.
*/
public interface PersistenceProvider {

 /**
 * Invoked when the persistence class creates EntityManagerFactory.
 *
 * @param emName Name of the persistence unit
 * @param map property Map used by the JPA provider.
 * The property specified here is used to overwrite the property
 * corresponding to the persistence.xml file or to specify
 * the property that is not specified in the persistence.xml file.
 * (If the property need not be specified, null is passed)
 * @return EntityManagerFactory of persistence unit
 * If the JPA provider is incorrect, null is returned.
 */
 public EntityManagerFactory createEntityManagerFactory(String
 emName, Map map);

 /**
 * Invoked when the container creates EntityManagerFactory.
 *
 * @param info Meta data to be used by the JPA provider
 * @param map Integration level property to be used by the JPA provider
 * (if not specified, null is passed)
 * @return EntityManagerFactory of the persistence unit specified
 * in Meta data
 */
 public EntityManagerFactory createContainerEntityManagerFactory(
 PersistenceUnitInfo info, Map map);
}
```

(4) javax.persistence.spi.PersistenceUnitInfo interface

The `javax.persistence.spi.PersistenceUnitInfo` interface definition is as follows:

```
import javax.sql.DataSource;
/**
* This interface is implemented by the container and is passed to
* the JPA provider when EntityManagerFactory is created.
*/
public interface PersistenceUnitInfo {

 /**
 * @return Persistence unit name defined in persistence.xml
 */
 public String getPersistenceUnitName();

 /**
 * @return Fully qualified class name of the JPA provider
 * implementation class defined in the <provider> element of
 * persistence.xml
 */
```

```
public String getPersistenceProviderClassName();

/**
* @return Returns the transaction type of EntityManager
* created by EntityManagerFactory
* Type specified in the transaction-type attribute of
* persistence.xml.
*/
public PersistenceUnitTransactionType getTransactionType();

/**
* @return Returns a data source with JTA enabled
* for use by the JPA provider.
* Data source specified in the <jta-data-source> element of
* persistence.xml or the data source determined by the container
* during deployment.
*/
public DataSource getJtaDataSource();

/**
* @return Returns a data source with JTA disabled for the JPA provider
* to access the data outside the JTA transaction.
* Data source specified in <non-jta-data-source> element of
* persistence.xml or the data source determined by the container
* during deployment.
*/
public DataSource getNonJtaDataSource();

/**
* @return List of mapping file names that must be loaded for the
* JPA provider to determine the entity class mapping.
* The mapping file must have the standard XML mapping format.
* The mapping file must have a unique name and must be
* loadable as a resource from the application class path.
* The mapping file names are specified in the
* <mapping-file> tag of persistence.xml.
*/
public List<String> getMappingFileNames();

/**
* Returns the URL list of JAR files or directory deploying the
* JAR files, required for searching the managed class of the
* persistence unit by the JPA provider.
* Each URL is specified in the <jar-file> tag of the persistence.xml
* file.
* The URL is in a file URL format indicating JAR files or directory
* deploying the JAR files or in another URL format that can obtain
* InputStream in the JAR format.
*
* @return List of URL objects indicating the JAR files or directories
*/
public List<URL> getJarFileUrls();

/**
* Returns the URL of the JAR file or directory forming the persistence
* unit root
* (If the persistence unit root is WEB-INF/classes directory,
* returns the URL of WEB-INF/classes directory)
```

```
 * The URL is in a file URL format indicating JAR files or directory
 * deploying the JAR files or in another URL format that can obtain
 * InputStream in the JAR format.
 *
 * @return URL object indicating JAR files or directories
 */
 public URL getPersistenceUnitRootUrl();

 /**
 * @return List of class names that the JPA provider must handle
 * as managed classes.
 * Each class name is specified in the <class> tag of the
 * persistence.xml file
 */
 public List<String> getManagedClassNames();

 /**
 * @return Returns whether to handle the class that is deployed
 * in the persistence unit root and is not explicitly specified
 * as managed class, as a managed class.
 * This value is specified in the <exclude-unlisted-classes> tag
 * of the persistence.xml file.
 */
 public boolean excludeUnlistedClasses();

 /**
 * @return Properties object.
 * Each property is specified in the <property> tag of the
 * persistence.xml file.
 */
 public Properties getProperties();

 /**
 * @return ClassLoader that can be used by the JPA provider
 * to load classes and resources and to open URLs.
 */
 public ClassLoader getClassLoader();

 /**
 * Class loader returned by the PersistenceUnitInfo.getClassLoader
 * method and registers the JPA provider transformer that is invoked
 * every time a new class is defined or a class is re-defined.
 * This transformer is returned by the  * PersistenceUnitInfo.getNewTempClas
sLoader method
 * and does not affect the classes loaded by the class loader.
 * Even if some persistence units are defined in the class loading
 * scope, the class is only converted once in the same class loading
 * scope.
 *
 * @param transformer JPA provider transformer invoked by the
 * container to define (re-define) a class.
 */
 public void addTransformer(ClassTransformer transformer);

 /**
 * Returns a new instance of the class loader
 * that can be used temporarily by the JPA provider to load classes
 * and resources and to open the URLs.
 * The scope and class path of this class loader
 * is exactly similar to the class loader returned
 * by PersistenceUnitInfo.getClassLoader.
 * The classes loaded with this class loader cannot
 * be referenced from the application components.
 * The JPA provider can use this class loader only in
 * the extended invocation of createContainerEntityManagerFactory.
 *
 * @return Temporary class loader having the same scope
 * or class path as the current class loader.
 */
 public ClassLoader getNewTempClassLoader();

}
```

Reference note

With Application Server, if the JTA data source and non-JTA data source are not defined in the persistence unit, `getJtaDataSource()` or `getNonJtaDataSource()` return `null`. The preceding text "if the JTA data source and non-JTA data source are not defined in the persistence unit" indicates the following state:

- When `<jta-data-source>` and `<non-jta-data-source>` are omitted in `persistence.xml` and the default value is not defined in the system properties `ejbserver.jpa.defaultJtaDsName` and `ejbserver.jpa.defaultNonJtaDsName`
- When the system properties `ejbserver.jpa.overrideJtaDsName` and `ejbserver.jpa.overrideNonJtaDsName` are also not defined

G. BNF for JPQL

This appendix describes BNF for JPQL provided in the JPA 1.0 specifications.

Table G–1: Rules for the BNF expressions

Expression	Contents
{ }	Indicates a group.
[ ]	Indicates the option syntax.
*	Indicates 0 or more.
A|B	Indicates A or B.

Note that highlighted **bold** character strings represent keywords.

BNF is as follows:

```
QL_statement ::= select_statement | update_statement | delete_statement
select_statement ::= select_clause from_clause [where_clause] [groupby_claus
e]
[having_clause] [orderby_clause]
update_statement ::= update_clause [where_clause]
delete_statement ::= delete_clause [where_clause]
from_clause ::=
FROM identification_variable_declaration
{, {identification_variable_declaration | collection_member_declaration}}*
identification_variable_declaration ::= range_variable_declaration { join |
fetch_join }*
range_variable_declaration ::= abstract_schema_name [AS] identification_vari
able
join ::= join_spec join_association_path_expression [AS] identification_vari
able
fetch_join ::= join_spec FETCH join_association_path_expression
association_path_expression ::=
collection_valued_path_expression | single_valued_association_path_expressi
on
join_spec::= [ LEFT [OUTER] | INNER ] JOIN
join_association_path_expression ::= join_collection_valued_path_expressio
n |
join_single_valued_association_path_expression
join_collection_valued_path_expression::=
identification_variable.collection_valued_association_field
join_single_valued_association_path_expression::=
identification_variable.single_valued_association_field
collection_member_declaration ::=
IN (collection_valued_path_expression) [AS] identification_variable
single_valued_path_expression ::=
state_field_path_expression | single_valued_association_path_expression
state_field_path_expression ::=
{identification_variable | single_valued_association_path_expression}.state_
field
single_valued_association_path_expression ::=
identification_variable.{single_valued_association_field.}* single_valued_as
sociation_field
collection_valued_path_expression ::=
identification_variable.{single_valued_association_field.}*collection_valued
_association_field
state_field ::= {embedded_class_state_field.}*simple_state_field
update_clause ::= UPDATE abstract_schema_name [[AS] identification_variable]
SET update_item {, update_item}*
update_item ::= [identification_variable.]{state_field | single_valued_assoc
iation_field} =
new_value
new_value ::=
simple_arithmetic_expression |
string_primary |
datetime_primary |
boolean_primary |
enum_primary
simple_entity_expression |
NULL
delete_clause ::= DELETE FROM abstract_schema_name [[AS] identification_vari
able]
select_clause ::= SELECT [DISTINCT] select_expression {, select_expression}*
select_expression ::=
single_valued_path_expression |
aggregate_expression |
identification_variable |
OBJECT(identification_variable) |
constructor_expression
constructor_expression ::=
NEW constructor_name ( constructor_item {, constructor_item}* )
constructor_item ::= single_valued_path_expression | aggregate_expression
aggregate_expression ::=
{ AVG | MAX | MIN | SUM } ([DISTINCT] state_field_path_expression) |
COUNT ([DISTINCT] identification_variable | state_field_path_expression |
single_valued_association_path_expression)
where_clause ::= WHERE conditional_expression
groupby_clause ::= GROUP BY groupby_item {, groupby_item}*
groupby_item ::= single_valued_path_expression | identification_variable
having_clause ::= HAVING conditional_expression
orderby_clause ::= ORDER BY orderby_item {, orderby_item}*
orderby_item ::= state_field_path_expression [ ASC | DESC ]
subquery ::= simple_select_clause subquery_from_clause [where_clause]
[groupby_clause] [having_clause]
subquery_from_clause ::=
FROM subselect_identification_variable_declaration
{, subselect_identification_variable_declaration}*
subselect_identification_variable_declaration ::=
identification_variable_declaration |
association_path_expression [AS] identification_variable |
collection_member_declaration
simple_select_clause ::= SELECT [DISTINCT] simple_select_expression
simple_select_expression::=
single_valued_path_expression |
aggregate_expression |
identification_variable
conditional_expression ::= conditional_term | conditional_expression OR cond
itional_term
conditional_term ::= conditional_factor | conditional_term AND conditional_f
actor
conditional_factor ::= [ NOT ] conditional_primary
conditional_primary ::= simple_cond_expression | (conditional_expression)
```

```
simple_cond_expression ::=
comparison_expression |
between_expression |
like_expression |
in_expression |
null_comparison_expression |
empty_collection_comparison_expression |
collection_member_expression |
exists_expression
between_expression ::=
arithmetic_expression [NOT] BETWEEN
arithmetic_expression AND arithmetic_expression |
string_expression [NOT] BETWEEN string_expression AND string_expression |
datetime_expression [NOT] BETWEEN
datetime_expression AND datetime_expression
in_expression ::=
state_field_path_expression [NOT] IN ( in_item {, in_item}* | subquery)
in_item ::= literal | input_parameter
like_expression ::=
string_expression [NOT] LIKE pattern_value [ESCAPE escape_character]
null_comparison_expression ::=
{single_valued_path_expression | input_parameter} IS [NOT] NULL
empty_collection_comparison_expression ::=
collection_valued_path_expression IS [NOT] EMPTY
collection_member_expression ::= entity_expression
[NOT] MEMBER [OF] collection_valued_path_expression
exists_expression::= [NOT] EXISTS (subquery)
all_or_any_expression ::= { ALL | ANY | SOME} (subquery)
comparison_expression ::=
string_expression comparison_operator {string_expression | all_or_any_expres
sion} |
boolean_expression { =|<>} {boolean_expression | all_or_any_expression} |
enum_expression { =|<>} {enum_expression | all_or_any_expression} |
datetime_expression comparison_operator
{datetime_expression | all_or_any_expression} |
entity_expression { = | <> } {entity_expression | all_or_any_expression} |
arithmetic_expression comparison_operator
{arithmetic_expression | all_or_any_expression}
comparison_operator ::= = | > | >= | < | <= | <>
arithmetic_expression ::= simple_arithmetic_expression | (subquery)
simple_arithmetic_expression ::=
arithmetic_term | simple_arithmetic_expression { + | - } arithmetic_term
arithmetic_term ::= arithmetic_factor | arithmetic_term { * | / } arithmetic
_factor
arithmetic_factor ::= [{ + | - }] arithmetic_primary
arithmetic_primary ::=
state_field_path_expression |
numeric_literal |
(simple_arithmetic_expression) |
input_parameter |
functions_returning_numerics |
aggregate_expression
string_expression ::= string_primary | (subquery)
string_primary ::=
state_field_path_expression |
string_literal |
input_parameter |
functions_returning_strings |
aggregate_expression
datetime_expression ::= datetime_primary | (subquery)
datetime_primary ::=
state_field_path_expression |
input_parameter |
functions_returning_datetime |
aggregate_expression
boolean_expression ::= boolean_primary | (subquery)
boolean_primary ::=
state_field_path_expression |
boolean_literal |
input_parameter |
enum_expression ::= enum_primary | (subquery)
enum_primary ::=
state_field_path_expression |
enum_literal |
input_parameter |
entity_expression ::=
single_valued_association_path_expression | simple_entity_expression
simple_entity_expression ::=
identification_variable |
input_parameter
functions_returning_numerics::=
LENGTH(string_primary) |
LOCATE(string_primary, string_primary[, simple_arithmetic_expression]) |
ABS(simple_arithmetic_expression) |
SQRT(simple_arithmetic_expression) |
MOD(simple_arithmetic_expression, simple_arithmetic_expression) |
SIZE(collection_valued_path_expression)
functions_returning_datetime ::=
CURRENT_DATE|
CURRENT_TIME |
CURRENT_TIMESTAMP
functions_returning_strings ::=
CONCAT(string_primary, string_primary) |
SUBSTRING(string_primary,
simple_arithmetic_expression, simple_arithmetic_expression)|
TRIM([[trim_specification] [trim_character] FROM] string_primary) |
LOWER(string_primary) |
UPPER(string_primary)
trim_specification ::= LEADING | TRAILING | BOTH
```

H. Main Functionality Changes in Each Version

The following describes the main functionality changes in versions of Application Server earlier than version 11-00 and the purpose of each change. For details on the main functionality changes in version 11-00, see *1.3 Major functionality changes in Application Server 11-00*.

The following provides detailed descriptions:

- The following describes the main functionality that was changed in each version of Application Server and provides an overview on each change. For details on the functionality, see the descriptions in the *Reference manual* and *Reference* columns in the following table. In the *Reference manual* and *Reference* columns, you can find the locations where the main functionality is described in the manual for version 11-00.

Note that *uCosminexus Application Server* is omitted from the manual names in the *Reference manual* column.

H.1 Main functionality changes in 09-87

(1) Supporting the standard and existing functionality

The following table shows the item that was changed to support the standard and existing functionality.

Table H–1: Changes made for supporting the standard and existing functionality

Item	Overview of changes	Reference manual	Reference
Support for Java SE 11	Functionality of Java SE 11 can now be used.	*Maintenance and Migration Guide*	*Chapter 9*

H.2 Main functionality changes in 09-80

(1) Supporting the standard and existing functionality

The following table shows the items that were changed to support the standard and existing functionality.

Table H–2: Changes made for supporting the standard and existing functionality

Item	Overview of changes	Reference manual	Reference
Use of lambda expressions in the JAX-RS functionality	Lambda expressions can now be used in the classes included in the packages and subpackages specified for the servlet initialization parameters in web.xml.	*Web Service Development Guide*	*11.2*
Support for Java SE 9	The functionality of Java SE 9 can now be used.	*Maintenance and Migration Guide*	*Chapter 9*

(2) Maintaining and improving reliability

The following table shows the items that were changed to maintain and improve reliability.

Table H–3: Changes made for maintaining and improving reliability

Item	Overview of changes	Reference manual	Reference
Support for Apache2.4 Web servers	Apache2.4 is now supported as the base version of Web servers.	*HTTP Server User Guide*	*Chapter 6*, *Appendix G*
Use of elliptic-curve cryptography in SSL communication	SSL communication using elliptic-curve cryptography can now be used.	*HTTP Server User Guide*	*Chapter 5*, *Appendix G*
Change of the SSL library	The SSL library providing the SSL functionality is now changed to OpenSSL.	*HTTP Server User Guide*	*Chapter 5*, *Appendix G*

H.3 Main functionality changes in 09-70

(1) Supporting the standard and existing functionality

The following table shows the items that were changed to support the standard and existing functionality.

Table H–4: Changes made for supporting the standard and existing functionality

Item	Overview of changes	Reference manual	Reference
Addition of the JSP compiled version to the management portal	Compilation methods compliant with JDK 1.7 specifications and those compliant with JDK 7 specifications are now supported as compilation methods for servlets generated from JSP files on the J2EE server.	*Management Portal User Guide*	*10.8.4*
		Definition Reference Guide	*4.11.2*
Support for metaspace in JDK8	The option to be used to start JavaVM was changed from the option for the `Permanent` area to the option for the `Metaspace` area.	*System Setup and Operation Guide*	*Appendix A.2*
		Management Portal User Guide	*10.8.7*
		Definition Reference Guide	*5.2.1, 5.2.2, 8.2.3*
Support of SHA-2 by user authentication for integrated user management	SHA-224, SHA-256, SHA-384, and SHA-512 were added as hash algorithms for user authentication for integrated user management.	*Security Management Guide*	*5.3.1, 5.3.9, 5.10.7, 11.4.3, 12.4.3, 12.5.3, 13.2, 14.2.2*
Addition of automatic startup, automatic restart, and automatic termination to Red Hat Enterprise Linux Server 7	Methods of automatic startup, automatic restart, and automatic termination of Management Server and Administration Agent were added to Red Hat Enterprise Linux Server 7.	*Operation, Monitoring, and Linkage Guide*	*2.6.3, 2.6.4, 2.6.5*
		Command Reference Guide	*7.2*

(2) Maintaining and improving operability

The following table shows the item that was changed to maintain and improve operability.

Table H–5: Changes made for maintaining and improving operability

Item	Overview of changes	Reference manual	Reference
Support for version upgrade to V9.7	The procedure for changing the option to use to start JavaVM at version upgrade from the option for the `Permanent` area to the option for the `Metaspace` area was added.	*Maintenance and Migration Guide*	*10.3.1, 10.3.2, 10.3.4*

(3) Changes for other purposes

The following table shows the item that was changed for other purposes.

Table H–6: Changes made for other purposes

Item	Overview of changes	Reference manual	Reference
Targets to collect in snapshot logs	JavaVM event logs and Management Server thread dumps were added as the targets for collection in snapshot logs.	*Maintenance and Migration Guide*	*Appendix A.2*

H.4 Main functionality changes in 09-60

(1) Supporting the standard and existing functionality

The following table shows the items that were changed to support the standard and existing functionality.

Table H–7: Changes made for supporting the standard and existing functionality

Item	Overview of changes	Reference manual	Reference
Support for G1GC	G1GC can now be selected.	*System Design Guide*	*7.15*
		Definition Reference Guide	*14.5*
Support for the object-pointer compression function	The object-pointer compression function can now be used.	*Maintenance and Migration Guide*	*9.16*

(2) Maintaining and improving reliability

The following table shows the items that were changed to maintain and improve reliability.

Table H–8: Changes made for maintaining and improving reliability

Item	Overview of changes	Reference manual	Reference
Addition of the finalize-retention resolution function	Retention of finalization processing can now be resolved, and delays in the release of OS resources can now be suppressed.	*Maintenance and Migration Guide*	*9.17*

(3) Other purposes

The following table shows the item that was changed for other purposes.

Table H–9: Changes made for other purposes

Item	Overview of changes	Reference manual	Reference
Addition of the log file asynchronous output functionality	Log files can now be output asynchronously.	*Definition Reference Guide*	*14.2*

H.5 Main functionality changes in 09-50

(1) Improving development productivity

The following table shows the items that were changed to improve development productivity.

Table H–10: Changes made for improving development productivity

Item	Overview of changes	Reference manual	Reference
Simplification of Eclipse setup	You can now set an Eclipse environment by using GUI.	*Application Development Guide*	*1.1.5, 2.4*
Support for debugging by using user-extended performance analysis trace	Setup files for user-extended performance analysis trace can now be created in a development environment.	*Application Development Guide*	*1.1.3, 6.4*

(2) Simplifying implementation and setup

The following table shows the item that was changed to simplify implementation and setup.

Table H–11: Changes made for simplifying implementation and setup

Item	Overview of changes	Reference manual	Referenc e
Addition of system configurations in a virtual environment	The types of tiers (`http-tier`, `j2ee-tier`, and `ctm-tier`) usable in a virtual environment increased. Therefore, the following system configurations are now available: • Configuration in which the Web server and J2EE server are placed on different hosts • Configuration in which the front end (servlets, JSP) and back end (EJB) are placed separately • Configuration in which CTM is used	*Virtual System Setup and Operation Guide*	*1.1.2*

(3) Supporting the standard and existing functionality

The following table shows the items that were changed to support the standard and existing functionality.

Table H–12: Changes made for supporting the standard and existing functionality

Item	Overview of changes	Reference manual	Referenc e
Support for the JDBC 4.0 specifications	DB Connector now supports HiRDB Type4 JDBC Driver of the JDBC 4.0 specifications and JDBC driver of the SQL Server.	*Common Container Functionality Guide*	*3.6.3*
Modification of naming conventions in the Portable Global JNDI names	The characters that can be used in the Portable Global JNDI names were added.	*Common Container Functionality Guide*	*2.4.3*
Support for the Servlet 3.0 specifications	Changes to the HTTP Cookie name and URL path parameter name in Servlet 3.0 can now be used in Servlet 2.5 and earlier versions.	*Web Container Functionality Guide*	*2.7*
Expanded use of applications that can be linked with Bean Validation	Validation on CDI and UAP can now be performed by using Bean Validation.	*Common Container Functionality Guide*	*Chapter 9*

Item	Overview of changes	Reference manual	Referenc e
Support for JavaMail	The email sending and receiving functionality using the API that complies with JavaMail 1.4 can now be used.	*Common Container Functionality Guide*	*Chapter 7*
Expanded use of the OS in which the `javacore` command can be used	You can now use the `javacore` command to acquire Windows thread dumps.	*Command Reference Guide*	*javacore (Acquiring the thread dump/in Windows)*

(4) Maintaining and improving reliability

The following table shows the items that were changed to maintain and improve reliability.

Table H–13: Changes made for maintaining and improving reliability

Item	Overview of changes	Reference manual	Referenc e
Avoiding depletion of the code cache area	You can now avoid the depletion of the area by checking the size of the code cache area used by the system and changing the threshold value before the area is depleted.	*System Design Guide*	*7.2.6*
		Maintenance and Migration Guide	*5.7.2, 5.7.3*
		Definition Reference Guide	*14.1, 14.2, 14.4*
Support for efficient use of the Explicit Memory Management functionality	The functionality was added to control the objects to be moved to an Explicit heap, as the functionality to reduce the automatic release processing time and to efficiently apply the Explicit Memory Management functionality. • The functionality that controls object movement to an Explicit memory block • The functionality for specifying classes in which use of the Explicit Memory Management functionality is excluded • The functionality that outputs object release rate information to Explicit heap information	*System Design Guide*	*7.14.6*
		Expansion Guide	*7.2.2, 7.6.5, 7.10, 7.13.1, 7.13.3*
		Maintenance and Migration Guide	*5.5*
Expanded output range of statistical information for each class	A reference relation based on a static field can now be output to the extended thread dump that contains statistical information for each class.	*Maintenance and Migration Guide*	*9.6*

(5) Maintaining and improving operability

The following table shows the items that were changed to maintain and improve operability.

Table H–14: Changes made for maintaining and improving operability

Item	Overview of changes	Reference manual	Referenc e
Support for the EADs session failover functionality	The EADs session failover functionality that achieves the session failover functionality by linking with EADs is now supported.	*Expansion Guide*	*Chapter 5*
Operation based on WAR	A WAR application that consists of WAR files only can now be deployed on the J2EE server.	*Web Container Functionality Guide*	*2.2.1*
		Common Container Functionality Guide	*15.9*

Item	Overview of changes	Reference manual	Reference
		Command Reference Guide	*cjimportwar (Import a WAR application)*
Starting and stopping by synchronous execution of the operation management functionality	Starting and stopping of the operation management functionality (Management Server and Administration Agent) was added to the options to be synchronously executed.	*Operation, Monitoring, and Linkage Guide*	*2.6.1, 2.6.2, 2.6.3, 2.6.4*
		Command Reference Guide	*adminagentctl (start or stop Administration Agent)*, *mngautorun (Set up/canceling the set up of autostart and autorestart)*, *mngsvrctl (start, stop, or setup Management Server)*
Forced release of Explicit memory blocks by using the Explicit Memory Management functionality	The release processing of Explicit memory blocks can now be performed any time by using the `javagc` command.	*Expansion Guide*	*7.6.1, 7.9*
		Command Reference Guide	*javagc (forcibly perform GC)*

(6) Other purposes

The following table shows the items that were changed for other purposes.

Table H–15: Changes made for other purposes

Item	Overview of changes	Reference manual	Reference
Acquisition of definition information	The `snapshotlog` command (collect snapshot log) can now be used to collect definition files only.	*Maintenance and Migration Guide*	*2.3*
		Command Reference Guide	*snapshotlog (collect snapshot logs)*
Log output of the `cjenvsetup` command	Execution information for setup (the `cjenvsetup` command) of the Component Container Administrator can now be output to the message log	*System Setup and Operation Guide*	*4.1.4*
		Maintenance and Migration Guide	*4.20*

Item	Overview of changes	Reference manual	Reference
		Command Reference Guide	*cjenvsetup (set up Component Container Administrator)*
Support for BIG-IP v11	BIG-IP v11 was added to the types of the available load balancers.	*System Setup and Operation Guide*	*4.7.2*
		Virtual System Setup and Operation Guide	*2.1*
Output of CPU time to the event log of the Explicit Memory Management functionality	The CPU time spent on the processing to release the Explicit memory block can now be output to the event log of the Explicit Memory Management functionality.	*Maintenance and Migration Guide*	*5.11.3*
Extension of the user-extended performance analysis trace functionality	The following changes were made to the user-extended performance analysis trace functionality: • In addition to the usual unit of method, trace targets can now also be specified in the unit of package or class. • The range of available event IDs was extended. • The limitation on the number of rows that can be specified in the settings file of user-extended performance analysis trace was released. • You can now specify the trace acquisition level in the settings file of user-extended performance analysis trace.	*Maintenance and Migration Guide*	*7.5.2, 7.5.3, 8.23.1*
Improved information analysis when using asynchronous invocation of Session Bean	The requests at the invocation source can now be compared against the requests at the invocation destination by using root application information of PRF trace.	*EJB Container Functionality Guide*	*2.17.3*

H.6 Main functionality changes in 09-00

(1) Simplifying implementation and setup

The following table shows the items that were changed to simplify implementation and setup.

Table H–16: Changes made for simplifying implementation and setup

Item	Overview of changes	Reference manual	Reference
Changed the unit of the setup and operation target in the virtual environment	The units to be operated when you set up and operate the virtual environment have been changed from the virtual server to the virtual server group. You can now define the information of a virtual server group in a file and register multiple virtual servers to a management unit in a batch.	*Virtual System Setup and Operation Guide*	*1.1.2*
Removed the restrictions on environments that can be built by using the Setup Wizard	Removed the restrictions on the environment that can be built by using the Setup Wizard. You can now unset up and set up any of the existing environments set up with a different functionality, by using the Setup Wizard.	*System Setup and Operation Guide*	*2.2.7*
Simplification of the procedure for removing an installed environment	Added functionality (`mngunsetup` command) for removing the system environment that is set up by using Management Server, thereby simplifying the removal procedure.	*System Setup and Operation Guide*	*4.1.37*

Item	Overview of changes	Reference manual	Referenc e
		Management Portal User Guide	*3.6*, *5.4*
		Command Reference Guide	*mngunsetup (Deleting the Management Server configuration environment)*

(2) Supporting the standard and existing functionality

The following table shows the items that were changed to support the standard and existing functionality.

Table H–17: Changes made for supporting the standard and existing functionality

Item	Overview of changes	Reference manual	Referenc e
Supporting Servlet 3.0	Servlet 3.0 is now supported.	*Web Container Functionality Guide*	*Chapter 7*
Supporting EJB 3.1	EJB 3.1 is now supported.	*EJB Container Functionality Guide*	*Chapter 2*
Supporting JSF 2.1	JSF 2.1 is now supported.	*Web Container Functionality Guide*	*Chapter 3*
Supporting JSTL 1.2	JSTL 1.2 is now supported.	*Web Container Functionality Guide*	*Chapter 3*
Supporting CDI 1.0	CDI 1.0 is now supported.	*Common Container Functionality Guide*	*Chapter 8*
Using Portable Global JNDI names	You can now look up objects for which Portable Global JNDI names are used.	*Common Container Functionality Guide*	*2.4*
Supporting JAX-WS 2.2	JAX-WS 2.2 is now supported.	*Web Service Development Guide*	*1.1*, *16.1.5*, *16.1.7*, *16.2.1*, *16.2.6*, *16.2.10*, *16.2.12*, *16.2.13*, *16.2.14*, *16.2.16*, *16.2.17*, *16.2.18*, *16.2.20*, *16.2.22*, *19.1*, *19.2.3*, *37.2*, *37.6.1*, *37.6.2*, *37.6.3*

Item	Overview of changes	Reference manual	Referenc e
Supporting JAX-RS 1.1	JAX-RS 1.1 is now supported.	*Web Service Development Guide*	*1.1*, *1.2.2*, *1.3.2*, *1.4.2*, *1.5.1*, *1.6*, *2.3*, *Chapter 11*, *Chapter 12*, *Chapter 13*, *Chapter 17*, *Chapter 24*, *Chapter 39*

(3) Maintaining and improving reliability

The following table shows the items that were changed to maintain and improve reliability.

Table H–18: Changes made for maintaining and improving reliability

Item	Overview of changes	Reference manual	Referenc e
Using TLSv1.2 for SSL/TLS communication	You can now use RSA BSAFE SSL-J to execute the SSL/TLS communication with a security protocol containing TLSv1.2.	--	--

Legend:
--: This functionality has been deleted in version 09-70.

(4) Maintaining and improving operability

The following table shows the items that were changed to maintain and improve operability.

Table H–19: Changes made for maintaining and improving operability

Item	Overview of changes	Reference manual	Referenc e
Monitoring the total pending queues of the entire Web container	You can now monitor the total pending queues of the entire Web container when the total is output to the operation information.	*Operation, Monitoring, and Linkage Guide*	*Chapter 3*
Output of the trace based performance analysis for applications (user extended trace)	The trace based performance analysis used for analyzing the processing performance of user-developed applications can now be output without changing the applications.	*Maintenance and Migration Guide*	*Chapter 7*
Operations performed by using the user script in a virtual environment	The user-created script (user script) can now be executed on a virtual server at any time	*Virtual System Setup and Operation Guide*	*7.8*
Improving the management portal	Changes have been made so that the messages describing the procedure are now displayed on the following management portal windows: • Deploy the preference information window	*Management Portal User Guide*	*10.10.1*, *11.9.2*, *11.10.2*, *11.10.4*, *11.10.6*, *11.11.2*,

Item	Overview of changes	Reference manual	Referenc e
	• Start window for the Web server, J2EE server, and SFO server • Batch start, batch restart, and startup windows for Web server cluster and J2EE server cluster		*11.12.2, 11.12.4, 11.12.6*
Adding restart functionality for operation management functionality	You can now set the automatic restart in the operation management functionality (Management Server and Administration Agent). Due to the automatic restart functionality, it is now possible to continue operations even if an error occurs in the operation management functionality. The procedure for automatic start setting has also been changed.	*Operation, Monitoring, and Linkage Guide*	*2.4.1, 2.4.2, 2.6.3, 2.6.4*
		Command Reference Guide	*mngautorun (Set up/ canceling the set up of autostart and autorestart)*

(5) Other purposes

The following table shows the items that were changed for other purposes.

Table H–20: Changes made for other purposes

Item	Overview of changes	Reference manual	Referenc e
Changing the unit for switching log output files	You can now change log output files by date.	*Maintenance and Migration Guide*	*3.2.1*
Changing the Web server name	The name of the Web server included in Application Server has been changed to uCosminexus HTTP Server.	*HTTP Server User Guide*	--
Supporting a direct connection with the API (SOAP architecture) in BIG-IP	Direct connection is now supported by using APIs (SOAP architecture) in BIG-IP (load balancer). Note that the procedure for setting up the connection environment of the load balancer has been changed for using a direct connection through APIs.	*System Setup and Operation Guide*	*4.7.3, Appendix J*
		Virtual System Setup and Operation Guide	*2.1, Appendix C*
		Security Management Guide	*8.2, 8.4, 8.5, 8.6, 18.2.1, 18.2.2, 18.2.3*

Legend:

--: Reference the entire manual.

H.7 Main functionality changes in 08-70

(1) Simplifying implementation and setup

The following table shows the items that were changed to simplify implementation and setup.

Table H–21: Changes made for simplifying implementation and setup

Item	Overview of changes	Reference manual	Referenc e
Improving the Management portal	The changes have been made to enable the user to set the property (settings of the Connector property file) for defining the resource adapter attributes and perform the connection test in the management portal window. Also, you can now use the Management portal window to upload J2EE applications (ear file and zip file) on Management Server.	*First Step Guide*	3.5
		Management Portal User Guide	--
Adding functionality for implicitly importing the import property for the `page/tag` directive	You can now use the functionality for implicitly importing the import property of the `page/tag` directive.	*Web Container Functionality Guide*	2.3.7
Support for automating the environment settings corresponding to the JP1 products in a virtual environment	The changes have been made so that when Application Server is set up on a virtual server, the environment settings of JP1 products can be automatically set for the virtual server by using the hook script.	*Virtual System Setup and Operation Guide*	7.7.2
Improving the Integrated user management functionality	When using a database in a user information repository, you can now connect to the database with the JDBC driver of database products. The database connection through the JDBC driver of Cosminexus DABroker Library is not supported anymore. You can now set the integrated user management functionality using the Easy Setup definition file and the management portal windows. The Active Directory now supports double byte characters such as Japanese language in DN.	*Security Management Guide*	Chapter 5, 14.2.2
		Management Portal User Guide	3.5, 10.8.1
Enhancing HTTP Server settings	You can now directly set the directive (settings of `httpsd.conf`) that defines the operation environment of HTTP Server using the Easy Setup definition file and the management portal windows.	*System Setup and Operation Guide*	4.1.21
		Management Portal User Guide	10.9.1
		Definition Reference Guide	4.10

Legend:

--: Reference the entire manual.

(2) Supporting the standard and existing functionality

The following table shows the items that were changed to support the standard and existing functionality.

Table H–22: Changes made for supporting the standard and existing functionality

Item	Overview of changes	Reference manual	Referenc e
Adding the items to be specified in `ejb-jar.xml`	You can now specify a class level interceptor and a method level interceptor in `ejb-jar.xml`.	*EJB Container Functionality Guide*	2.15
Supporting the parallel copy garbage collection	You can now select the parallel copy garbage collection.	*Definition Reference Guide*	14.5
Supporting the global transaction of the Inbound resource adapter conforming to the Connector 1.5 specifications	You can now use `Transacted Delivery` in resource adapters conforming to the Connector 1.5 specifications. This enables the participation of EIS invoking the Message-driven Bean in the global transaction.	*Common Container Functionality Guide*	3.16.3

Item	Overview of changes	Reference manual	Referenc e
Supporting MHP of a TP1 inbound adapter	You can now use MHP as the OpenTP1client that invokes Application Server by using the TP1 inbound adapter.	*Common Container Functionality Guide*	*Chapter 4*
Supporting the FTP inbound adapter of the `cjrarupdate` command	An FTP inbound adapter has been added to the resource adapters that can be upgraded by using the `cjrarupdate` command.	*Command Reference Guide*	*2.2*

(3) Maintaining and improving reliability

The following table shows the items that were changed to maintain and improve reliability.

Table H–23: Changes made for maintaining and improving reliability

Item	Overview of changes	Reference manual	Referenc e
Improving the database session failover functionality	The user can now select a mode that does not obtain the lock of the database in which the global session information is stored in a performance-centric system. Also, exclusive requests for references can now be defined without updating the database.	*Expansion Guide*	*Chapter 6*
Expansion of a process for the OutOfMemory handling functionality	A process for the OutOfMemory handling functionality has been added.	*Maintenance and Migration Guide*	*2.5.4*
		Definition Reference Guide	*14.2*
Adding the memory saving functionality for the Explicit heap used in an HTTP session	A functionality to minimize the amount of the Explicit heap memory used in the HTTP session has been added.	*Expansion Guide*	*7.11*

(4) Maintaining and improving operability

The following table shows the items that were changed to maintain and improve operability.

Table H–24: Changes made for maintaining and improving operability

Item	Overview of changes	Reference manual	Referenc e
Supporting an user authentication using JP1 products in the virtual environment (handling cloud operations)	The administration and authentication of users using a virtual server manager can now be performed by using the authentication server of JP1 products when integrating JP1.	*Virtual System Setup and Operation Guide*	*1.2.2*, *Chapter 3*, *Chapter 4*, *Chapter 5*, *Chapter 6*, *7.9*

(5) Other purposes

The following table shows the items that were changed for other purposes.

Table H–25: Changes made for other purposes

Item	Overview of changes	Reference manual	Referenc e
Supporting a direct connection using APIs (REST Architecture) to the load balancing functionality	A direct connection using APIs (REST architecture) is now supported as a method to connect to the Load balancing functionality. ACOS (AX2500) has been added in the types of available load balancing functions.	*System Setup and Operation Guide*	*4.7.2*, *4.7.3*
		Virtual System Setup and Operation Guide	*2.1*
		Definition Reference Guide	*4.2.4*
Improving response timeout when collecting snapshot logs and collection targets	You can now stop the snapshot log collection (timeout) at a specified time. The contents collected as primary delivery data have been changed.	*Maintenance and Migration Guide*	*Appendix A*

H.8 Main functionality changes in 08-53

(1) Simplifying implementation and setup

The following table shows the item that was changed to simplify implementation and setup.

Table H–26: Changes made for simplifying implementation and setup

Item	Overview of changes	Reference manual	Referenc e
Setting up a virtual environment supporting various hypervisors	You can now set up Application Server on a virtual server implemented by using various hypervisors. An environment in which multiple hypervisors co-exist is also supported now.	*Virtual System Setup and Operation Guide*	*Chapter 2*, *Chapter 3*, *Chapter 5*

(2) Supporting the standard and existing functionality

The following table shows the items that were changed to support the standard and existing functionality.

Table H–27: Changes made for supporting the standard and existing functionality

Item	Overview of changes	Reference manual	Referenc e
Invocation from OpenTP1 supporting the transaction integration	You can now integrate transactions when the Message-driven Bean running on Application Server is invoked from OpenTP1	*Common Container Functionality Guide*	*Chapter 4*
JavaMail	The mail receiving functionality, which uses the APIs conforming to JavaMail 1.3 by integrating with the mail server conforming to POP3, is now available.	*Common Container Functionality Guide*	*Chapter 7*

(3) Maintaining and improving reliability

The following table shows the items that were changed to maintain and improve reliability.

Table H–28: Changes made for maintaining and improving reliability

Item	Overview of changes	Reference manual	Referenc e
Enhancing the JavaVM troubleshooting functionality	You can now use the following functionality as the JavaVM troubleshooting functionality: • You can now change the operations when `OutOfMemoryError` occurs. • You can now set up an upper limit for the amount of the C heap allocated during the JIT compilation. • You can now set up the maximum thread count. • Output items of the extended verbosegc information have been extended.	*Maintenance and Migration Guide*	*Chapter 4*, *Chapter 5*, *Chapter 9*

(4) Maintaining and improving operability

The following table shows the items that were changed to maintain and improve operability.

Table H–29: Changes made for maintaining and improving operability

Item	Overview of changes	Reference manual	Referenc e
Supporting JP1/ITRM	JP1/ITRM, a product that uniformly manages the IT resources, is now supported.	*Virtual System Setup and Operation Guide*	*1.3*, *2.1*

(5) Other purposes

The following table shows the items that were changed for other purposes.

Table H–30: Changes made for other purposes

Item	Overview of changes	Reference manual	Referenc e
Supporting Microsoft IIS 7.0 and Microsoft IIS 7.5	Microsoft IIS 7.0 and Microsoft IIS 7.5 are now supported as Web servers.	--	--
Supporting HiRDB Version 9 and SQL Server 2008	The following products are now supported as the database: • HiRDB Server Version 9 • HiRDB/Developer's Kit Version 9 • HiRDB/Run Time Version 9 • SQL Server 2008 Also, SQL Server JDBC Driver is now supported as the JDBC driver corresponding to SQL Server 2008.	*Common Container Functionality Guide*	*Chapter 3*

Legend:
--: Not applicable.

H.9 Main functionality changes in 08-50

(1) Simplifying implementation and setup

The following table shows the item that was changed to simplify implementation and setup.

Table H–31: Changes made for simplifying implementation and setup

Item	Overview of changes	Reference manual	Referenc e
Changing the tags with mandatory specification in `web.xml` at the Web service provider side	Changed the specification of the `listener` tag, `servlet` tag, and `servlet-mapping` tag in `web.xml` at the Web Service provider side from mandatory to optional.	*Definition Reference Guide*	*2.2.3*
Using the network resources of the logical server	Added a functionality for accessing the network resources and network drives on other hosts from the J2EE application.	*Operation, Monitoring, and Linkage Guide*	*1.2.3*, *5.2*, *5.7*
Simplifying the execution procedure of sample programs	Simplified the execution procedure of sample programs by providing some sample programs in the EAR format.	*First Step Guide*	*3.5*
		System Setup and Operation Guide	*Appendix L*
Improving the operation of the window of management portal	The default window refresh intervals were changed from `Do not refresh` to `3 seconds`.	*Management Portal User Guide*	*7.4.1*
Improving the Setup Wizard completion window	Changes have been made to display the Easy Setup definition file and the HITACHI Connector Property file used for setup in the window displayed during completion of the Setup Wizard.	*System Setup and Operation Guide*	*2.2.6*
Setting up the virtual environment	Added the procedure for setting up Application Server on a virtual server implemented by using hypervisors.#	*Virtual System Setup and Operation Guide*	*Chapter 3*, *Chapter 5*

#

When setting up in the 08-50 mode, see *Appendix D Settings for Using the Virtual Server Manager in the 08-50 Mode* in the *uCosminexus Application Server Virtual System Setup and Operation Guide*.

(2) Supporting the standard and existing functionality

The following table shows the items that were changed to support the standard and existing functionality.

Table H–32: Changes made for supporting the standard and existing functionality

Item	Overview of changes	Reference manual	Referenc e
Supporting invocation from OpenTP1	Enabled the invocation of the Message-driven Bean operating on the Application Server from OpenTP1.	*Common Container Functionality Guide*	*Chapter 4*
Supporting JMS	Enabled the use of the Cosminexus JMS provider functionality compliant with JMS 1.1 specifications.	*Common Container Functionality Guide*	*Chapter 6*
Supporting Java SE 6	Enabled the use of the Java SE 6 functionality.	*Maintenance and Migration Guide*	*5.5*, *5.8.1*
Supporting the use of generics	Enabled the use of generics in EJB.	*EJB Container Functionality Guide*	*4.2.18*

(3) Maintaining and improving reliability

The following table shows the items that were changed to maintain and improve reliability.

Table H–33: Changes made for maintaining and improving reliability

Item	Overview of changes	Reference manual	Referenc e
Improving the usability of the Explicit Memory Management functionality	Enabled easy usage of the Explicit Memory Management functionality by using the Automatic Deployment setup file.	*System Design Guide*	*7.2*, *7.7.3*, *7.11.4*, *7.12.1*
		Expansion Guide	*Chapter 7*
Blocking the database session failover functionality in the URI unit	Enabled specification of requests that are to be set outside the scope of the database session failover functionality during the use of the functionality in the URI unit.	*Expansion Guide*	*5.6.1*
Monitoring failures in the virtual environment	Enabled the monitoring of virtual servers and detecting the occurrence of failures in a virtual system.	*Virtual System Setup and Operation Guide*	*Appendix D*

(4) Maintaining and improving operability

The following table shows the items that were changed to maintain and improve operability.

Table H–34: Changes made for maintaining and improving operability

Item	Overview of changes	Reference manual	Reference
Omitting the management user account	Enabled the omission of the input of the user login ID and password in the management portal, commands of the Management Server, and commands of the Smart Composer functionality.	*System Setup and Operation Guide*	*4.1.15*
		Management Portal User Guide	*2.2*, *7.1.1*, *7.1.2*, *7.1.3*, *8.1*, *8.2.1*, *Appendix F.2*
		Command Reference Guide	*1.4*, *mngsvrctl (Starting, stopping, and setting up the Management Server)*, *mngsvrutil (Management commands of the Management Server)*, *8.3*, *cmx_admin_passwd (Setting the management user account of the Management Server)*
Operating the virtual environment	Added the procedure for executing batch startup, batch stop, scale-in, and scale-out of multiple virtual servers in a virtual system.#	*Virtual System Setup and Operation Guide*	*Chapter 4*, *Chapter 6*

#

When operating in the 08-50 mode, see *Appendix D Settings for Using the Virtual Server Manager in the 08-50 Mode* in the *uCosminexus Application Server Virtual System Setup and Operation Guide*.

(5) Other purposes

The following table shows the items that were changed for other purposes.

Table H–35: Changes made for other purposes

Item	Overview of changes	Reference manual	Referenc e
Statistical functionality for unused objects within the Tenured area	Enabled the identification of only unused objects within the Tenured area.	*Maintenance and Migration Guide*	*9.8*

Item	Overview of changes	Reference manual	Referenc e
Base point object list output functionality for Tenured increment factors	Enabled the output of information of the object that acts as the base point of unused objects identified using the statistical functionality for unused objects within the Tenured area.		*9.9*
Class-wise statistical information analysis functionality	Enabled the output of class-wise statistical information in the CSV format.		*9.10*
Cluster node switching due to detection of excess auto restart frequency of the logical server	Enabled node switching when the logical server stops abnormally (when the auto restart frequency is exceeded, or when a failure is detected when the auto restart frequency is set to 0) in the case of cluster configuration in which Management Server is a target for monitoring node switching.	*Operation, Monitoring, and Linkage Guide*	*18.4.3*, *18.5.3*, *16.2.2*, *16.3.3*, *16.3.4*
Node switching system for the host unit management model	Enabled node switching for the host unit management model during system operation linked with cluster software.		*Chapter 16*
Supporting ACOS (AX2000, BS320)	Added ACOS (AX2000, BS320) in the types of available load balancers.	*System Setup and Operation Guide*	*4.7.2*, *4.7.3*, *4.7.5*, *4.7.6*, *Appendix J*, *Appendix J.2*
		Definition Reference Guide	*4.2.4*, *4.3.2*, *4.3.4*, *4.3.5*, *4.3.6*, *4.7.1*
Adding transaction attributes that can be specified in a Stateful Session Bean (SessionSynchronization) when performing transaction management in CMT	Changes have been made to specify `Supports`, `NotSupported`, and `Never` as transaction attributes in a Stateful Session Bean (SessionSynchronization) when performing transaction management in CMT.	*EJB Container Functionality Guide*	*2.7.3*
Forced termination of the Administration Agent during the occurrence of OutOfMemoryError	Enabled forced termination of the Administration Agent during the occurrence of OutOfMemoryError in JavaVM.	*Maintenance and Migration Guide*	*2.5.5*
Asynchronous parallel processing of threads	Enabled the implementation of the asynchronous timer processing and asynchronous thread processing using TimerManager and WorkManager.	*Expansion Guide*	--

H.10 Main functionality changes in 08-00

(1) Improving development productivity

The following table shows the items that were changed to improve development productivity.

Table H–36: Changes made for improving development productivity

Item	Overview of changes	Reference manual	Reference
Simplification of migration from other Application Server products	Enabled the use of the following functionality for smooth migration from other Application Server products: • Enabled the judgment of upper limit of the HTTP sessions through an exception. • Enabled the inhibition of occurrence of a translation error when the ID of JavaBeans is duplicate, and when the upper-case characters and lower-case characters are different in the attribute name of the custom tag and in the TLD definition.	*Web Container Functionality Guide*	*2.3*, *2.7.5*
Provision of `cosminexus.xml`	Enabled the start of J2EE applications without setting the properties after importing them into the J2EE server by describing the properties unique to the Cosminexus Application Server in `cosminexus.xml`.	*Common Container Functionality Guide*	*13.3*

(2) Supporting the standard functionality

The following table shows the items that were changed to support the standard functionality.

Table H–37: Changes made for supporting the standard functionality

Item	Overview of changes	Reference manual	Referenc e
Servlet 2.5 support	Supported Servlet 2.5.	*Web Container Functionality Guide*	*2.2*, *2.5.4*, *2.6*, *Chapter 7*
JSP 2.1 support	Supported JSP 2.1.	*Web Container Functionality Guide*	*2.3.1*, *2.3.3*, *2.5*, *2.6*, *Chapter 7*
JSP debug	Enabled the execution of JSP debugging in the development environment using MyEclipse. #	*Web Container Functionality Guide*	*2.4*
Storage of the tag library in the library JAR, and TLD mapping	Enabled the search of TLD files within the library JAR by the Web container during the start of the Web application, and their subsequent automatic mapping, when the tag libraries are stored in the library JAR.	*Web Container Functionality Guide*	*2.3.4*
Omission of `application.xml`	Enabled the omission of `application.xml` in a J2EE application.	*Common Container Functionality Guide*	*13.4*
Combined use of annotations and DDs	Enabled the combined use of annotations and DDs, and also enabled the update of annotation contents in the DD.	*Common Container Functionality Guide*	*14.5*
Conformance of annotations to Java EE 5 standard (default interceptor)	Enabled the storage of the default interceptor in the library JAR. Also enabled the execution of DI from the default interceptor.	*Common Container Functionality Guide*	*13.4*
Reference resolution of `@Resource`	Enabled the reference resolution of resources with `@Resource`.	*Common Container Functionality Guide*	*14.4*
JPA support	Supported JPA specifications.	*Common Container Functionality Guide*	*Chapter 5*

#
In version 09-00 and later, you can use the JSP debug functionality in the development environment using WTP.

(3) Maintaining and improving reliability

The following table shows the items that were changed to maintain and improve reliability.

Table H–38: Changes made for maintaining and improving reliability

Item	Overview of changes	Reference manual	Referenc e
Persistence of session information	Enabled the inheritance of session information of an HTTP session by saving the information in the database.	*Expansion Guide*	*Chapter 5*, *Chapter 6*
Inhibition of a Full GC	Enabled the inhibition of occurrence of a Full GC by deploying the objects responsible for the Full GC outside the Java heap.	*Expansion Guide*	*Chapter 7*
Client performance monitor	The time required for client processing can now be checked and analyzed.	--	--

Legend:

--: This functionality has been deleted in version 09-00.

(4) Maintaining and improving operability

The following table shows the items that were changed to maintain and improve operability.

Table H–39: Changes made for maintaining and improving operability

Item	Overview of changes	Reference manual	Referenc e
Improving the operability of applications on the management portal	The server management commands and management portal can now be interoperated for application and resource operations.	*Management Portal User Guide*	*1.1.3*

(5) Other purposes

The following table shows the items that were changed for other purposes.

Table H–40: Changes made for other purposes

Item	Overview of changes	Reference manual	Referenc e
Deletion if disabled HTTP Cookies	Enabled the deletion of disabled HTTP Cookies.	*Web Container Functionality Guide*	*2.7.4*
Failure detection in the Naming Service	Enabled prompt detection of the error by the EJB client, when a failure occurs in the Naming Service.	*Common Container Functionality Guide*	*2.9*
Connection failure detection timeout	Enabled the specification of the timeout period for a connection failure detection timeout.	*Common Container Functionality Guide*	*3.15.1*
Oracle11g support	Enabled the use of Oracle11g as a database.	*Common Container Functionality Guide*	*Chapter 3*
Scheduling of batch processing	Enabled the scheduling of execution of batch applications by CTM.	*Expansion Guide*	*Chapter 4*
Batch processing log	The retry frequency and retry interval can now be specified for the size and number of log files of the batch execution command and the failure of exclusive processing of the log.	*Definition Reference Guide*	*3.2.5*

Item	Overview of changes	Reference manual	Referenc e
snapshot log	Changed the collection contents of the snapshot log.	*Web Container Functionality Guide*	*Appendix A.1*, *Appendix A.2*
Publication of protected area of method cancellation	Published the contents of protected area list that is outside the scope of method cancellation.	*Operation, Monitoring, and Linkage Guide*	*Appendix C*
Pre-statistical garbage collection selection functionality	Enabled the selection of whether or not to execute a garbage collection before the output of class-wise statistical information.	*Web Container Functionality Guide*	*9.7*
Tenuring distribution information output functionality of the Survivor area.	Enabled the output of tenuring distribution information of Java objects of the Survivor area to the Hitachi JavaVM log file.	*Web Container Functionality Guide*	*9.11*
Finalize retention cancellation functionality	Enabled the cancellation of retention of the finalize processing of JavaVM after monitoring its status.	--	--
Change of the maximum heap size of server management commands	Changed the maximum heap size used by server management commands.	*Definition Reference Guide*	*5.2.1*, *5.2.2*
Action for cases when un-recommended display names are specified	Provided the output of messages when un-recommended display names are specified in J2EE applications.	*Messages*	*KDJE42374-W*

Legend:

--: This functionality is deleted in 09-00.

I. Glossary

Terminology used in this manual

For the terms used in the manual, see the *uCosminexus Application Server and BPM/ESB Platform Terminology Guide*.

Index