HITACHI
Inspire the Next

uCosminexus Application Server

Maintenance and Migration Guide

3021-3-J11-10(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

HITACHI, Cosminexus, DABroker, HA Monitor, HiRDB, JP1, OpenTP1, TPBroker, uCosminexus are either trademarks or registered trademarks of Hitachi, Ltd. in Japan and other countries.

AIX is a trademark of International Business Machines Corporation, registered in many jurisdictions worldwide.

AMD is a trademark (or registered trademark) of Advanced Micro Devices, Inc.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Microsoft, Active Directory are trademarks of the Microsoft group of companies.

Microsoft, Excel are trademarks of the Microsoft group of companies.

Microsoft, Internet Explorer are trademarks of the Microsoft group of companies.

Microsoft, SQL Server are trademarks of the Microsoft group of companies.

Microsoft, Windows are trademarks of the Microsoft group of companies.

Microsoft, Windows Server are trademarks of the Microsoft group of companies.

Microsoft is a trademark of the Microsoft group of companies.

Oracle, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Red Hat is a registered trademark of Red Hat, Inc. in the United States and other countries.

Red Hat Enterprise Linux is a registered trademark of Red Hat, Inc. in the United States and other countries.

UNIX is a trademark of The Open Group.

Other company and product names mentioned in this document may be the trademarks of their respective owners.

Eclipse is an open development platform for tools integration provided by Eclipse Foundation, Inc., an open source community for development tool providers.

This product includes software developed by the Apache Software Foundation (http://www.apache.org/).

■ Microsoft product screen shots

Microsoft product screen shots reprinted with permission from Microsoft Corporation.

■ Issued

Aug. 2022: 3021-3-J11-10(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

Appendixes 709

1 Application Server Functionality

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

1.1 Classifications of functionality

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

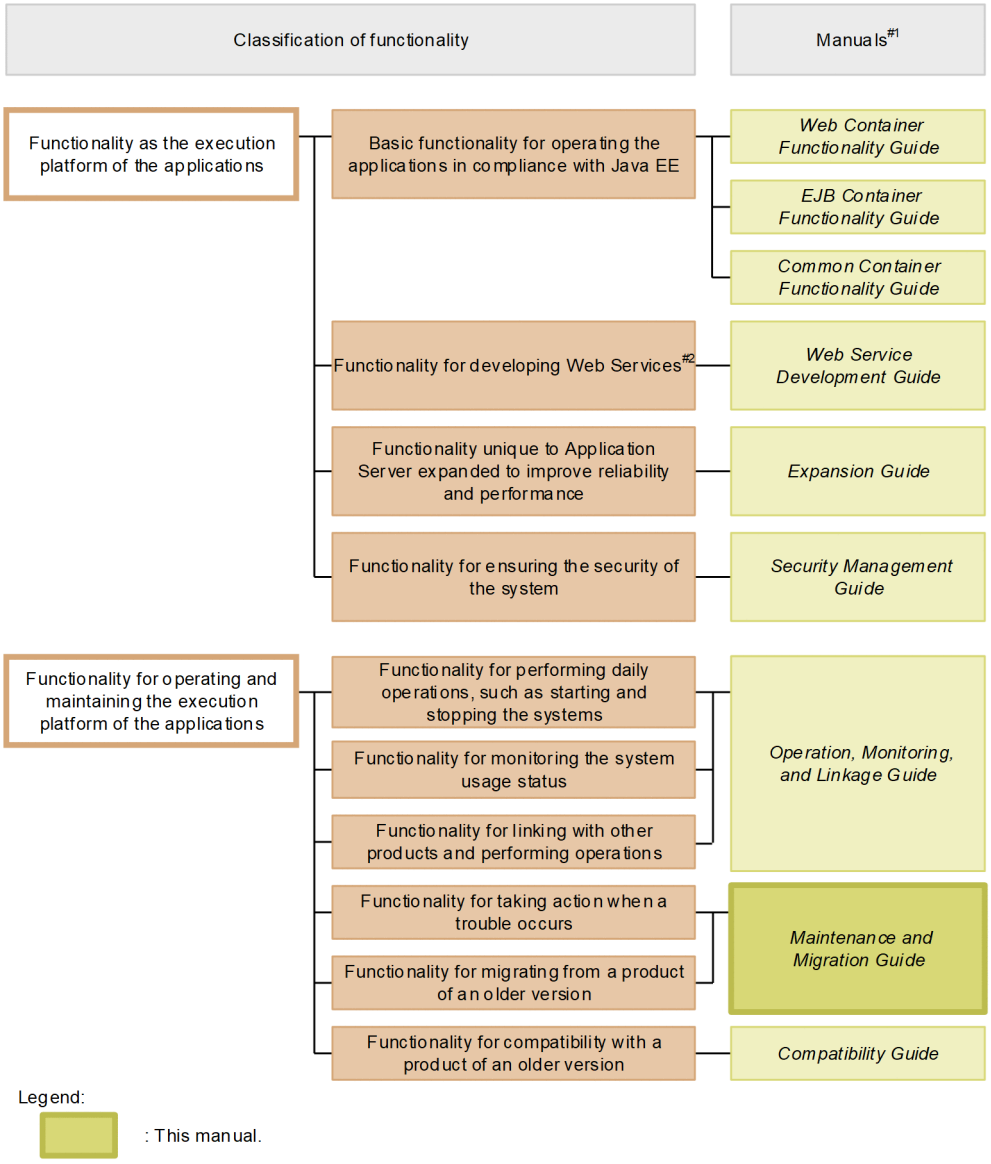
The following are the broad classifications of Application Server functionality:

- Functionality that serves as an execution platform for the applications
- Functionality that is used for operating and maintaining the execution platform for the applications

The above-mentioned functionality can be further classified according to the positioning and the intended use of the functionality. Application Server manuals are provided according to the classification of the functionality.

The following figure shows the classification of the Application Server functionality and the set of manuals corresponding to the functionality.

Figure 1–1: Classification of the Application Server functionality and the set of manuals corresponding to each functionality



#1

uCosminexus Application Server has been omitted from the manual names mentioned in the *Manuals* column.

#2

You can execute SOAP Web Services and RESTful Web Services with Application Server. Depending on the purpose, see the following manuals other than the *uCosminexus Application Server Web Service Development Guide*.

To develop and execute SOAP applications

- *uCosminexus Application Server SOAP Application Development Guide*

To ensure security for SOAP Web Services and SOAP applications

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

To learn about XML processing in detail

- *uCosminexus Application Server XML Processor User Guide*

The following subsections describe the classifications of functionality and the manuals corresponding to the functionality.

1.1.1 Functionality as an application execution platform

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

You must determine whether you want to use functionality that serves as the execution platform for the applications, even before you perform the system building or application development.

The following are the classification-wise descriptions of functionality that serve as the application execution platform:

(1) Basic functionality to operate applications (basic development functions)

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

Application Server provides a J2EE server that supports Java EE 7. The J2EE server provides functionality that is compliant with the standard specifications and is independent of Application Server.

The basic development functionality can be further classified into three types according to the types of the J2EE applications for which you use functionality. The manuals for Application Server function guide have been separated according to this classification.

The following is an overview of each classification:

- Functionality for executing the Web applications (Web containers)

  This classification includes the Web container functionality that serves as the execution platform for Web applications and functionality executed by linking the Web containers and the Web servers.
- Functionality for executing the Enterprise Bean (EJB containers)

  This classification includes the EJB container functionality that serves as a platform for executing Enterprise Beans. This classification also includes the EJB client functionality for invoking the Enterprise Beans.
- Functions used in both Web applications and Enterprise Beans (Container common function)

  This classification includes functionality that can be used in the Web applications and the Enterprise Beans running on the Web containers and the EJB containers respectively.

(2) Functionality for developing Web Services

This includes the functionality for the execution and development environment of Web Services.

The following engines are provided with Application Server:

- JAX-WS engine that binds the SOAP messages in accordance with the JAX-WS specifications
- JAX-RS engine that binds the RESTful HTTP messages in accordance with the JAX-RS specifications

(3) Application Server independent functionality extended for improving reliability and performance (expansion functionality)

This includes the functionality extended independently on Application Server. This also includes the functionality implemented by using non-J2EE server processes such as batch server, CTM, and database.

On Application Server, various functionality are extended to improve reliability of the system and to implement stable operations. Furthermore, functionality is also extended to operate applications other than J2EE applications (batch applications) in the Java environment.

(4) Functionality for ensuring the security of a system (security management functionality)

This is the functionality used for ensuring the security of an Application Server-based system. This includes functionality such as the authentication functionality used for preventing unauthorized access and the encryption functionality used for preventing information leakage from communication channels.

1.1.2 Functionality for operating and maintaining the application execution platform

This functionality is used for effectively operating and maintaining the application execution platform. You use this functionality, after starting the system operations, as and when required. However, depending on the functionality, you must implement the settings and applications in advance.

The following are the classification-wise descriptions of functionality used for operating and maintaining the application execution platform:

(1) Functionality used for daily operations, such as starting and stopping the systems (operation functionality)

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

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

This classification includes the functionality used for monitoring the system usage and resource depletion. This classification also includes functionality to output the information used in monitoring the system operation history.

(3) Functionality for operating the systems by linking with other products (linkage functionality)

This classification includes the functionality to be linked and implemented with other products, such as JP1 and cluster software.

(4) Functionality for troubleshooting (maintenance functionality)

This classification includes the functionality used for troubleshooting. This functionality also includes the functionality used to output the information that will be referenced during the troubleshooting.

(5) Functionality for migrating from products of earlier versions (migration functionality)

This classification includes the functionality used for migrating from an older Application Server to a new Application Server.

(6) Functionality for compatibility with products of earlier versions (compatibility functionality)

This classification includes the functionality used for compatibility with earlier versions of Application Server. For the compatibility functionality, we recommend the migration with the corresponding recommended functionality.

1.1.3 Functionality and corresponding manuals

The function guides for Application Server have been separated according to the classifications of functionality.

The following table describes the classifications of functionality and the manuals corresponding to the functionality.

Table 1–1: Classifications of functionality and corresponding manuals describing the functionality

Category	Functionality	Reference manual#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 during Enterprise Bean implementation	
	Naming management	*Common Container Functionality Guide*
	Managing resource connection and transactions	
	Invoking Application Server from OpenTP1 (TP1 inbound integrated function)	
	Using JPA 2.1	
	Cosminexus JMS Provider	
	Using Java Mail	
	Using CDI with Application Server	
	Using Bean Validation with Application Server	
	Java Batch	
	JSON-P	
	Concurrency Utilities	

Category	Functionality	Reference manual[#1]
	Managing application attributes	
	Using annotations	
	Formatting and deploying J2EE applications	
	Container extension library	
Extended functionality	Executing applications using the batch server	*Expansion Guide*
	Scheduling and load balancing requests using CTM	
	Scheduling the batch applications	
	Inheriting the session information between the J2EE servers (Session failover functionality)	
	Database session failover functionality	
	Controlling Full GC using the explicit Explicit Memory Management functionality	
	Output of the application user log	
Security management functionality	Authentication using integrated user management	*Security Management Guide*
	Authentication using application settings	
	Using TLSv 1.2 for SSL/TLS communication	
	Controlling with the management functionality of load balancers that use API-based direct connections	
Operation functionality	Starting and stopping the system	*Operation, Monitoring, and Linkage Guide*
	Managing J2EE applications	
Watch functionality	Monitoring statistics information (statistics collection functionality)	
	Monitoring resource depletion	
	Database audit trail linkage function	
	Output of statistical information by operation management command	
	Notification of management event and auto execution of process by management action	
	Statistics collection of CTM	
	Output of console log	
Linkage functionality	Operation of systems linked with JP1	
	Centralized monitoring of system (Integrating with JP1/IM)	
	Automatic operations of systems by using jobs (integrating with JP1/AJS)	
	Collecting and consolidating the audit log (Integrating with JP1/Audit Management - Manager)	
	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 (linked with cluster software)	

Category	Functionality	Reference manual#1
	Node switching system (integrating with cluster software) for host unit management model.	
Maintenance functionality	Troubleshooting related functions	*Maintenance and Migration Guide*#2
	Performance analysis using the trace based performance analysis	
	JavaVM functionality of the product (hereafter abbreviated as JavaVM)	
Migration functionality	Migrating from an older version of Application Server	
	Migration to recommended functions	
Compatibility functionality	Functionality for compatibility with the basic development functionality	*Compatibility Guide*
	Functionality for compatibility with the extended functionality	

#1

uCosminexus Application Server has been omitted from the manual names.

#2

This manual.

1.2 Functionality corresponding to the purpose of the system

On Application Server, you must choose the applicable functionality according to the purpose of the system to be built and operated.

This section describes the cases in which the following functionality, described in this manual, can be used. The functionality is independently extended on Application Server:

- Functionality for system maintenance
- Functionality of JavaVM
- Functionality for migrating from old version products

The functionality-wise support for the following items are described here:

- **Performance**
  This functionality is best used with a system that adds value to performance.
  This functionality used for performance tuning of the system is included.
- **Operation and maintenance**
  This functionality is best used when efficient operation and maintenance is to be performed.
- **Others**
  This functionality is used for complying with other individual purposes.

1.2.1 Functionality for system maintenance

The following table lists the functionality for system maintenance. Select the functionality according to the system purpose. For details on the functionality, see the *Reference* column in the following table.

Table 1–2: Functionality for system maintenance

Functionality	Purpose of the system			Reference
	Performance	Operation and maintenance	Others	
Troubleshooting	--	Y	--	*Chapter 2*, *Chapter 3*, *Chapter 4*, *Chapter 5*, *Chapter 6*
Performance analysis by using trace based performance analysis	Y	Y	--	*Chapter 7*, *Chapter 8*

Legend:
Y: Applicable
--: Not applicable

1.2.2 JavaVM functionality of the product

The following table lists the JavaVM functionality of the product. Select the functionality according to the system purpose. For details on the functionality, see the *Reference* column in the following table.

Table 1–3: JavaVM functionality of the product

Functionality	Purpose of the system			Reference
	Performance	Operation and maintenance	Others	
Class-wise statistical functionality	Y	Y	--	*9.3*
Instance statistical functionality	Y	Y	--	*9.4*
STATIC member statistical functionality	Y	Y	--	*9.5*
Reference-related information output functionality	Y	Y	--	*9.6*
Pre-statistical GC selection functionality	Y	Y	--	*9.7*
Unused objects statistical functionality for the Tenured area	Y	Y	--	*9.8*
Base object list output functionality for increase in the Tenured area	Y	Y	--	*9.9*
Class-wise statistical information analysis functionality	Y	Y	--	*9.10*
Tenuring distribution information output functionality of the Survivor area	Y	Y	--	*9.11*
`hndlwrap` functionality	--	Y	--	*9.12*
Functionality to set the upper limit of the allocation size of C heap during JIT compilation	--	Y	--	*9.13*
Functionality to set the upper limit of the number of threads	--	Y	--	*9.14*

Legend:

Y: Applicable

--: Not applicable

1.2.3 Functionality for migrating from products of earlier versions

The following table lists the functionality for migrating from the products of earlier versions. Select the functionality according to the system purpose. For details on the functionality, see the *Reference* column in the following table.

Table 1–4: Functionality for migrating from products of earlier versions

Functionality	Purpose of the system			Reference
	Performance	Operation and maintenance	Others	
Migrating from Application Server of earlier versions	--	--	Y	*Chapter 10*

Legend:

Y: Applicable

--: Not applicable

1.3 Description of the functionality described in this manual

This section describes the meaning of the classifications used when describing the functionality in this manual, and also provides an example of the tables used for describing each classification.

1.3.1 Meaning of classifications

The description of functionality in this manual is classified into the following five categories. You can select and read the required location depending on the purpose for referencing this manual.

- Description

  This is the description about the functionality. This section describes the purpose, features, and mechanism of the functionality. Read this section when you want an overview of the functionality.
- Implementation

  This section describes the methods such as the coding method and the DD writing method. Read this section when developing applications.
- Settings

  This section describes the required property settings for building systems. Read this section when building a system.
- Operations

  This section describes the operation method. This section describes the operating procedures and the execution examples of commands to be used. Read this section when operating a system.
- Notes

  This section describes the general precautions for using the functionality. Make sure that you read the notes.

1.3.2 Example of tables describing classifications

Tables are used to describe the *classifications of the description of functionality*. The title of each table is either "Organization of this chapter" or "Organization of this section".

The following is an example table describing the classification for the description of functionality.

Example of table describing the classification of functionality description

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

Category	Title	Reference
Description	What is the *XX* functionality	*X.1*
Execution	Execution of applications	*X.2*
	Definitions in DD and cosminexus.xml#	*X.3*
Settings	Settings in the execution environment	*X.4*
Operations	Operations using the *XX* functionality	*X.5*
Notes	Precautions when using the *XX* functionality	*X.6*

#

For `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 applications that do not contain `cosminexus.xml`, you set or change the properties after importing the properties into the execution environment. You can also change the set properties in the execution environment.

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

The tags specified in the property file correspond to the DD or `cosminexus.xml`. For details on the DD or `cosminexus.xml` and the property file tags, see *3. Property Files Used for Setting J2EE Applications* in the *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.

1.4 Main functionality changes in Application Server 11-00

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

The contents described in this section are as follows:

- This section gives an overview and describes the main changes in the functionality of Application Server 11-00. For details on the functionality, see the description in the section column of the reference. The description of a particular functionality is mentioned in the section column or that reference.
- *uCosminexus Application Server* is omitted from the manual names mentioned in the *Reference location* column.

1.4.1 Simplifying implementation and setup

The following table describes the items that are changed to simplify installation and setup.

Table 1–5: Changes made for simplifying implementation and setup

Item	Overview of changes	Reference manual	Reference
Supporting Windows Server in the development environment	Windows Server OSs have been added to the OSs supported by uCosminexus Developer, so application development environments can now be built in cloud environments.	--	--

1.4.2 Supporting standard and existing functionality

The following table describes the items that are changed to support the standard and existing functionality.

Table 1–6: Changes made for supporting the standard and existing functionality

Item	Overview of changes	Reference manual	Reference
Supporting Servlet 3.0 and 3.1	Asynchronous servlets of Servlet 3.0 and asynchronous I/O APIs of Servlet 3.1 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 NIO HTTP server function	Instead of the conventional redirector function and in-process HTTP server function, the NIO HTTP server function was added as an in-process HTTP server that supports non-blocking I/O processing of asynchronous servlets and WebSocket.	*Web Container Functionality Guide*	*Chapter 6*
Supporting JPA 2.1	JPA 2.1 is now supported and JPA providers that support JPA 2.1 can now be used.	*Common Container Functionality Guide*	*Chapter 5*
Supporting CDI 1.2	CDI 1.2 is now supported.	*Common Container Functionality Guide*	*Chapter 8*

Item	Overview of changes	Reference manual	Reference
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	A function to relay WebSocket communication from HTTP Server to a J2EE server was added.	*HTTP Server*	*4.15*

1.4.3 Maintaining and improving reliability

The following table describes the items that are changed for maintaining and improving reliability.

Table 1–7: Changes made for maintaining and improving reliability

Item	Overview of changes	Reference manual	Reference
Changing the encryption communication module	The module mod_ssl was adopted as the encryption communication module for HTTP Server.	*HTTP Server*	*Chapter 5*

1.4.4 Other purposes

The following table describes the items that are changed for other purposes.

Table 1–8: Changes due to other purposes

Item	Overview of changes	Reference manual	Reference
Adding V9 compatibility mode	V9 compatibility mode was added for users migrating from J2EE server version 9 or earlier to maintain compatibility with version 9.	This manual	*10.3.3*

2 Troubleshooting

The Application Server provides functionality to output various data that can be used for handling trouble that occurs during system operations. This chapter describes notes about the acquisition of troubleshooting data, the procedures for handling the data, and troubleshooting.

2.1 Organization of this chapter

This chapter gives an overview of the acquisition and the handling of data for troubleshooting.

If a failure occurs in the running system, it is necessary to collect the data for analysis purpose based on the type of the error that has occurred. The application server provides a function that outputs the log that can be used as data for maintaining the system.

The following table describes the organization of this chapter.

Table 2–1: Organization of this chapter (Troubleshooting (overview of data acquisition and handling))

Category	Title	Reference
Explanation	Overview of troubleshooting	*2.2*
	Acquiring the Data	*2.3*
	Types of Required Data	*2.4*
Operation	Troubleshooting and Recovery	*2.5*
Notes	Precautions Related to Troubleshooting	*2.6*

For details about the data acquisition and output settings for troubleshooting, methods to output the data individually, the output destination and contents of the data, and the troubleshooting procedure, reference the following respective chapters:

- Settings related to data acquisition and output
  3. Preparing for Troubleshooting
- Default output destination of data and methods to output data individually
  4. Output Destinations and Output Methods of Data Required for Troubleshooting
- Data Output Contents
  5. Problem Analysis
- Troubleshooting Procedure
  6. Troubleshooting Procedure

2.2 Overview of troubleshooting

In component software, when an error occurs, the status of the software when the error occurs is output to the log as troubleshooting information. Collect and analyze this log and study the cause of error. You can even identify the location of error occurrence from the processing status of the request.

In a system built with the application server, you can collect the log and definition file together in the case of an error in the component software. This information is called *snapshot log*. The batch collection of a snapshot log is executed automatically, just before the logical server is terminated or the J2EE servers are restarted manually in a batch, in the management domain.

Moreover, when you use CTM, you can specify settings in the J2EE server so that when the J2EE server is terminated abnormally, the error is not returned to the client immediately. CTM locks and controls the scheduling queue of the J2EE application until the J2EE server is restarted. CTM maintains the registered requests and continues to accept requests from clients. As a result, if you restart the J2EE server immediately, you can continue with the operations without causing any trouble that the client might notice. Note that you can use CTM for the component software only with the product including Cosminexus Component Transaction Monitor. For details about the products that you can use, see *2.2.2 Component software functional guide* in the manual *uCosminexus Application Server Overview.*

The following subsections describe the flow of data acquisition and handling when a trouble occurs.

2.2.1 Overview of Troubleshooting

This subsection describes the procedure for dealing with problems that may occur during operation of an application server system.

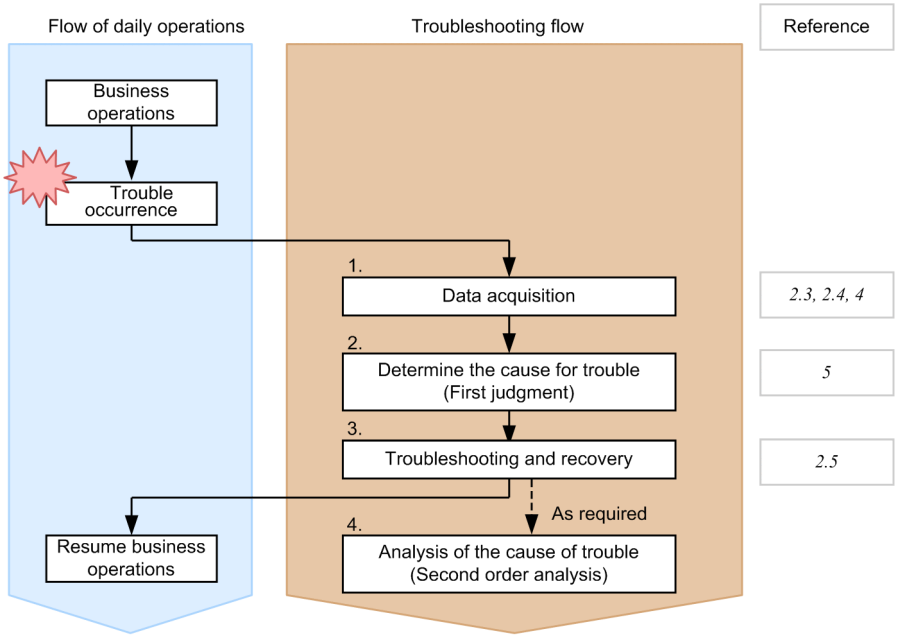
Note that among the information to be acquired when a problem occurs, preparation before starting operation is required to acquire the information below.

- JavaVM log (JavaVM log file)
  The JavaVM GC log is also output to this file.
- User dump (in Windows)
- core dump (in UNIX)
- OS statistical information (in Windows)

Perform preparations to acquire the information as and when required during system configuration. For details, see *3.2 Overview of data acquisition settings* and *3.3 Execution environment settings*.

To deal with problems that occur during system operation, follow the procedure in the figure below. The following figure shows the flow of the process for dealing with the occurred problem.

Figure 2–1: Flow for handling of data when a trouble occurs



1. Acquiring data

   Acquire all the required data for troubleshooting.

   For details about the data to be acquired, see *2.4 Types of Required Data*. Note that a memory dump is required only when restarting a J2EE Server or CORBA Naming Service.

   Moreover, for details about methods of data acquisition, see *2.3 Acquiring the Data*, or *4. Output Destinations and Output Methods of Data Required for Troubleshooting*.

   Reference note

   - You can acquire the data either collectively as a snapshot log or acquire individual logs.
   - You can acquire snapshot logs using the management command and with the automatic acquisition as per the settings.
   - You can automate the process of acquiring required data when an error occurs in the logical server. For details, see *2.3.1 Data That Can Be Acquired Automatically When a Problem Occurs*.

2. Identify the cause of the problem (Primary distinction)

   Identify the cause of the problem. Analyze the acquired data to identify the problem. For details about how to investigate data, see *5. Problem Analysis*.

3. Troubleshooting and recovery

   Troubleshoot the cause of the problem and perform recovery.

   For details about dealing with each type of problem, see *2.5 Troubleshooting and Recovery*.

4. Analyzing the cause of the problem (Secondary analysis)

   Analyze the cause of a trouble, as and when required. Request that maintenance personnel further analyze the cause of the problem.

Tip

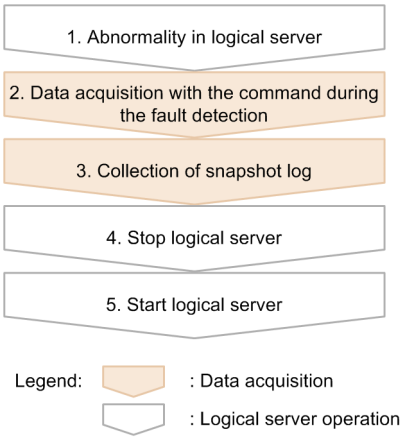
For details on the troubleshooting specific to the time at which a problem occurs, see *6. Troubleshooting Procedure*.

2.2.2 Flow of data acquisition when a trouble occurs

You can acquire the data of troubleshooting automatically in the system built on the Application Server. When you start the logical server, the Administration Agent starts monitoring the logical server. If a failure occurs in the logical server, the Administration Agent detects the failure and notifies it to the Management Server. The Management Server gets and collects the log snapshot data, and stops and restarts the logical server.

The following figure shows the flow when the data is obtained automatically.

Figure 2–2: Flow for automatic data acquisition



Using commands that are executed when an error is detected, mentioned in step 2, the information required for troubleshooting is output. Collect the information output by these commands and any other information that is required for troubleshooting in a compiled form as a snapshot log in step 3. Now, the snapshot log can also be collected without using the failure detection command, after the logical server is stopped in step 4. of the figure, but the information collected this time is that of J2EE server only. Hitachi, therefore, recommends that you use the failure detection command for collecting the snapshot log in step 3 of the figure. For details about the data acquisition using failure detection commands, see *2.3.2 Collecting the Material Using Commands during Error Detection* and about snapshot log, see *2.3.3 Collecting the Snapshot Log*.

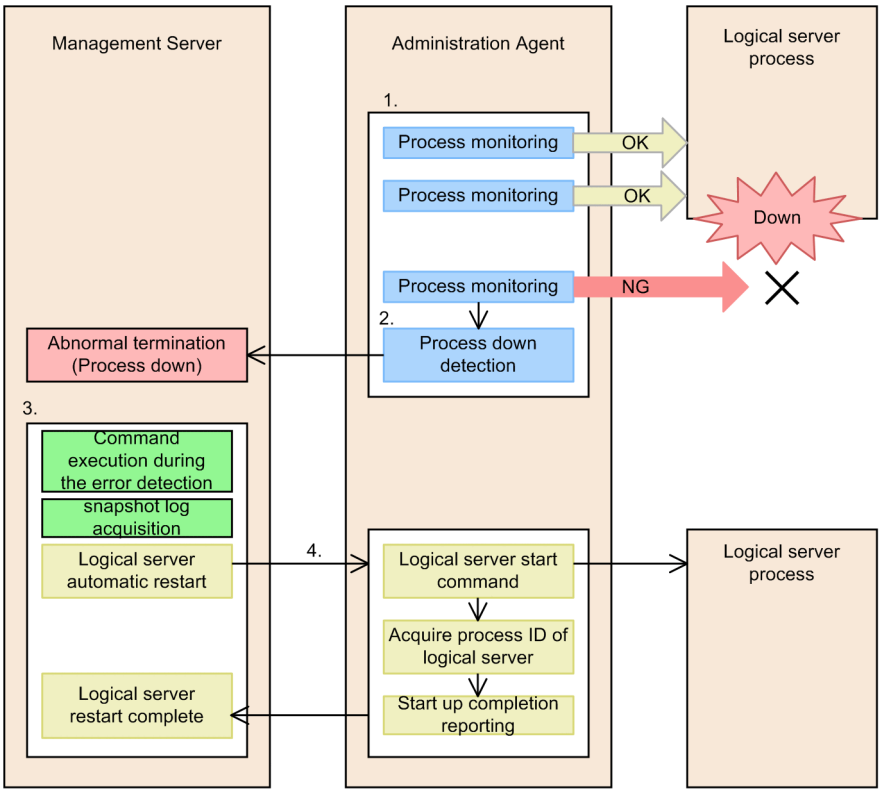
You can also use the management command (mngsvrutil) of the Management Server to collect the snapshot log at any time. For details about collecting snapshot logs using the management command, see *2.3.3(4) Collecting snapshot logs using the management commands*.

The processing flow when the process of the logical server is down or when the process of the logical server has hung up is as follows:

(1) The processing flow when the process of logical server is down

After the logical server is started, the process monitoring of the Administration Agent periodically monitors the process by using the process ID of the logical server process. The following figure shows the processing flow when the logical server process is down.

Figure 2–3: The processing flow when the process of logical server is down



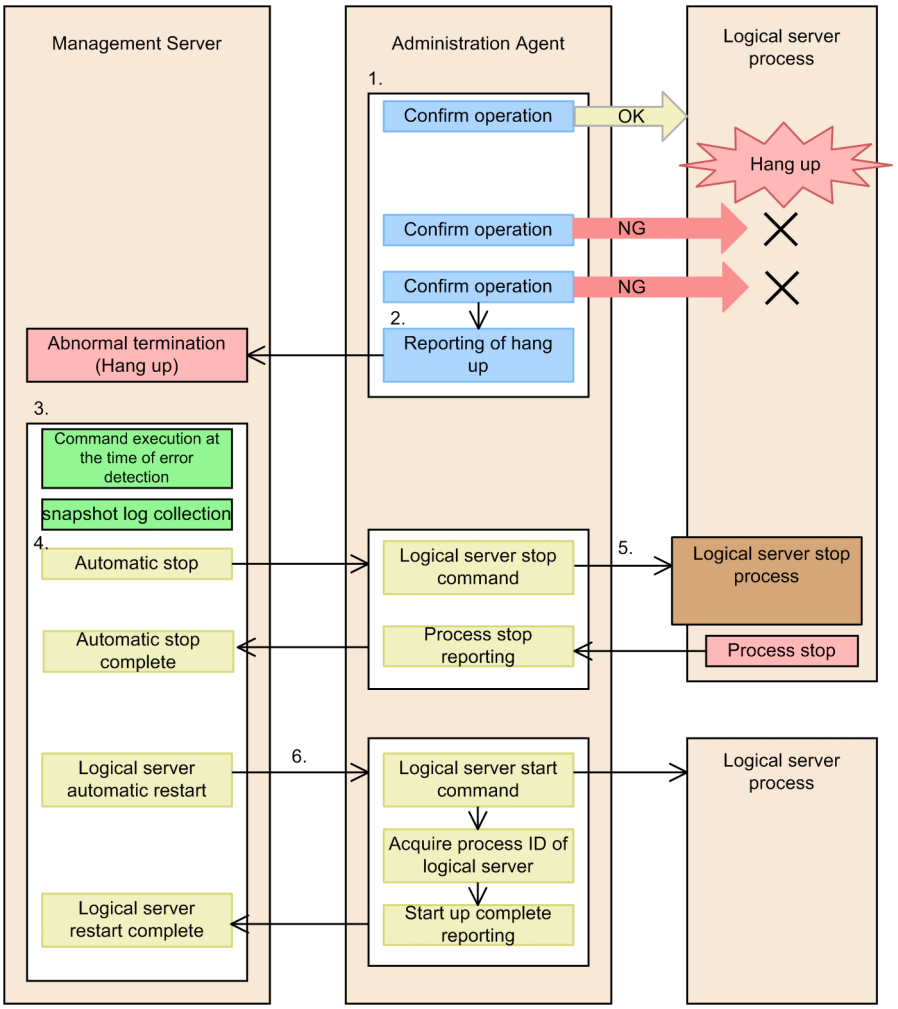
1. The process is monitored periodically by using the process ID of the logical server process.
2. If the logical server process ends abnormally, the Administration Agent detects that the process is down, and notifies it to the Management Server.

   The existence of the process ID is checked during the monitoring of processes. The contents of this check differ according to the type of the logical server. For details, see *2.3.1 Starting the Logical Server and Checking the Operations* in the *uCosminexus Application Server Operation, Monitoring, and Linkage Guide and 2.3.2 Stopping the Logical Server* in the *uCosminexus Application Server Operation, Monitoring, and Linkage Guide.*
3. If the Management Server finds that the process is down, the failure detection command is executed and the snapshot log is collected.
4. The logical server restarts automatically after the execution of the failure detection command and the collection of snapshot log.

(2) Processing flow when the logical server process hangs

After the logical server starts, the process monitoring of the Administration Agent periodically checks for the logical server process that the logical server is running. The following figure shows the processing flow when the logical server process hangs up while checking the operation.

Figure 2–4: Processing flow when the logical server process hangs up



1. Check the operation of the logical server process periodically.

   The operation is checked after the existence of the process ID is checked by the process monitoring. The contents of this check differ according to the type of the logical server.

   For details, see *2.3.1 Starting the Logical Server and Checking the Operations* in the *uCosminexus Application Server Operation, Monitoring, and Linkage Guide and 2.3.2 Stopping the Logical Server* in the *uCosminexus Application Server Operation, Monitoring, and Linkage Guide.*

2. If the operation check fails twice consecutively (default value), the Administration Agent detects that the process has hung up and notifies it to the Management Server. You can change the number of failed attempts of the operation check before a process is determined as hung up.

3. If the Management Server detects that the process has hung up, it executes the failure detection command and collects the snapshot log.

4. The automatic stop process is executed because the process is still running when it is detected as hung up.

5. In the Administration Agent, execute the stop command of the logical server.

   The force stop command is executed if the logical server does not stop even after a certain time period has elapsed.

6. The logical server restarts automatically after the execution of the failure detection command and the collection of snapshot log.

2.3 Acquiring the Data

You can acquire the data required for troubleshooting with any of the following methods:

- Acquire automatically when a problem occurs
- Acquire in a batch using the management command
- Collect each information separately

The information required for troubleshooting can be collected by compiling it together as a snapshot log. The status, when a problem occurs in the component software in a system, is output to the *snapshot log*. In a system built with the Application Server, you can collect the snapshot log for each type of component software collectively and output as a log file in a ZIP format. Furthermore, the snapshot log can be acquired by dividing into primary delivery data and secondary delivery data. You can also use the `snapshotlog` command to collect the definition sending data.

This section describes the snapshot logs for data acquisition in a batch and the failure detection commands executed for data acquisition, as methods for data acquisition when a trouble occurs.

For details about the information that is not acquired as a snapshot log, see *2.4.3 Correspondence Between Acquisition Methods and Investigation Methods* and for details about the methods for acquiring the respective information individually, see *4. Output Destinations and Output Methods of Data Required for Troubleshooting*.

Note that you need to specify the settings in advance depending upon the data to be acquired. For example, the data such as OS statistics or user dump will not be obtained unless you specify the settings for acquiring this data when building the system. Hitachi recommends that you obtain this data because it is required for troubleshooting.

You can change the default settings to obtain the data such as the output destination of the log file, number of log files, and the maximum size of each file. Change the settings for acquiring data as per your requirements.

The data that cannot be collected as snapshot log as per the default settings, can be collected collectively in the form of a snapshot log by defining the get destination of the data as the target of snapshot log collection.

Important note

Notes on data collection settings in Windows

With Windows, you must specify the settings in the registry to obtain data for investigation when an error occurs. The registry settings affect the whole system, so take adequate precautions when you specify the settings.

For details about the data acquisition settings required for troubleshooting, see *3. Preparing for Troubleshooting*.

2.3.1 Data That Can Be Acquired Automatically When a Problem Occurs

In a system built with the application server, when a problem occurs in the logical server, you can automatically acquire and compile the information required for troubleshooting.

Tip

For automatic data acquisition when a trouble occurs, you must specify the settings required while building systems. For details about data acquisition settings for troubleshooting, see *3.2 Overview of data acquisition settings*, and *3.3 Execution environment settings*.

Except for the following information, you can make the specification so that you can automatically collect all the other information required for troubleshooting using the commands executed during error detection and by collecting the snapshot log.

- OS statistical information (in Windows)

Use the following settings to collect the information that is not collected with default settings:

1. You use the commands created by the user to be executed when an error is detected to output the required information.
2. Add the output destination of the information output in 1. to the path of the snapshot log to be collected.
   For details about the settings for the snapshot log acquisition, see *3.3.3 Settings for collecting snapshot logs (Systems for executing J2EE applications)*, or *3.3.4 Settings for collecting snapshot log (Systems for executing batch applications)*.

Important note

Note that among the logs of the component software that can be acquired in the snapshot log, the acquisition of the logs shown in the following table is not explained in this manual. For details about methods to acquire these logs, see Reference manual of the following table.

Table 2–2: Component software for which the method of acquiring the logs is not mentioned in this manual

Log type	Reference	Section
SOAP application execution infrastructure log of Cosminexus Component Container	*uCosminexus Application Server SOAP Application Development Guide*	*Chapter 14*
Cosminexus Web Services - Security log	*uCosminexus Application Server Web Service Security Users Guide*	*Chapter 6*
Cosminexus TPBroker log	*TPBroker User's Guide*	

2.3.2 Collecting the Material Using Commands during Error Detection

The *failure detection command* is executed by the system when the Management Server detects the logical server failure. To execute the error detection command, specify the values in the keys of `mserver.properties` (Management Server environment configuration file) during system setup.

Important note

If a logical server process terminates abnormally when Administration Agent is not running, the error detection command is not executed even if you start Administration Agent later. In such cases, when Administration Agent

starts, the KEOS20071-E message with details "The command is not defined." is output. For details on the message, see the manual *uCosminexus Application Server Messages*.

You can use the failure detection command to execute the process for acquiring the thread dump and the user dump when a failure occurs, and obtain the error data quickly. Moreover, if you use the failure detection command, you can also automatically collect the snapshot log at any of the following timing. Decide the acquisition time according to the settings.

- If a failure occurs in the logical server, the failure detection command is executed and the snapshot log is collected before the logical server is stopped.
- If a failure occurs in the J2EE server or batch server, the failure detection command is executed and the snapshot log is collected before the J2EE server or batch server is restarted.

There are two types of failure detection commands; the failure detection command provided by the system and the failure detection command created by the user.

Failure detection command provided by the system

This command is already defined in the application server. When a failure occurs in the logical server, the failure detection command provided by the system obtains the information such as the thread dump and the trace based performance analysis of JavaVM of the logical server in which the failure occurred. The data obtained by using the failure detection command provided by the system can be collected as the snapshot log.

According to the default settings, the failure detection command provided by the system is executed when a failure occurs in the logical server and the snapshot log is collected before the logical server in which the error occurred is stopped.

Failure detection command created by the user

A failure detection command can also be created by the user. A batch file or shell script, in which the required process is described by the user for acquiring the data, can be executed as the failure detection command.

To collect the data obtained by using the failure detection command created by the user, as the snapshot log, you need to pre-define the get destination of the data as the collection destination of the snapshot log.

For details about changing the operation settings of commands provided by the system during failure detection or for settings required for commands created by the user during failure detection, see *3.3.1 Data acquisition settings using failure detection time commands (Systems for executing J2EE applications)* or *3.3.2 Data acquisition settings using failure detection time commands (Systems for executing batch applications)*.

Tip

If the necessary settings are not performed when configuring a system, commands during error detection are not executed. Confirm the following settings:

- When true is set in the `com.cosminexus.mngsvr.sys_cmd.abnormal_end.enabled` key of `mserver.properties` (Management Server environment settings file), commands during error detection provided by the system are executed.
- When true is set in the `com.cosminexus.mngsvr.usr_cmd.abnormal_end.enabled` key of `mserver.properties` (Management Server environment settings file), commands created by the user are executed when an error is detected.

The information that can be acquired by each type of the above-mentioned command is described below.

(1) Information that can be acquired by executing the failure detection time commands provided by the system

The following table describes information that can be acquired by executing the commands during error detection provided by the system. The information that can be acquired by executing the commands during error detection differs, depending on the type of error (system down or hang-up) and the OS used.

Note that it is necessary to perform the required settings in adminagent.properties (Administration agent property file) to acquire this information in the snapshot log.

The following table describes the information that can be acquired by executing the commands during error detection.

Table 2–3: Information that can be acquired by command during failure detection

Logical server	OS	Information that you can acquire		J2EE Application	Batch Application
		When process down is detected	When hang up is detected		
Logical performance tracer	Windows UNIX	• PRF trace file where the buffer contents are output	--	Y	Y
Logical J2EE server#	Windows	• Trace based performance analysis file	• Thread dump • Trace based performance analysis file	Y	Y
	UNIX	• Stack trace information of JavaVM • Trace based performance analysis file	• Thread dump • Trace based performance analysis file	Y	Y
Logical Web server	Windows UNIX	• Trace based performance analysis file	• Internal trace • Trace based performance analysis file	Y	--
Other logical servers	Windows UNIX	• Trace based performance analysis file	• Trace based performance analysis file	Y	--

Legend:

J2EE application: System for executing a J2EE application.

Batch application: System for executing a batch application.

--: Not applicable.

Y: Can be acquired.

#

Define a batch server as a logical J2EE server.

- **Thread dump of a J2EE server or a batch server where the problem occurred**

  You can acquire a Thread dump when true is set in the `adminagent.j2ee.sys_cmd.abnormal_end.threaddump` key of `adminagent.properties` (Administration agent property file).

- **Trace based performance analysis**

  You can acquire the trace based performance analysis when true is set in the `adminagent.sys_cmd.abnormal_end.prftrace` key of `adminagent.properties` (Administration agent property file). The trace based performance analysis file is output to *Application-Server-installation-directory*`/manager/tmp`. Note that you can change the output destination with the `adminagent.prftrace_dir` key in `adminagent.properties` (Administration Agent property file).

Important note

A Thread dump is not output when it is determined that the J2EE server process where the problem occurred does not exist. Moreover, the trace based performance analysis is not output when it is determined that the performance tracer process does not exist.

(2) Information that can be acquired by executing the commands during error detection created by the user

You can execute batch files or shell scripts in which any process required for acquiring the data is coded, as the user created commands during error detection. For example, you can acquire a user dump or core dump by executing a command such as the `drwtsn32` command in this command. For details about how to create failure detection time commands, see *3.3.1 Data acquisition settings using failure detection time commands (Systems for executing J2EE applications)* or *3.3.2 Data acquisition settings using failure detection time commands (Systems for executing batch applications)*.

Note that the command during error detection created by the user is executed if the `adminagent.`*serverkind*`.usr_cmd.abnormal_end` key of `adminagent.properties` (Administration agent property file) is specified.

2.3.3 Collecting the Snapshot Log

The log in which the status of the system when failure occurred is output by the component software of the system is called the *snapshot log*. The snapshot log contains the thread dump, trace based performance analysis, information necessary for system maintenance, and the information necessary for application maintenance and so on, in addition to the logs of the different types of component software. In a system built with the application server, this information can be collected together in the form of snapshot log and obtained as a log file in the ZIP format. The administrator can counter the errors by collecting and analyzing the snapshot log.

The settings are specified in such a way that the data necessary for troubleshooting is automatically collected by executing the failure detection command and collecting the snapshot log. To get the information not collected as per the default settings, in the form of the snapshot log, define the get destination of the data as the collection target of snapshot log. For details on the information that is not collected by default, see the description on the collectability of the snapshot log and the changing of collection-related settings in *Appendix A. List of Snapshot Logs to Be Collected*.

You can change the settings for the target of snapshot log collection, output destination directory, and the number of files, when building the system.

This subsection describes about the timing for collecting snapshot logs, the data that you can collect, and the flow of collection. Also, this subsection describes about collecting snapshot logs by using management commands as a way of collecting the snapshot logs at any time. Note that for the settings of snapshot log collection, see *3.3.3 Settings for collecting snapshot logs (Systems for executing J2EE applications)* or *3.3.4 Settings for collecting snapshot log (Systems for executing batch applications)*.

(1) Timing for collecting snapshot logs

You can collect snapshot logs automatically or at a specific timing. The following table describes the timing for the collection of the snapshot log. Note that you can collect the snapshot logs in the host on which a logical server is running.

Table 2–4: Timing for collecting the snapshot log

Category	Collection timing
Collect automatically[#1]	Immediately before the logical server stops automatically due to an error[#2]
	Just before automatically restarting the J2EE server or the batch server managed in the management domain[#2]
	Immediately before the J2EE server or batch server is restarted manually in a batch
Collect at the specified timing	When the management command (`mngsvrutil`) of the Management Server is executed to collect the snapshot log(For the execution method, see *(4) Collecting snapshot logs using the management commands*)

#1

When building a system you can change the settings for collecting snapshot logs at any of the following timings According to the default settings, the snapshot log is collected when the logical server stops.

- Before terminating the logical server
- Before restarting the J2EE server

#2

The Management Server executes the failure detection command for collecting the snapshot log at this time.

Reference note

You can collect the snapshot log at following timings:

- When **Collect** button in Snapshot Log window in Start/Stop Logical Server of Management Portal is clicked[#1]
- Executing the `Snapshotlog` command [#2]

#1

You can collect snapshot logs on a J2EE server or a batch server on which the collection procedure is executed.

#2

You can execute the command irrespective of whether Management Server is used.

For the snapshot log window, see *11.9.5 Collecting snapshot log of a J2EE server* in the *uCosminexus Application Server Management Portal User Guide*. For the `snapshotlog` command, see *snapshotlog (collect snapshot logs)* in the *uCosminexus Application Server Command Reference Guide*.

(2) Data that can be collected as snapshot log

This section describes the data that you can collect as a snapshot log. The snapshot log includes the information required for system maintenance and application maintenance.

The data required for troubleshooting is classified as primary delivery data and secondary delivery data depending on the timing at which the data is delivered to the maintenance service. You can collect the primary delivery data and the secondary delivery data in snapshot logs. You can also use the `snapshotlog` command to collect the definition sending data.

- Primary delivery data

  Material that is of a file size that can be sent as an attachment to mail. You can send the data immediately to the maintenance service by using mail.

- Secondary delivery data

  In addition to the primary delivery data, comparatively large size files are applicable. You must send large files separately to the maintenance service, since sending of large files takes more time than usual.

- Definition sending data

  The data that collects only the definition files to investigate the errors in the definition file settings. The file is small and can be sent quickly to the maintenance personnel through a mail. Note that you can also collect this data by using the `snapshotlog` command even if Management Server and the J2EE server are not running.

  Important note

  The definition information collected in the definition sending data might include the user ID and password.

The data to be collected as snapshot log can be changed with the settings of the snapshot log collection target definition files. For example, if you want to collect the information output by a command during error detection created by the user and if the settings are changed, such information will be added to the information to be collected. The snapshot log collection target definition file is set during system configuration.

You can confirm which information will be actually collected in the real environment from the following files:

- snapshotlog.conf

  The contents to be acquired as the primary delivery data are defined in this file.

  The location for storing `snapshotlog.conf` is as follows:

  - In Windows

    Application-Server-installation-directory`\manager\config\snapshotlog.conf`
  - In UNIX

    `/opt/Cosminexus/manager/config/snapshotlog.conf`

- snapshotlog.2.conf

  The contents that are to be acquired as the secondary delivery data are defined in this file.

  The location for storing `snapshotlog.2.conf` is as follows:

  - In Windows

    Application-Server-installation-directory`\manager\config\snapshotlog.2.conf`
  - In UNIX

    `/opt/Cosminexus/manager/config/snapshotlog.2.conf`

- **snapshotlog.param.conf**

  This file defines the content to be obtained as the definition sending data.

  The location for storing `snapshotlog.param.conf` is as follows:

  - In Windows

    Application-Server-installation-directory`\manager\config\snapshotlog.param.conf`
  - In UNIX

    `/opt/Cosminexus/manager/config/snapshotlog.param.conf`

However, depending on how the file is specified in `adminagent.properties` (administration agent property file), there are cases when the information that can be collected differs.

For default settings, when a problem occurs the snapshot log acquired automatically includes the following information:

Primary delivery data

- Information under the directory defined to collect the primary delivery data for the snapshot log[#1]
- Thread dump of a J2EE server[#2]
- Installation information
- OS status log

Secondary delivery data

- Trace based performance analysis[#2]
- Information under the directory defined to collect the secondary delivery data for the snapshot log[#1]
- Installation information
- OS status log

#1

For details about the snapshot logs to be collected, see *Appendix A. List of Snapshot Logs to Be Collected*.

#2

You can collect this information when the settings are set to output a thread dump and trace based performance analysis by using the system provided commands that are to be executed when an error is detected.

Reference note

Files that can be collected in the default state differ between the snapshot log that can be collected by using the `snapshotlog` command and the snapshot log that can be collected by using the management command (`mngsvrutil`) of the Management Server or other methods. The snapshot log that can be collected by using the management command (`mngsvrutil`) of the Management Server or other methods can collect installation information as well as OS status and logs in the default state. However, to collect such information by using the `snapshotlog` command, you need to edit the snapshot log collection target definition file specified when the command is executed and add the snapshot log collection destination.

An example of the information that can be collected by default and information that cannot be collected by default is as follows:

(a) Information that can be collected by default

An example of the information that can be collected by default is as follows:

Information required for application maintenance

The message log and user log of Cosminexus Component Container are collected as the information required for application maintenance.

- Cosminexus Component Container[#]
- Cosminexus Component Transaction Monitor
- Cosminexus Developer's Kit for Java
- Cosminexus Performance Tracer
- Cosminexus TPBroker
- Cosminexus Web Services - Security
- Cosminexus HTTP Server

Other than the above, Hitachi Trace Common Library log and program product information (in UNIX) are also collected.

Also includes the SOAP application execution infrastructure information.

Information required for application maintenance

The message log and user log of Cosminexus Component Container are collected as the information required for application maintenance.

The following information is also collected by default:

Component Container-related information

In Windows

All the items beneath *Application-Server-installation-directory*`\CC\server\public`

In UNIX

All the items beneath `/opt/Cosminexus/CC/server/public`

Note that the default directories created when installing the component software are defined as the collection target for the snapshot log by default. Always change the collection destination when changing the log output destination.

(b) Information that cannot be collected by default

The information described below cannot be collected when operating with default settings. Perform the settings in such a way so that you can collect the information as and when required.

In Windows

Cosminexus Component Container

- EAR/JAR file (if it cannot be deployed or imported)

Microsoft IIS

The following data is acquired when integrated with Microsoft IIS.

- `C:\inetpub\logs` (Specify the system drive in place of C:)

OS

- Set of the data related to the system monitor (see subsection *4.13*)
- Event log (application, system)
- OS operation data

  From the winmsd start-up window, extract the data from the Operation-Save As Text File menu (It may take 5 to 10 minutes).

In UNIX

Cosminexus Component Container

- EAR/JAR file (if it cannot be deployed or imported)

OS

- OS operation data

(3) Flow of snapshot log collection

If trouble occurs on the logical server, a snapshot log collection service request is executed from Management Server for Administration Agent. The procedure of the process that is executed during snapshot log collection is as follows. Note that the executed process differs according to the settings.

1. Executing the failure detection command provided by the system
2. Executing the failure detection command created by a user
3. Collecting snapshot logs (primary delivery data)
4. Collecting snapshot logs (secondary delivery data)

Collection of snapshot logs might take more time than usual depending on the operating environment of the machine and status of failure occurrence (such as the status in which an unexpected error occurred). Therefore, you can set respective timeouts for these processes.

Note that you set a timeout for collection of snapshot logs in cases such as when excessive time is consumed for collecting snapshot logs due to the occurrence of an unexpected failure and the logical server is expected to be restarted on priority. If a timeout is set when collecting snapshot logs, the collection process is suspended forcefully and the information required for trouble shooting cannot be acquired.

(4) Collecting snapshot logs using the management commands

You can use the management command (`mngsvrutil`) to collect the snapshot log at any time, in addition to automatic collection.

Note that the type of information that can be acquired is the same as the snapshot log that is collected automatically in the case of problems. See *(2) Data that can be collected as snapshot log*.

For details about the log output destinations, see the value of `adminagent.snapshotlog.log_dir` key of `adminagent.properties` file (Administration Agent property file). The default output destination is the following directory.

- In Windows
  Log-output-directory-of-Manager`\snapshot`
- In UNIX
  Log-output-directory-of-Manager`/snapshot`

For collecting the snapshot log, specify the subcommand `collect` in the management command `mngsvrutil` and then execute. You can collect the snapshot log from the host specified with `-t` option.

The execution format and an example of executing the command are described below. In this example, collect both the primary delivery file and the secondary delivery file. In n, specify whether to collect the primary delivery data or secondary delivery data.

Execution format

```
mngsvrutil -m Management-Server-host-name[:Port number]-u Management-user
-ID -p Management-password -t host-name -k host collect snapshot n
```

Execution example

```
mngsvrutil -m mnghost -u user01 -p pw1 -t host01 -k host collect snapsho
t 1
mngsvrutil -m mnghost -u user01 -p pw1 -t host01 -k host collect snapsho
t 2
```

2.3.4 Location to store the acquired information

The information acquired automatically by collecting the snapshot log is saved with the following names under the directory specified in the adminagent.snapshotlog.log_dir key of adminagent.properties (Administration agent property file).

- File collected as the primary delivery data

  snapshot-`host name`[#1]-log-*yyyyMMddHHmmss*[#2].zip
- File collected as the secondary delivery data

  snapshot-`host name`[#1]-log-*yyyyMMddHHmmss*[#2].2.zip

Note:

The definition sending data is not collected.

#1

When a logical server is specified and the snapshot log is collected, the name of that logical server will be the host name.

#2

yyyy: year MM: month dd: day HH: hour mm: minutes ss: seconds

Note that the default output directory is as follows:

- In Windows

  Log-output-directory-of-Manager`\snapshot`
- In UNIX

  Log-output-directory-of-Manager`/snapshot`

> **Reference note**
>
> The list of files to be collected as the snapshot log and the list of files actually collected can be confirmed from the following files. These files are in the archived zip file.
>
> - List of files to be collected
>
>   snapshot-*host-name*-log-*yyyyMMddHHmmss*[#]/filelist_pre.txt
>
>   snapshot-*host-name*-log-*yyyyMMddHHmmss*[#].2/filelist_pre.txt
> - List of files actually collected
>
>   snapshot-*host-name*-log-*yyyyMMddHHmmss*[#]/filelist_post.txt
>
>   snapshot-*host-name*-log-*yyyyMMddHHmmss*[#].2/filelist_post.txt
>
> # yyyy: year MM: month dd: day HH: hour mm: minutes ss: seconds

2.4 Types of Required Data

This section describes the type of a trouble that might occur in a system of the Application Server and the data required according to the trouble.

The troubleshooting data differs depending upon the type of failure that occurs. For example, the troubleshooting data differs depending upon whether an error message is output, or whether the server process of the application server ended abnormally.

Moreover, you need to contact the maintenance personnel depending upon the type of failure. *Maintenance personnel* refers to the person you need to contact as per the purchase contract.

Problems for which you need to contact maintenance personnel

- "Contact maintenance personnel" is mentioned in the action for the output error message
- The cause of the error is not known
- The action corresponding to the error message cannot be performed

When you contact maintenance personnel, report the following information as accurately as possible:

- Error occurrence time
- Operations performed just before the error occurred
- Window operations when the error occurred
- Data acquired when the error occurred

The troubleshooting data is classified into the primary data, secondary data, and definition sending data according to the time at which the data is sent to the maintenance personnel.

- Data to be sent immediately through e-mail (Primary delivery data and Definition sending data)

  Includes the data of the file size that you can attach and send by e-mail. It can be sent to the maintenance personnel quickly by e-mail.

  For example, the following data can be obtained in the form of primary delivery data:

  - Message log

    The status and the error information of the J2EE server, and so on, is output in the message log.
  - User log

    The information of the standard output and the standard error output in the application is output in the user log.
  - Exception log

    The exception information when the error occurred in the system is output in the exception log.
  - Maintenance log

    The failure maintenance information when the error occurred in the system is output in the maintenance log. The maintenance personnel use this information for analyzing the failure.

  Also, only the collected definition file information is output as the definition sending data. This information is used to investigate the definition errors.
- Data to be sent separately (Secondary delivery data)

  This is applicable to the data that you need to send separately without using mail because the file sizes are relatively large and it will take time to send the files to maintenance personnel. Note that secondary delivery data contains primary delivery data.

  For example, the following data can be obtained in the form of secondary delivery data.

- Trace based performance analysis

  The trace information that is output in one cycle of processing of the request is output in the trace based performance analysis.
- Memory dump

  The memory image of the process is output in the memory dump.

2.4.1 Trouble types and the required data

Types of problems that can occur in an application server system are as follows:

- Error message output

  Error messages are output.
- System down

  The server process of the application server terminates abnormally.
- Hang-up (No response)

  There is no response to an application call.
- Slow down

  There is a response to an application call but the response time is excessive.

The following table describes the data to be acquired for each type of problem.

Table 2–5: List of data to be acquired for each type of problem

Item No.	Type of material	Problem type			
		Error message output	System down	Hang-up (No response)	Slow down
1	Message log[#1]	Y	Y	Y	Y
2	User log[#1]	Y	Y	Y	Y
3	Exception log[#1]	Y	Y	Y	Y
4	Maintenance log[#1]	Y	Y	Y	Y
5	Trace based performance analysis	Y	Y	Y	Y
6	Thread dumps of JavaVM	Y	Y	Y	Y
7	GC logs of JavaVM	P	Y	Y	Y
8	Memory dump	--	Y	Y	Y
9	JavaVM log file[#2]	--	Y	Y	Y
10	Error report files	--	Y	--	--
11	OS state or log	Y	Y	Y	Y
12	Statistical information of OS	P	Y	Y	Y
13	Definition information	Y	Y	Y	Y
14	Operation directory[#4]	P	P	P	P
15	Resource setting	Y	Y	Y	Y

Item No.	Type of material	Problem type			
		Error message output	System down	Hang-up (No response)	Slow down
16	Web server log	Y	Y	Y	Y
17	JavaVM stack trace#3	--	Y	--	--
18	Event log of the Explicit Memory Management functionality of JavaVM#2	--	Y	Y	Y

Legend:

Y: Required

P: Whether to be acquired depends on the actual error contents

--: Do not acquire

#1

Log output by the Application Server component software.

#2

Output when JavaVM start option is specified. For details about acquisition options, see *4.10 JavaVM log (JavaVM log file)*. For details about options, see *14. Options for Invoking JavaVM* in the *uCosminexus Application Server Definition Reference Guide*.

#3

Output only when using UNIX.

#4

The default working directories are *Application-Server-installation-directory*`\CC\server\public` (in Windows), or `/opt/Cosminexus/CC/server/public` (in UNIX).

2.4.2 List of Required Data to Be Acquired

The following table describes a list of data required for troubleshooting in the application server. Moreover, the same table also describes the timing to send the data to maintenance personnel when required.

Table 2–6: List for data acquisition

Item No.	Type of material	Description	Delivery timing
1	Message log#1	The status and the error information of the J2EE server, and so on, is output in the message log.	Primary/ Secondary
2	User log#1	The information of the standard output and the standard error output in the application is output in the user log.	Primary
3	Exception log#1	The exception information when the error occurred in the system is output in the exception log.	Primary
4	Maintenance log#1	The failure maintenance information when the error occurred in the system is output in the maintenance log. The maintenance personnel use this information for analyzing the failure.	Primary/ Secondary
5	Trace based performance analysis	The trace information that is output in one cycle of processing of the request is output in the trace based performance analysis.	Secondary
6	Thread dumps of JavaVM	Outputs the JavaVM operation information and thread stack status.	Primary
7	GC logs of JavaVM	Outputs the activity status of JavaVM GC.	Primary
8	Memory dump	The memory image of the process is output in the memory dump.	Secondary

Item No.	Type of material	Description	Delivery timing
9	JavaVM log file[#2]	Outputs the JavaVM log provided with Cosminexus Developer Kit for Java.	Primary
10	Error report files	A log file output when you stop the JavaVM.	Primary
11	OS state or log	A record of the OS operation information.	Primary/ Secondary
12	Statistical information of OS	A record of the OS statistical information.	Primary
13	Definition information	Definition information of various types of the application server settings.	Primary/ Definition
14	Operation directory	A directory that stores the work file during the application server operation.	Primary/ Secondary/ Definition
15	Resource setting	The information of settings of resources used in the application server (data source, resource adapter).	Secondary/ Definition
16	Web server log	Log of Cosminexus HTTP Server, Microsoft IIS.	Primary/ Secondary
17	JavaVM stack trace[#3]	The information used for cause investigation if JavaVM terminates abnormally.	Primary/ Secondary
18	Event log of the Explicit Memory Management functionality of JavaVM[#2]	Information of an event (initialization of Explicit memory block, object creation to Explicit memory block, release process of Explicit memory block) occurred due to the Explicit Memory Management functionality will be output.	Primary

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

#1

Log output by the Application Server component software.

#2

Output when JavaVM start option is specified. For details about acquisition options, see *4.10 JavaVM log (JavaVM log file)*. For details about options, see *14. Options for Invoking JavaVM* in the *uCosminexus Application Server Definition Reference Guide*.

#3

Output only when using UNIX.

A log file is created automatically in the directory specified in the environment settings file of each component software. A serial number from 1 to the number of specified log files is attached to the file name. When the size of the log file reaches its maximum size for one file, the output destination switches to the next log file. After the number of specified log files are output, the log is output once again to the log file with the number of the file name as 1. Note that there are times when the file size of the log file may slightly increase as compared to the size specified in the environment settings file to ensure that one record of a log file is output without a break. The maximum length of one record is about 4 KB.

2.4.3 Correspondence Between Acquisition Methods and Investigation Methods

This section describes the reference destination of methods for acquiring and investigating the required data to be acquired. Moreover, this section also describes the types of information that can be acquired as a snapshot log.

Table 2–7: Reference destination of methods for acquiring and investigating the required data to be acquired

Item No.	Type of material	Possibility of acquiring in snapshot log with default settings[#1]	Reference	
			Individual collection method	Investigation method
1	Message log	Y	*4.3*, *4.4*, *4.5*	*5.2*, *5.3*
2	User log	Y		
3	Exception log	Y		
4	Maintenance log	Y		
5	Trace based performance analysis	Y	*4.6*	*5.4*, *7.7*
6	JavaVM thread dump	Y[#2]	*4.7*	*5.5*
7	GC logs of JavaVM	--	*4.8*	*5.6*
8	Memory dump	--[#2]	*4.9*	--[#3]
9	JavaVM log file[#4]	--	*4.10*	*5.7*
10	Error report files	--	*4.11*	*5.8*
11	OS status and log	Y	*4.12*	*5.9*
12	OS statistical information	--	*4.13*	--[#3]
13	Definition information	Y	*4.14*	--[#3]
14	Working directory	--	*4.15*	--[#3]
15	Resource setting	--	*4.16*	--[#3]
16	Web server log	--	*4.17*	--[#3]
17	JavaVM stack trace[#3, #5]	--	*4.18*	*5.10*
18	Event log of the Explicit Memory Management functionality of JavaVM[#4]	--	*4.19*	*5.11*

Legend:

Y: Can be acquired

--: Cannot be acquired

#1

This log can be acquired if the directory, in which snapshot log is collected, is added.

#2

This log can be collected only if it is output using commands when an error is detected in advance. (When collecting the snapshot log, the commands that output a dump cannot be executed).

#3

Maintenance personnel use this data.

#4

Output when JavaVM start option is specified. For details about acquisition options, see *4.10 JavaVM log (JavaVM log file)*. For details about options, see *14. Options for Invoking JavaVM* in the *uCosminexus Application Server Definition Reference Guide*.

#5

Output only when using UNIX.

2.5 Troubleshooting and Recovery

This section describes how to troubleshoot and recover when following errors occur:

- If the configuration software process (logical server) terminates abnormally
- If forced termination of a J2EE application fails
- If a problem occurs when using the database session failover function
- If JavaVM terminates abnormally
- If the Administration Agent is terminated forcibly, when `OutOfMemoryError` occurs
- If a problem occurs in the system linked with JP1
- If a problem occurs in 1-to-1 node switching systems
- If a problem occurs in N-to-1 recovery systems
- If a problem occurs in the node switching system for the host unit management model
- If a problem occurs in the EJB client

2.5.1 If the Configuration Software Process (Logical Server) Terminates Abnormally

The method for restarting the system, when Application Server configuration software process terminates abnormally, is described here.

In a system build on Application Server, if the start order is set up when the component software process (logical server) of Application Server terminates abnormally, the Management Server will automatically restart the component software. In the Management Server, the processing of component software is managed as *Logical Server*.

When restarting manually, if the abnormally terminated configuration software has startup dependencies, confirm that the configuration software, on which the abnormally terminated configuration software depends for starting, is running. The configuration software, on which the abnormally terminated configuration software depends, varies depending on the configuration of the system. The startup dependencies of the configuration software and the method of restarting the configuration software are explained below:

Tip

In a system using CTM, if component software is restarted immediately by CTM, you can specify settings in such a way so that the system is recovered before the error is returned to the client. However, when the number of maximum registration of the request queue exceeds, an error will return to the client.

For details on a system using CTM, see *3. Scheduling and Load Balancing of Requests Using CTM* in the *uCosminexus Application Server Expansion Guide*.

(1) Configuration software startup dependencies

The startup dependencies of the configuration software process configuring a system built with Application Server are explained here.

Hereafter, the configuration software process startup dependencies are explained:

(a) Dependencies of the process

The dependencies of the process are as follows:

Table 2–8: Dependency of processes

Process type	Dependant process
Performance tracer	--
Smart Agent[#1, #2]	--
CTM domain manager[#1]	Smart Agent[#1, #2]
CORBA naming service	--
CTM daemon[#1]	• Performance tracer • Smart Agent • CTM domain manager • CORBA naming service
J2EE server	• Performance tracer • Smart Agent • CTM domain manager[#3] • CORBA naming service[#4] • CTM daemon[#3]
Web server	Performance tracer

Legend:

--: There is no prerequisite process.

#1

It is a process to be started when using CTM.

#2

It is a process to be started when using the transaction service.

#3

This process is a prerequisite when using CTM.

#4

It is a process to be started when using the CORBA naming service in out-process. It is not required when using the CORBA naming service in in-process.

(2) Restart method of the process

The restart method of a process when the process terminates abnormally in a system is as follows.

Tip

When a CTM-related process abnormally terminates and cannot be started, use the following procedure:

1. When you cannot restart the process, check the cause of the trouble from the output error message.
2. Execute the `ctmrasget` command, and acquire backup of the execution environment.

(a) Restart procedure

The procedure of process restart (recovery) is as follows:

Table 2–9: Procedure to restart (recovery) processes

Abnormal process	Start command	Restart (recovery) procedure
Database server	--	Restart the DB server.
OpenTP1	--	Restart the OpenTP1.
Performance tracer	`cprfstart`	Restart the performance tracer.
Smart Agent[#1, #2]	`Osagent`	Restart the Smart Agent.
CTM domain manager[#1]	`ctmdmstart`	Restart the CTM domain manager.
CORBA naming service	`Nameserv`	Perform actions based on the procedure shown below. Furthermore, when CTM is not used, procedure 1 and 4 are not required. 1. Forcefully terminate the CTM daemon 2. Forcefully terminate the J2EE server 3. Restart the CORBA naming service 4. Restart the CTM daemon 5. J2EE server restart
CTM daemon[#1]	`ctmstart`	Perform actions based on the procedure shown below. 1. CORBA naming service forced termination 2. Forcefully terminate the J2EE server 3. Restart the CORBA naming service 4. Restart the CTM daemon 5. Before restarting the J2EE server
CTM regulator	--	Since restarted automatically by CTM daemon, restart is not required.
J2EE server	`cjstartsv`	Restart the J2EE server.
Web server	--	Restart the Web server.

Legend:

--: The start command differs depending on the products used or there is no corresponding start command.

#1

It is a process to be started when using CTM.

#2

It is the process to be started when using the transaction service.

2.5.2 If forced termination of a J2EE application fails

When forced termination of a J2EE application fails, the following information is output by the J2EE server:

- **Message**

  Message log is output. For details on output destinations of message logs, see *4.3 Application Server log (Systems for executing J2EE applications)*.

- **Stack trace**

  Output in the exception log and thread dump. For details on output destinations of exception logs, see *4.3 Application Server log (Systems for executing J2EE applications)*. For details on a thread dump, see *5.5 JavaVM Thread Dump*.

Note that in forced termination of a J2EE application, method cancellation is performed internally. For the error information output during method cancellation, see *5.3.13 Log information that is output while monitoring the execution time J2EE applications* in the *uCosminexus Application Server Operation, Monitoring, and Linkage Guide*.

2.5.3 If a Problem Occurs When Using the Database Session Failover Function

The actions for two types of errors in the case of errors in the database session failover function are described here:

- When an error occurs in an J2EE server
- When a trouble occurs in a database

(1) When an error occurs in an J2EE server

During a process that includes changes of the data in a database, if the process is down due to an error in the J2EE server, the consistency of the system will be maintained, because the database will return to the state prior to the occurrence of error the rollback operation. You can eliminate the error of J2EE server and restarting the Web application for recovering from the trouble.

When the host of the J2EE server hangs up or when an error occurs in the network, the operation of the database might be interrupted and the mutual exclusion (lock of the global session information) might become in an unresolved state. In this case, you must resolve the mutual exclusion of the unresolved state before the J2EE server is recovered and the Web application is re-opened.

Reference note

To release the unreleased mutual exclusion, an invalid connection to the database from the client is detected and the connected session can be terminated forcefully. For details on releasing an unreleased mutual exclusion, see the description related to the UAP processing time monitoring functionality of the manual *HiRDB UAP Development Guide* when using HiRDB. When using Oracle, see the Oracle manual.

(2) When a trouble occurs in a database

When an error occurs in a database and you recover the database, the Web application using the database session failover functionality can re-open the business, because the database will return to the state prior to occurrence of the error by the rollback operation. You are not required to restart the Web application.

For details on the countermeasures to be taken when an error occurrence in other databases, see the manual of each database.

2.5.4 If JavaVM Terminates abnormally

The actions when JavaVM terminates abnormally and the information output in the case of abnormal termination are described below:

(1) Actions in the case of abnormal termination

In UNIX, in the case of abnormal termination, perform actions in the following order. Further, these actions are not required in Windows.

1. Execute the `javatrace` command on the machine where abnormal termination occurs.

   Output the `javatrace.log` file. For the execution method of the `javatrace` command, see *4.18 JavaVM stack trace information*.

2. Send the acquired `javatrace.log` file to maintenance personnel along with the error report file (`hs_err_pid`*process-ID*`.log`).

   For the method of acquiring the error report file, see *4.11 JavaVM Output Message Logs (Standard Output or Error Report File)*.

3. Execute the following commands to create the archive file for the JavaVM executable file, the library loaded when the error occurred, and the core dump.

   - In AIX

     Execute the `snapcore` command. An archive file compressed with the `pax` command is created.

   - In Linux

     If the `compress` command is installed, execute the `car_tar_Z` command. An archive file compressed with the `compress` command is created.

     If the `gzip` command is installed, execute the `car_tar_gz` command. An archive file compressed with the `gzip` command is created.

4. Send the created archive file to maintenance personnel.

(2) Information output at the time of abnormal termination

The information output when the following kind of abnormal termination occurs in JavaVM is described below:

- When C heap is insufficient during JavaVM processing
- When an OutOfMemoryError occurs due to a reason other than insufficient C heap during JavaVM processing
- When an internal logical error occurred

(a) When C heap is insufficient during JavaVM processing

If C heap is insufficient, in continuation to the following messages, the memory status, Java heap information and stack trace information are output to the standard output and error report file (`hs_err_pid`*process-ID*`.log`). After that, JavaVM is forcefully terminated.

Check the output information and take an appropriate action.

```
Exception in thread thread-name java.lang.OutOfMemoryError:requested size-re
quired-for-securing-memory bytes [for message-for-internal-investigation].
```

(b) When an OutOfMemoryError occurs due to a reason other than insufficient C heap during JavaVM processing

When setting `-XX:+HitachiOutOfMemoryAbort` as the option when starting the J2EE server, and when an OutOfMemoryError occurs due to the following causes, output the message and perform forced termination of JavaVM. Check the output information and take an appropriate action.

When the OutOfMemory handling functionality is enabled (when `-XX:+HitachiOutOfMemoryHandling` is set), if an OutOfMemoryError occurs due to Java heap insufficiency or insufficient metaspace area, JavaVM might not be forcibly terminated.

Causes for forced termination

- Java heap insufficiency
- Insufficient metaspace area

- Insufficient C heap in J2SE class library

Further, when C heap is insufficient in the JavaVM process, perform forced termination regardless of the specification of this option.

When terminating, output the following message.

```
java.lang.OutOfMemoryError occurred.
JavaVM aborted because of specified -XX:+HitachiOutOfMemoryAbort options.
```

Forced termination timing

The timing when JavaVM is terminated forcefully differs depending on the option settings.

- When `-XX:+HitachiOutOfMemoryStackTrace` is specified as the option, terminate after performing output of the stack trace. However, even when registering the process that is to be executed at JavaVM termination time by the `java.io.File.deleteOnExit` method and the `java.lang.Runtime.addShutdownHook` method beforehand, perform forced termination without executing these.
- When setting `-XX:+HitachiOutOfMemoryAbortThreadDump` as the option, terminate after performing output of the thread dump. Especially, when the cause is Java heap insufficiency or insufficient metaspace area, if `-XX:+HitachiOutOfMemoryAbortThreadDumpWithJHeapProf` is also set as an option, JavaVM is terminated after the thread dump with statistical information according to the class is output.

Operation when the OutOfMemory Handling functionality is enabled

If `-XX:+HitachiOutOfMemoryHandling is set in addition to -XX:+HitachiOutOfMemoryAbort` as an option when starting a J2EE server, the OutOfMemory handling functionality gets enabled. In this case, if there is a Java heap insufficiency or insufficient metaspace area, a process to determine whether to perform forced termination is executed by OutOfMemory handling. As a result of this judgment process, if all of the following OutOfMemoryError throw conditions are satisfied, forceful termination is not executed.

OutOfMemoryError Throw Conditions

- OutOfMemory occurs due to Java heap insufficiency or insufficient metaspace area.
- OutOfMemory occurs in either the request processing in which the Web application (Servlet/JSP) on the Web container is running, the processing in which the Enterprise Bean invoked from an EJB client application is running, the processing in which a Message-driven Bean is running, or the processing in which the Enterprise Bean invoked from the TimerService is running.
- Does not correspond to the OutOfMemoryError throw exclusion condition.

OutOfMemoryError Throw exclusion condition

The total number of OutOfMemory occurrences due to Java heap insufficiency and OutOfMemory occurrences due to insufficient metaspace area (including the current OutOfMemory occurrence) within the past hour from the time when the current OutOfMemory occurred is greater than the value specified in the `-XX:HitachiOutOfMemoryHandlingMaxThrowCount` option.

For details, see *-XX:[+|-]HitachiOutOfMemoryHandling (OutOfMemory handling option)* in the *uCosminexus Application Server Definition Reference Guide*.

If the OutOfMemory handling functionality is enabled, whenever OutOfMemory occurs due to Java heap insufficiency or insufficient metaspace area, information on the frequency of OutOfMemory occurrences is output to the JavaVM log file.

(c) When an internal logical error occurred

When an internal logical error occurs, the message showing the JavaVM information where the error occurred, the examination error ID and the thread where the error occurred are output to the standard output and the error report file (`hs_err_pid`*process-ID*`.log`). Send the output information to maintenance personnel.

2.5.5 If Administration Agent is terminated forcibly when OutOfMemoryError occurs

When Administration Agent is running, if `java.lang.OutOfMemoryError` occurs, a log is output and the Administration Agent is terminated forcibly. The log is output at the following locations:

- In Windows
  Manager-log-output-directory`\adminagent.javalog[nn].log`
- In UNIX
  Manager-log-output-directory`/adminagent.javalog[nn].log`
  Note `[nn]` is a number from 0 to 99.

Note that if you have specified the settings to stop all the logical servers in the termination processing of the Administration Agent, the logical servers do not stop even if Administration Agent is terminated forcibly. To specify the settings for stopping the logical servers in the termination processing of the Administration Agent, specify `true` in the `adminagent.finalization.stop_servers` key of `adminagent.properties` (Administration Agent property file).

Also, if you use a product other than JDK that is provided by Cosminexus Application Server, the Administration Agent is not terminated forcibly because of `OutOfMemoryError`.

Execute the following operations, when Administration Agent is terminated forcibly because of `OutOfMemoryError`:

1. Revise the value specified in the following option of `adminagentuser.cfg`:
   Revise the value specified in the following option:
   `add.jvm.arg = -Xmx`
2. Restart the Administration Agent.

2.5.6 If a Problem Occurs in the System Linked with JP1

When an error occurs in the system linked with JP1, it is necessary to take the following actions:

(1) Actions for problems in the system linked with JP1/IM

Shown below are the actions for the problems expected in the system linked with JP1/IM:

Table 2–10: Errors expected in a system linked with JP1/IM and their actions

Problem type	Actions
Occurrence of an error at the time of auto generation of the monitoring tree	When an error occurs, investigate the cause of the error based on the Cosminexus adapter command message that is output in the JP1/Base plug-in service log file. For details on the JP1/Base plug-in service log file, see the manual *JP1/Base Operation Guide*.
No notification of the JP1 event to JP1/IM	Confirm the log of the Administration agent, Management agent, and Management Server to check whether the JP1 event is published to JP1/IM from the system built with Application Server. For details on the storage location of the log of Administration Agent, Management Agent, and Management Server, see *4.3.1 Acquiring the Cosminexus Component Container Logs* or *4.4.1 Acquiring the Cosminexus Component Container Logs (systems executing batch applications)*. Take the following actions for the contents of the log of Administration agent, Management agent, and Management Server. • When the JP1 event publishing log is not output, confirm the JP1 event publishing settings of the Management Server#. • When the JP1 event publishing log is output, confirm the contents of the JP1/Base configuration definition created in the JP1 integrated operation management server. Moreover, confirm the action environment settings of the JP1/Base event service in the Management Server# and J2EE server.
Web browser does not start after starting the JP1/IM-View monitor	Investigate the cause of the error that occurred based on the message output in the `mngsvrmonitor.log`, which is saved in the directory where the monitor start command is copied.

#
This is the Management Server of Application Server.

(2) Actions for problems in the system linked with JP1/AJS

(INTENTIONALLY DELETED)

2.5.7 If a problem occurs in 1-to-1 node switching systems

This subsection describes the actions that you must take with for each OS, when the node switching process for a standby host is timed out due to a failure in the database server (such as server is down or deadlock).

(1) In Windows

Manually keep the standby host online, after acquiring the log.

(a) Acquiring the log of 1-to-1 node switching system

If a trouble occurs in the 1-to-1 node switching system, you must acquire the cluster log. The following is the output destination of the cluster log when Windows is installed in the standard path:

```
C:\WINDOWS\cluster\cluster.log
```

The following information is output to a cluster log:

- **Operation log of cluster service**
- **Error message when there is an error in the syntax of VBScript**
- **Resource.LogInformation method of multi-purpose script**
- **Other messages**

For details on the cluster log, see the documentation on Windows.

If the J2EE server does not run, see Cosminexus Component Container log too.

(b) Manual recovery of 1-to-1 node switching system

You can manually recover the 1-to-1 node switching system according to the following procedure:

1. Cancel the cause of a timeout by restarting the database.

2. Create online target resources of the standby host.

(2) In UNIX

Resolve the cause of the timeout by restarting the database.

2.5.8 If a problem occurs in N-to-1 recovery systems

This subsection describes the recovery procedure for each OS, when problems occur in N-to-1 recovery systems.

(1) In Windows

With N-to-1 recovery systems, if the recovery processing of the standby node host (recovery server) is timed out because of a database server failure (such as server down or deadlock), collect the log and manually execute the recovery procedure. The recovery procedure is as follows:

1. Collect the log of the N-to-1 node switching system.

   If a problem occurs in the N-to-1 recovery system, you must collect cluster logs. Collect the logs same as you would collect when problems occur in 1-to-1 node switching systems. For details on the logs to be collected, see the subsection *2.5.7 If a problem occurs in 1-to-1 node switching systems*.

2. Recover the N-to-1 node switching system manually.

   Use one of the following methods to manually recover the N-to-1 recovery system:

   - Set up the target resources of the standby node host as online
   - Execute the transaction recovery command (`cjstartrecover`) of the J2EE server

(a) Setting the target resources of the standby node host as online and executing recovery

To set up the target resources of the standby node host as online and execute recovery:

1. Execute operations, such as restarting the database to resolve causes of the timeout.

2. Set up the target resources of the standby node host as online.

(b) Executing the transaction recovery command (cjstartrecover) of the J2EE server for recovery

To execute the transaction recovery command (`cjstartrecover`) of the J2EE server for recovery:

1. Execute operations, such as restarting the database to resolve causes of the timeout.

2. Create a folder in the path specified in `Dir_Name` using the universal script for the standby node host.
   If a folder already exists, delete the folder, and then create a new folder.

3. When the universal script for the standby node host is online, reference cluster logs, and then execute `cjstartrecover`.

4. When recovery is successful, delete the folder created in the path specified in `Dir_Name`.

For details on `cjstartrecover` command, see *cjstartrecover (recover J2EE server transaction)* in the *uCosminexus Application Server Command Reference Guide*.

(2) In UNIX

With N-to-1 recovery systems, if the recovery processing of the standby node host (recovery server) is timed out because of a database server failure (such as server down or deadlock), manually execute the recovery procedure as follows:

1. Execute operations, such as restarting the database to resolve causes of the timeout.

2. In the standby node host, execute `monbegin` for the failed executing node host.

   ```
   # monbegin server-identification-name
   ```

   In the underlined part, specify an identification name of the server for the executing node specified in the operand `alias` of the `servers` file.

3. In the standby node host, execute `monact` for the failed executing node host.

   ```
   # monact server-identification-name
   ```

   In the underlined part, specify an identification name of the server for the executing node specified in the operand `alias` of the `servers` file.

   You can use a standby node host (recovery server) to execute the recovery processing for the unconcluded transactions in the failed executing node host.

   Reference note

   For details on the environment settings to support servers in the HA monitor, such as defining the `servers` file, see *19.5.4 Setting the environment of the HA monitor* in the *uCosminexus Application Server Operation, Monitoring, and Linkage Guide*.

2.5.9 If a problem occurs in the node switching system for the host unit management model

An action for a problem that occurs in a node switching system of the host unit management model is same as would be the action of a problem that occurs in the 1-to-1 node switching system. For details, see the subsection *2.5.7 If a problem occurs in 1-to-1 node switching systems*.

2.5.10 If a Problem Occurs in the EJB Client

When an EJB client using a global transaction is down, recovery of the global transaction is necessary. The recovery of the global transaction can be executed by restarting the EJB client.

The method for confirming whether the global transaction recovery process is complete is described below:

- Execute the `cjlisttrn` command in Application Server and confirm whether the transaction is in an active state. For details on the `cjlisttrn` command, see *cjlisttrn (display information about transactions operating in J2EE server) in the uCosminexus Application Server Command Reference Guide*.
- Confirm whether the resource adapter can be stopped. If it can be stopped, the recovery process of the global transaction is complete.
- Confirm whether Application Server can be stopped successfully. When it can be stopped successfully, the recovery process of the global transaction is complete.
- Confirm using tools and commands provided by each resource.

Moreover, when multiple EJB clients exist, to examine which EJB client is required for the recovery of global transaction:

1. Confirm the unconcluded transactions by executing the `cjlisttrn` command by specifying the `-pending` option.

   Execution format

   ```
   cjlisttrn server-name -pending -bqual
   ```

   Execution example

   ```
   cjlisttrn MyServer -pending -bqual
   ```

2. From the unconcluded transactions confirmed in procedure 1., confirm whether all the following conditions apply to those transactions:
   - Elapsed time increases every time the `cjlisttrn` command is published
   - Branch type is Sub or Sub(recovered)
   - `TmHash` value of the KFCB40051-I message output at the time of starting the stopped EJB client is included in the global transaction ID

   The transactions to which all these conditions apply are required for the recovery of global transactions.

2.6 Precautions Related to Troubleshooting

This section describes the precautions related to troubleshooting.

2.6.1 Precautions Related to the System Log of an EJB Client Application

The precautions when referencing a system log output by an EJB client application in a shared sub-directory mode and operating the shared sub-directory mode EJB client applications are described below:

- The valid data in the log file is from the start of the file up to the EOF.

  In the system log of an EJB client application, the log data is overwritten from the beginning sequentially when the log file is wrapped around, without deleting the log data prior to wrap around. For this reason, when referencing a log file, ignore the data after the EOF. The data after the EOF is the invalid log file data prior to wrap around.

  The end of the valid log file data is the data shown below:

  ```
  EOF CRLF CRLF CRLF CRLF----------< End of Data >----------CRLF CRLF
  ```

  EOF are characters (0x1A) denoting the end of trace data. CRLF denotes line feed (0x0D, 0x0A).

  The example of output is described below. Further, the character showing the end of trace is described as [EOF].

  - In Windows

    ```
    **** "OS name (including details such as OS version)"               TZ=
    "time zone name"                      xxxx/xx/xx xx:xx:xx.xxx
         yyyy/mm/dd hh:mm:ss.sss               pid      tid      message-i
    d            message(LANG=ja)
    0000 xxxx/xx/xx xx:xx:xx.xxx     HEJB       BE3F6FE9 015EE671 KDJEXXXXX
    -W           xxxxxxxxx
    0001 xxxx/xx/xx xx:xx:xx.xxx     HEJB       BE3F6FE9 015EE671 KDJEYYYYY
    -I           yyyyyyyyy
    0002 xxxx/xx/xx xx:xx:xx.xxx     HEJB       BE3F6FE9 015EE671 KDJEZZZZZ
    -I           zzzzzzzzz
    [EOF]


    ----------< End of Data >----------

    <<Invalid data before wraparound>>...
    ...
    ...
    ```

  - In UNIX

    ```
    **** "OS name (Including details such as OS version)"               TZ=
    Asia/Tokyo                    xxxx/xx/xx xx:xx:xx.xxx
         yyyy/mm/dd hh:mm:ss.sss                     pid      tid      mes
    sage-id            message(LANG=ja)
    0000 xxxx/xx/xx xx:xx:xx.xxx     HEJB            BE3F6FE9 015EE671 KDJ
    EXXXXX-W           xxxxxxxxx
    0001 xxxx/xx/xx xx:xx:xx.xxx     HEJB            BE3F6FE9 015EE671 KDJ
    EYYYYY-I           yyyyyyyyy
    0002 xxxx/xx/xx xx:xx:xx.xxx     HEJB            BE3F6FE9 015EE671 KDJ
    EZZZZZ-I           zzzzzzzzz
    [EOF]
    ```

```
----------< End of Data >----------

<< Invalid data before wraparound>>...
...
...
```

- At the time of starting an EJB client application process, a number of trace files specified in the `ejbserver.logger.channels.define.`*channel-name*`.filenum` key of system properties are created. At this time, trace files are initialized as space (0x20).
- The capacity of the user log file is fixed as specified in the `ejbserver.logger.channels.define.`*channel-name*`.filesize` key of system properties. The trace files of the specified size are created at the time of process startup. For this reason, capacity does not increase or decrease as per the log output.
- When you want to change the log file capacity or the number of files, you need to stop all the processes that output log in the relevant log files, and delete the log management files under the `mmap` directory and the log files or move them to a different directory.
- In cases other than when you want to change the log file capacity or the number of files, do not change or delete the log files and the log management files. If you change or delete them, thereafter, the log may not be output correctly.
- Do not delete the sub directories using the `cjcldellog` command where the EJB client application running in the shared sub-directory mode output log. If you delete such sub-directories, thereafter, the log may not be output correctly.
- When starting an EJB client application in the shared sub-directory mode in the environment where the EJB client application is already running in the exclusive sub directory mode, specify a value different than that of the EJB client application running in an exclusive sub-directory mode in the `ejbserver.client.ejb.log` key of system properties. If the same value is specified, you cannot correctly manage the number of sub directories of the EJB client application operating in the exclusive sub-directory mode. Note that the exclusive sub-directory mode is used for compatibility with earlier versions.

2.6.2 Precautions When Using CTM

The precautions to be taken when using CTM are described below. In CTM, take care of the following things:

- Failure information of CTM is acquired under the CTMSPOOL environment variable in the CTM domain unit. As the error information is acquired even at the time of restart after the CTM daemon and CTM domain manager are down, save the error information after the occurrence of error.
- The directories under the `CTMSPOOL` environment variable are the operation directories of the product. Therefore, do not delete the files and directories.

2.6.3 Precautions when using PRF

The precautions to be taken when using PRF are described below. In PRF, take care of the following things:

- PRF error information is acquired under the PRFSPOOL environment variable for every PRF identifier. As the error information is acquired even at the time of restart after the PRF daemon is down, save the error information after the occurrence of error.

- The directories under the `PRFSPOOL` environment variable are the operation directories of the product. Therefore, do not delete the files and directories.

2.6.4 JavaVM data-related considerations

This subsection describes the JavaVM data-related considerations.

- The following JavaVM data does not support multi-byte characters. The applicable characters, such as the Japanese class names, are corrupted.
  - JavaVM thread dump log file
  - JavaVM log file
  - Error report file (JavaVM output message log)
  - Event log file of the Explicit Memory Management functionality
- If you specify names containing the third and fourth level characters of JIS X0213:2004 in the following options, the JavaVM log file of the product and the event log file of the Explicit Memory Management functionality are not output:
  - `-XX:HitachiJavaLog` option
  - `-XX:HitachiExplicitMemoryJavaLog` option

  Also, if the third and fourth level characters of JIS X0213:2004 are included in the output destination directory of the extended thread dump file, the extended thread dump is not output to a file.

3

Preparing for Troubleshooting

You can automatically receive the information required for troubleshooting, when a trouble occurs in a system built on the Application Server. Some of the data that is required for troubleshooting need preparation before starting operations. This chapter describes the settings for respective data acquisition.

3.1 Organization of this chapter

This section describes the settings to acquire the data required for troubleshooting.

You can automatically acquire the data required for troubleshooting by default settings, but you must set up a part of the data before starting operations. You can also change the output destination and the log size that is set up by default.

For the information to be acquired when trouble occurs, you must specify the settings before starting the operations to acquire the following information:

- JavaVM log (JavaVM log file)
  The JavaVM GC log is also output to this file.
- User dump (in Windows)
- core dump (in UNIX)
- OS statistical information (in Windows)

Specify the settings to acquire this information at system configuration time as required.

The following table describes the organization of this chapter.

Table 3–1: Organization of this chapter (Troubleshooting (Data acquisition settings))

Category	Title	Reference
Explanation	Overview of data acquisition settings	*3.2*
Settings	Execution environment settings	*3.3*

For overview of troubleshooting, data output methods and output contents, see the respective sections:

- Overview of troubleshooting and methods to output the data automatically
  2. Troubleshooting
- Output destination of collected data and methods to output data separately
  4. Output Destinations and Output Methods of Data Required for Troubleshooting
- Contents output in data
  5. Problem Analysis

3.2 Overview of data acquisition settings

For certain materials required for troubleshooting, you need to specify collection of the material before starting the operations. For example, if you do not make settings in advance for collecting material, you cannot acquire the statistical information of the OS, user dumps (in Windows) or core dumps (in UNIX), or GC log of JavaVM. Hitachi recommends that you obtain this data because it is required for troubleshooting.

If the material can be collected with default settings, specific settings are not required, however, if you want to change the log output destination and size, edit the Easy Setup definition file and user definition file to change the settings.

The following table describes handling of the data required for troubleshooting, data that require in advance settings, and the data for which you must change the settings.

Table 3–2: Handling of the data required for troubleshooting, data that require in advance settings, and the data for which the settings must be changed

Data necessary for troubleshooting		Data that require in advance settings or data for which the settings must be changed
Log of Application Server	Message log	• Snapshot log • Management Server log • J2EE server log • Batch server log • Web server log • Application user log • Cosminexus Manager log • Console log • Resource adapter log • Cosminexus TPBroker trace file • Cosminexus JMS Provider log • Server management command log • NIO HTTP server log
	User log	
	Exception log	
	Maintenance log	
EJB client application system log	Message log	• System log# • Application user log
	User log	
	Exception log	
	Maintenance log	
Trace based performance analysis		Trace based performance analysis file
Thread dumps of JavaVM		JavaVM material
GC logs of JavaVM		JavaVM material
Memory dump		• User dump (in Windows) • core dump (in UNIX)
Hitachi JavaVM log file		JavaVM material
Error report files		--
Compilation replay file		--
OS state or log		--
Statistical information of OS		Statistical information of OS
Definition information		--

Data necessary for troubleshooting	Data that require in advance settings or data for which the settings must be changed
Operation directory	--
Resource setting	--
Web server log	--
JavaVM stack trace	--
Event log of Explicit Memory Management functionality	JavaVM material

Legend:

--: No data for in advance settings and no data to change the settings

Note:

Besides this table, there is an operation information file for the data that requires in advance settings and the data requires changes in settings. An operation information file is a file used for acquiring the operation information, such as server performance for each functionality and the resource information. You can use this file for operation monitoring of systems. You can acquire the operation information file by default.

#

For details about the system logs of EJB client applications, see *3.8 System log output for EJB client application* in the *uCosminexus Application Server EJB Container Functionality Guide*.

This section gives an overview on the types and settings of the data that require in advance settings and the data that requires changes in the settings, for each system executed that executes an application.

3.2.1 Specifiable contents

This section describes the contents that can be specified as data acquisition methods.

You can specify the following contents as acquisition methods in some of the data to be used for troubleshooting. The contents that you can specify depend on the data.

- **Do you want to acquire the data?**
- **Data output destination**
- **Number of output destination files**
- **How to switch the output destination files**

Note that for some data output by the following processes, you can select the time at which the output destination files will be switched and the name assignment rules for the switched files:

- J2EE server
- Administration Agent
- Management Server#
- Internal setup tool for the virtual server manager
- Server Communication Agent

Applicable when the shift mode is set up for the integrated log.

This subsection describes the time at which the output destination files can be switched and the file name assignment rules that you can select in these processes.

(1) Time at which the output destination files are switched

For some of the data, you can select the switching methods for the data output destination files from the following two types of methods:

File size-based switching method

In this method, when the output destination file reaches a certain size, the log output destination is switched to the next file. You can specify the file size for each data item. However, the file size is fixed for some of the data.

Time and file size-based switching method

In this method, when the specified time is reached, or when the output destination file reaches a certain size, the log output destination is switched to the next file.

You can specify the switching method for each process.

If you select the time and file size-based switching method, and if you want to switch the files by time alone, specify a value larger than the assumed log output volume as the file size. Also, the header information of the new file, created after the output destination file is switched by time, becomes the date and time at which the process that has output the log initialized the log.

For details on the files that are switched by the time and file size-based switching method, see *5.2 Application Server Log*.

Important note

When you select the time and file size-based switching method, the time at which the output destination file is switched might be delayed from the specified time by several milliseconds to a few seconds, depending on the load status of the OS resources.

Also, if the process that performs log output (such as the J2EE server) is not running at the specified time, the log is added into the existing file instead of switching the files when the process next starts after the specified time is past. For example, if the switching time is specified at 12:00:00, and the process is stopped at 11:00:00, the process that starts at 13:00:00 adds the log to the existing file.

(2) File name assignment rules when the output destination files are switched

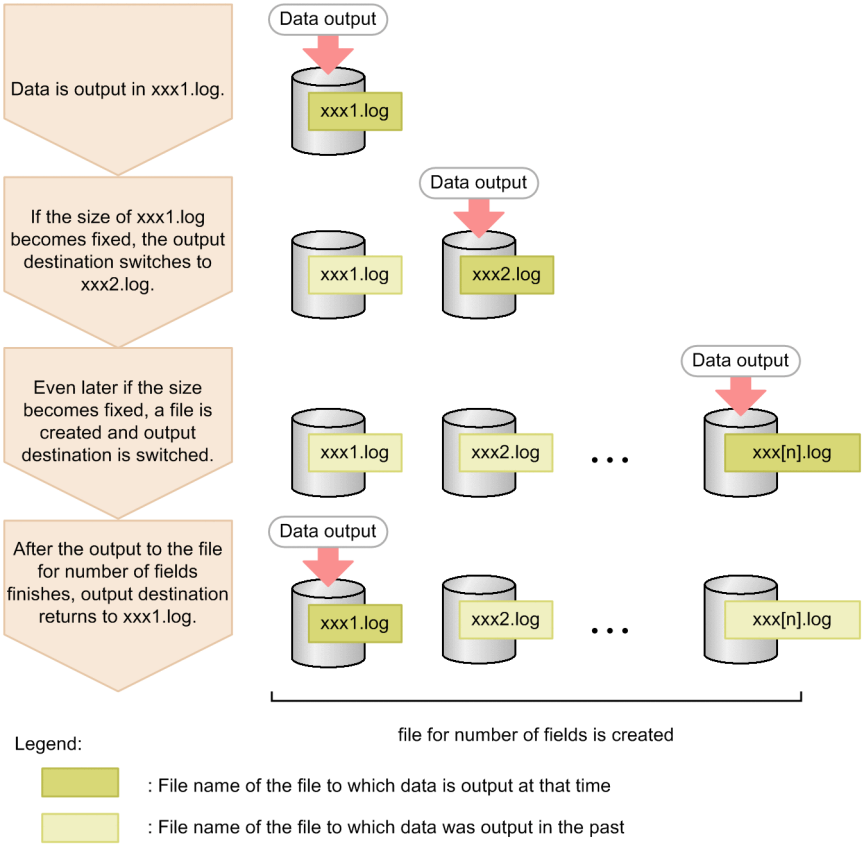
A file name configured with a data-specific string and serial number is set for the output destination files. You can select the serial number assignment rules from the following two modes:

Wraparound mode

If the data-specific string is *xxx*, the files are created with the serial number added as *xxx*`1.log`, *xxx*`2.log` and so on. The range of the serial numbers is from 1 up to (*number-of-output-destination-files*). If the output destination is changed after the specified number of output destination files are created, the data output destination switches to the file with the serial number 1.

The following figure shows file name assignment rules in the wraparound mode.

Figure 3–1: File name assignment rules in the wraparound mode



Shift mode

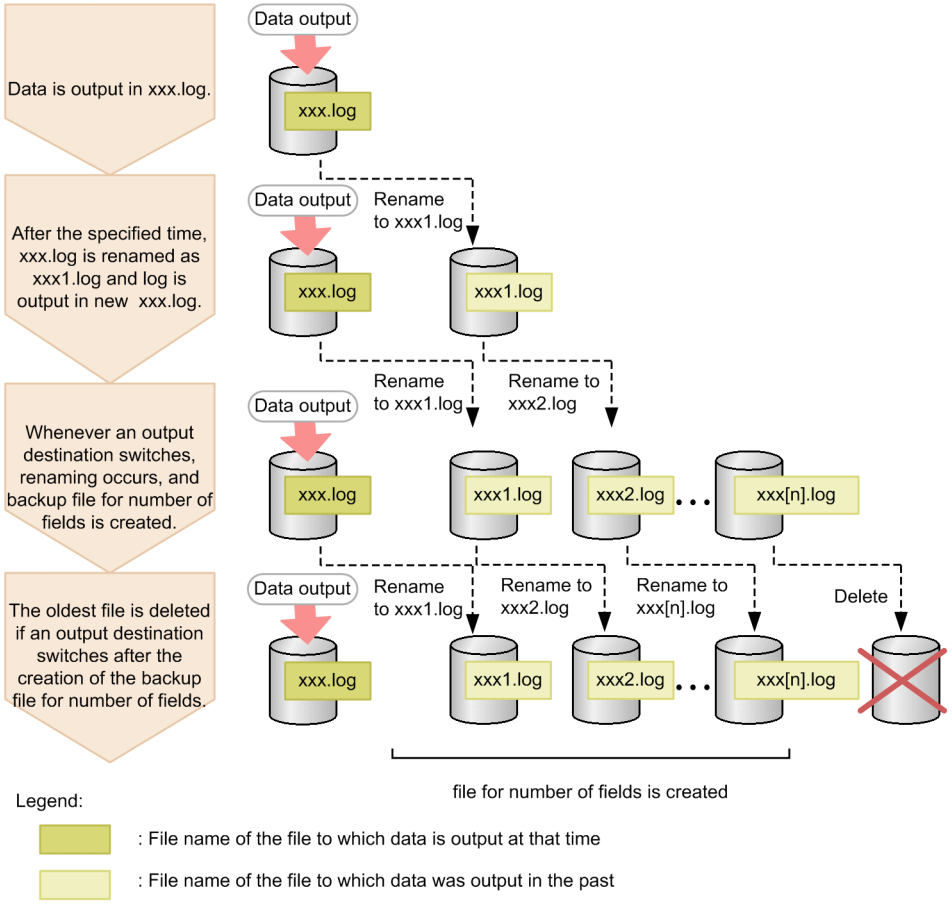
The file to which the data is being output currently has a file name containing only the data-specific string. No serial number is given. After the output destination is switched at the specified time, a serial number is given to the previous files.

If the data-specific string is *xxx*, the file to which the data is output at that time becomes xxx.log (file name without serial number). The names of the previous output destination files are changed (renamed) to file names with the serial number added as *xxx*`1.log`, *xxx*`2.log` and so on. At this time, the file with the smallest serial number sequentially becomes the latest file.

The range of the serial numbers is from 1 up to (*number-of-output-destination-files*). Therefore, the total number of files is the *specified-number-of-output-destination-files* `+ 1` (current output destination file).

The following figure shows the procedure of switching output destination in the shift mode.

Figure 3–2: Procedure of switching the output destination in the shift mode



For details on the files that are switched in the shift mode, see *5.2 Application Server Log*.

Note that when deleting a file in Windows, a file that is being used in another process might be stored temporarily. In this case, a file with `#removed#[n]` (n is an integer of 0 or more value) added at the end of the original file name is created temporarily. This file is automatically deleted when all the processes finish processing the file.

3.2.2 Overview of data acquisition settings (Systems that execute J2EE applications)

For changing the default settings (Output destination or log size) of data or acquiring the data that is not acquired by default, editing the Easy Setup definition file or User definition file and specify the settings. This subsection describes whether you must specify the settings of the systems executing J2EE applications to acquire the data (log) in advance. This subsection also provides an overview of the setting methods.

For each data type, the following tables separately describe the data items for which the data collection settings must be specified and the data items for which settings must only be specified when changing the default data collection settings.

- **Data items that require data collection settings**

Table 3–3: Data acquisition settings (Systems executing J2EE applications)

Type of material	Settings for collecting the material	Reference manual	Reference point
Application user log	Set up the logger or handler settings, output level of log, size, and the number of files in the <configuration> tag of logical J2EE server (j2ee-server), in the Easy Setup definition file. Use server.policy to specify the security policy.	*uCosminexus Application Server Expansion Guide*	*Chapter 8*
Statistical information of OS	In Windows, specify the performance data collection settings of system resources on the Windows system monitor.	*This manual*	*3.3.14*
User dump	In Windows, specify the settings for collecting user dumps by using Task Manager, the Windows debug tool, or the environment variable (CJMEMDUMP_PATH).	*This manual*	*3.3.15*
core dump	In Unix, specify the settings for collecting core file using the simple setup definition file and shell commands.	*This manual*	*3.3.16*
JavaVM material	Set up the thread dump of JavaVM, output methods, or output contents of the JavaVM logs (JavaVM log files) in the Easy Setup definition file. You can also set up name of a file that executes the event log output or the output level of log in the Explicit Memory Management functionality.	*This manual*	*3.3.17*

- **Data items that must be set up only when changing the default data collection settings**

Table 3–4: Data acquisition settings (Systems executing J2EE applications)

Type of material	Settings for collecting the material	Reference manual	Reference point
snapshot log	To change the collection destination, collection method, and collection timing of the snapshot log, edit the user definition file	*This manual*	*3.3.1*, *3.3.3*
Management Server log	Use mserver.properties (environment setting files of the Management Server), to set up an output level of a log or to set up the number of log files.	*This manual*	*3.3.5*
Trace based performance analysis file	Use mserver.properties (environment settings file of Management Server) to specify the number of Trace based performance analysis files. Specify the trace acquisition level of performance tracer or number of PFR trace files in the <configuration> tag of Logical Performance Tracer (performance-tracer) in the Easy Setup definition file.	*This manual*	*7.5*
J2EE server log	Set up the output level of a log, size, and number of files in <configuration> tag of logical J2EE server (j2ee-server) in the Easy Setup definition file. When settings of system log output are enabled by ejbserver.logger.systemlog.enabled in the <configuration> tag of logical J2EE server (j2ee-server), the start and stop of J2EE server and abnormal termination messages are output in an event log (in UNIX, syslog).	*This manual*	*3.3.6*
Web server log	Set up the output level of a log and output destination of Web server in the <configuration> tag of logical Web server (web-server) in the Easy Setup definition file.	*This manual*	*3.3.8*
NIO HTTP server log	Set up the availability of log output in the NIO HTTP server and the number of files in the <configuration> tag of logical J2EE server (j2ee-server), in the Easy Setup definition file. You can define the format to customize the output format of an access log.	*This manual*	*3.3.9*

Type of material	Settings for collecting the material	Reference manual	Reference point
	For details about customizing access logs of the NIO HTTP server, see *6.11.2 Customizing the access logs of the NIO HTTP server* in the *uCosminexus Application Server Web Container Functionality Guide*.		
Operation information file	Set up output destination of operation information file or number of files in the `<configuration>` tag of logical J2EE server (j2ee-server) in the Easy Setup definition file.	*uCosminexus Application Server Operation, Monitoring, and Linkage Guide*	*3.3.3*
Cosminexus Manager log	Use manager.cfg to specify the number and size of the files of the integrated log.	*This manual*	*3.3.10*
Console log	Use adminagent.properties to specify the application of output, the number, and the size of the console log files.	*uCosminexus Application Server Operation, Monitoring, and Linkage Guide*	*11.3*
Resource adapter log	Specify the application of log output for each resource adapter using Server management commands. Set up the output level of a log, size, and the number of files in the `<configuration>` tag of logical J2EE server (j2ee-server), in the Easy Setup definition file.	*This manual*	*3.3.11*
Cosminexus TPBroker trace file	Set up the output destination of trace file or number of files in the Easy Setup definition file (in the `<configuration>` tag of logical J2EE server (j2ee-server)) and `usrconf.bat` (in UNIX, `usrconf`) and `usrconf.properties` for the server management commands.	*This manual*	*3.3.12*
Cosminexus JMS Provider log	Set up the output level, number of files, and log file size in `commonconfig.properties` or `config.properties` and `admin.properties` for the CJMSP Broker and the management command (`cjmsicmd`) log. Also, set up the output level, number of files, and log file size of the CJMSP resource adapter log in the Connector property file.	*This manual*	*3.3.13*
Server management command log	You can use usrconf.bat (in UNIX, usrconf) and usrconf.properties for server management commands to specify the log output level.	*uCosminexus Application Server Application Setup Guide*	*3.4*

The snapshot logs described here can be collected in batches. In the trace file of Cosminexus TPBroker, the material that can be collected and the material that cannot be collected are mixed. In addition to this, you need to make the settings for collection and add the collection destination of the snapshot log for those logs that cannot be acquired with default settings. Collect user dumps (in Windows) or core dumps (in UNIX) for specific file names. To collect the material when an error occurs, you need to use the user created command at error detection time.

Note that you cannot change the log output destination for the following logs:

- The `install.log` created during setup or log files of migration commands
- The Java thread dump files output to the following directories:
  - In Windows
    working-directory`\ejb\`*server-name*
  - In UNIX
    working-directory`/ejb/`*server-name*

For details about log types or default values, channel names, details of logs that can be acquired or acquisition methods, see *2.4 Types of Required Data*.

3.2.3 Overview of data acquisition settings (Systems executing batch applications)

For changing the default settings (output destination or log size) of data or acquiring the data that is not acquired by default, edit the Easy Setup definition file or the user-defined file. This subsection describes whether you must specify the settings for the systems executing batch applications to acquire the data (log) in advance. This subsection also provides an overview of the setting methods.

For each data type, the following tables separately describe the data items for which the data collection settings must be specified and the data items for which settings must only be specified when changing the default data collection settings.

- **Data items that require data collection settings**

Table 3–5: Data acquisition settings (Systems executing batch applications)

Type of material	Settings for collecting the material	Reference manual	Reference point
Application user log	Set up the logger or handler settings, output level of a log, size and number of files in the `<configuration>` tag of logical J2EE server (j2ee-server), in the Easy Setup definition file. Use `server.policy` to specify the security policy.	*uCosminexus Application Server Expansion Guide*	*Chapter 8*
Statistical information of OS	In Windows, specify the performance data collection settings of system resources on the Windows system monitor.	*This manual*	*3.3.14*#
User dump	In Windows, specify the settings for collecting user dumps by using the Windows debug tool.	*This manual*	*3.3.15*#
core dump	In Unix, specify the settings for collecting core file using the simple setup definition file and shell commands.	*This manual*	*3.3.16*#
JavaVM material	Set up the thread dump of JavaVM, output methods or output contents of the JavaVM logs (JavaVM log files) in the Easy Setup definition file. You can also set up file names to the output event log or output level of logs of the Explicit Memory Management functionality.	*This manual*	*3.3.17*#

#

The setting method is same as that of the J2EE server. Substitute *J2EE server* in the description of reference sections to *batch server*, and read.

- **Data items that must be set up only when changing the default data collection settings**

Table 3–6: Data acquisition settings (Systems executing batch applications)

Type of material	Settings for collecting the material	Reference manual	Reference point
snapshot log	To change the collection destination, collection method, and collection timing of the snapshot log, edit the user definition file	*This manual*	*3.3.2*, *3.3.4*
Management Server log	Use `mserver.properties` (environment settings file of the Management Server) to set up the output level of a log and number of log files.	*This manual*	*3.3.5*#
Trace based performance analysis file	Use `mserver.properties` (environment settings file of Management Server) to specify the number of trace based performance analysis files.	*This manual*	*7.5*#

Type of material	Settings for collecting the material	Reference manual	Reference point
	Specify the trace acquisition level of performance tracer or number of PFR trace files in the <configuration> tag of logical performance tracer (performance-tracer) in the Easy Setup definition file.		
Batch server log	Set up the output level of a log, size, and number of files in the <configuration> tag of logical J2EE server (j2ee-server), in the Easy Setup definition file. When the settings of system log output are enabled by ejbserver.logger.systemlog.enabled in the <configuration> tag of logical J2EE server (j2ee-server), the start and stop of batch server and abnormal termination messages are output in an event log (in UNIX, syslog).	*This manual*	*3.3.7*
Operation information file	Set up the output destination of an operation information file or number of files in the <configuration> tag of logical J2EE server (j2ee-server), in the Easy Setup definition file.	*uCosminexus Application Server Operation, Monitoring, and Linkage Guide*	*3.3.3*
Cosminexus Manager log	Use manager.cfg to specify the number and size of the files of the integrated log.	*This manual*	*3.3.10*#
Console log	Use adminagent.properties to specify the application of output, the number, and the size of the console log files.	*uCosminexus Application Server Operation, Monitoring, and Linkage Guide*	*11.3*#
Resource adapter log	Specify the application of log output for each resource adapter using Server management commands. Set up the output level of a log, size, and number of files in the <configuration> tag of logical J2EE server (j2ee-server), in the Easy Setup definition file.	*This manual*	*3.3.11*#
Cosminexus TPBroker trace file	Set up the output destination of trace files or number of files in the Easy Setup definition file (in the <configuration> tag of logical J2EE server (j2ee-server)) and the usrconf.bat (in UNIX, usrconf) and usrconf.properties for the server management commands.	*This manual*	*3.3.12*#
Server management command log	You can use usrconf.bat (in UNIX, usrconf) and usrconf.properties for server management commands to specify the log output level.	*uCosminexus Application Server Application Setup Guide*	*3.4*

#

The setting method is same as that of the J2EE server. Substitute *J2EE server* in the description of reference sections to *batch server*, and read.

The snapshot log can collects the logs described in table, in batches. In the trace file of Cosminexus TPBroker, the material that can be collected and the material that cannot be collected are mixed. In addition to this, you need to make the settings for collection and add the collection destination of the snapshot log for those logs that cannot be acquired with default settings. Collect user dumps (in Windows) or core dumps (in UNIX) for specific file names. To collect the material when an error occurs, you need to use the user created command at error detection time.

Note that you cannot change the log output destination for the following logs:

- The install.log created during setup or log files of migration commands
- The Java thread dump files output to the following directories:
  - In Windows

working-directory\ejb*server-name*

- In UNIX
  working-directory/ejb/*server-name*

For details about the log types or default values, channel names, details of logs that can be acquired or acquisition methods, see *2.4 Types of Required Data*.

3.3 Execution environment settings

Among the data required for troubleshooting, some data require the settings before starting operations and for some data you can change the default settings. This section describes how to set up each material. For details about the data that require in advance settings or changes in the settings, see *3.2 Overview of data acquisition settings*.

3.3.1 Data acquisition settings using failure detection time commands (Systems for executing J2EE applications)

This subsection describes how to specify the settings to acquire the data for troubleshooting using the failure detection time commands. Note that you can collect the material acquired by the failure detection time commands as the snapshot log.

There are two types of failure detection time commands; commands that the system provides and commands that the user creates. According to the default settings, when an error occurs in a logical server, the failure detection time commands provided by the system are executed and thread dumps and trace based performance analysis are acquired. The snapshot log is collected before terminating the logical server where the error occurs. For the information that can be acquired by executing the failure detection time commands provided by the system, see *2.3.2(1) Information that can be acquired by executing the failure detection time commands provided by the system*.

To change the operation settings of the failure detection time commands provided by the system, *(1) Environment settings in the Management Server* and *(2) Environment settings in the Administration Agent* are necessary. Also, when using the user created failure detection time commands, *(1) Environment settings in the Management Server*, *(2) Environment settings in the Administration Agent*, and *(3) Creating a command file of the user created failure detection time commands* are necessary. Respective settings are described in points from *(1)* to *(3)*.

Important note

To collect the material acquired by the user created failure detection time commands as the snapshot log, you need to add the collection destination of that material to the snapshot log collection destination. For details about addition of the snapshot log collection destination, see *3.3.3(3) Customizing the snapshot log collection destination*.

(1) Environment settings in the Management Server

Use `mserver.properties` (environment settings file of Management Server) to specify the operation of the failure detection time commands.

Specify the operation of the failure detection time commands in the following keys.

Key	Description	Setting requirement	
		System	User
`com.cosminexus.mngsvr.sys_cmd.abnormal_end.enabled`	Specify whether to use system provided failure detection time commands. The default setting is true (use).	O	--
`com.cosminexus.mngsvr.usr_cmd.abnormal_end.enabled`	Specify whether to use the user created failure detection time commands. The default setting is false (do not use).	--	R

Key	Description	Setting requirement	
		System	User
com.cosminexus.mngsvr.sys_cmd.abnormal_end.timeout	Specify the waiting period for termination of system provided failure detection time commands. If the command does not terminate even after the specified time lapses, the user recovery process continues.	O	--
com.cosminexus.mngsvr.usr_cmd.abnormal_end.timeout	Specify the waiting period for termination of the user created failure detection time commands.	--	O
com.cosminexus.mngsvr.snapshot.auto_collect.enabled	Specify whether to acquire the snapshot log when an error occurs or for batch restart. The default setting is true (acquire the snapshot log).	O	O
com.cosminexus.mngsvr.snapshot.collect.point	Specify one of the following as the snapshot log collection timing: • Before terminating the logical server • Before restarting the J2EE server The default timing is before terminating the logical server.	O	O

Legend:

System: You need to set the system provided failure detection time commands.

User: You need to set the user created failure detection time commands.

R: Required

O: Required only when changing the default settings.

--: Not required

(2) Environment settings in the Administration Agent

Use adminagent.properties (Administration Agent property file) to specify the material to be acquired by the failure detection time commands.

In the following keys of adminagent.properties, specify the count of the material to be acquired, application of collection using the failure detection time commands, and the path of the failure detection time commands. For details on the files defining the snapshot log collection target, see *3.3.3(3) Customizing the snapshot log collection destination*

Key	Description	Setting requirement	
		System	User
adminagent.snapshotlog.num_snapshots	Specify the number of snapshot log files to be collected as the primary delivery data for each logical server.	O	O
adminagent.snapshotlog.listfile.2.num_snapshots	Specify the number of snapshot log files to be collected as the secondary delivery data for each logical server.	O	O
adminagent.j2ee.sys_cmd.abnormal_end.threaddump	Specify whether to acquire thread dumps using the system provided failure detection time commands.	O	--
adminagent.sys_cmd.abnormal_end.prftrace	Specify whether to acquire the trace based performance analysis file using the system provided failure detection time commands.	O	--
adminagent.*logical-server-type*.usr_cmd.abnormal_end	Specify the path of failure detection time commands to be executed for each type of logical server.	--	R

Legend:

System: You need to set the system provided failure detection time commands.

User: You need to set the user created failure detection time commands.

R: Required

O: Required only when changing the default settings.

--: Not required

(3) Creating a command file of the user created failure detection time commands

You can code the user created failure detection time commands in a command file (batch file or shell script file). At this time, you can code the environment variables described in the following table, in the command file to execute the commands using the information of the logical server where the error occurred and the information related to the error.

Table 3–7: Environment variables that you can code in the command file of the user created failure detection time commands

Environment variable	Description
COSMI_MNG_LSNAME	Logical server name of the logical server where the error occurred. When an error occurs in the naming service of the logical CTM, the logical server name of logical CTM will be set up.
COSMI_MNG_RSNAME	Actual server name of the logical server where the error occurred. For a logical server other than a J2EE server, the logical server name is set.
COSMI_MNG_LSPID	Process IDs to be monitored when the logical server starts. When monitoring multiple process IDs on an indirectly started logical user server, the process IDs are specified, demarcated by commas (,) in the order in which the process IDs are acquired by the command executed for acquiring the process IDs when the logical user server is started.
COSMI_MNG_LSARGS	Command line when the logical server is started.
COSMI_MNG_TIME_SUSPENDED	Time at which hang up is detected. Time lapsed (unit: ms) from 0 hour before January 1, 1970 of the universal coordinated time (UTC). Note that the value is set only if the response is detected.
COSMI_MNG_TIME_TERMINATED	Time at which abnormal termination (process down) is detected. Time lapsed (unit: ms) from 0 hour before January 1, 1970 of the universal coordinated time (UTC). Note that the value is not set if hang up occurs.
COSMI_MNG_WEB_SYSTEM	Web system affiliated to the logical server where an error occurs. The value is not required if you do not use the Smart Composer function.
COSMI_MNG_TIER	Physical tier affiliated to the logical server where an error occurs. The value is not required if you do not use the Smart Composer function.
COSMI_MNG_UNIT	Service unit affiliated to the logical server where an error occurs. The value is not required if you do not use the Smart Composer function.
COSMI_MNG_HWS	Cosminexus HTTP Server installation directory.

The Management Server cannot acquire the standard output and standard error output from the commands executed as commands to detect error. To acquire the standard output and standard error output of a command, information must be output to a file during command execution.

(a) Examples of obtaining the core dump

The following examples describe the execution of the `kill` command when an error is detected in the J2EE server and the collection of core dumps:

- **In UNIX**

```
#!/bin/sh

# Determine whether the rem error has occurred because the process is dow
n or hung up, from the date and time at which it is detected that the proc
ess is down.
if [ "$COSMI_MNG_TIME_TERMINATED" = "" ] ; then

# Acquire a core dump because the error occurred due to hang-up of the pro
cess.
/bin/kill -6 $COSMI_MNG_LSPID
fi
```

(b) Example of obtaining the thread dump

The following is an example of the case in which the `cjdumpsv` command is executed to obtain the J2EE server (real server name: `J2EEServer`) thread dump when a Web server error occurs.

In this example, the `cjdumpsv` command is executed multiple times to check the status transition of each thread in accordance with the lapsed time. As a standard, the `cjdumpsv` command is executed about ten times every three seconds.

- **In Windows**

```
Determine whether the rem error has occurred because the process is down o
r hung up, from the environment variables.
if defined COSMI_MNG_TIME_TERMINATED goto END

Acquire the thread dump because the rem error has occurred due to the hung
-up process.
set COUNT=10
set INTERVAL=3000
for /l %%n in (1,1,%COUNT%) do (
"C:\Program Files\Hitachi\Cosminexus\CC\server\bin\cjdumpsv.exe" J2EEServ
er
   if not "%%n" == "%COUNT%" (
       rem Stand by until the next thread dump is collected.(milliseconds)
       echo WScript.sleep %INTERVAL% > sleep.vbs
       "C:\WINDOWS\\system32\cscript.exe" sleep.vbs > NUL
       del sleep.vbs
   )
)
:END
```

- **In UNIX**

```
#!/bin/sh

# Determine whether the error has occurred because the process is down or
hung up, from the environment variables.
if [ "$COSMI_MNG_TIME_TERMINATED" = "" ] ; then

# Acquire the thread dump because the error has occurred due to the hung-u
p process.
COUNT=10
INTERVAL=3
for num in 'seq $COUNT'
```

```
do
  /opt/Cosminexus/CC/server/bin/cjdumpsv J2EEServer
  if [ "$num" -ne "$COUNT" ]; then
      # Stand by until the next thread dump is collected.(Seconds)
      sleep $INTERVAL
  fi
done
fi
```

(c) Operating user-created failure detection time commands

The logical CTM starts, stops, and monitors two processes; the global CORBA Naming Service and the CTM daemon. There are different execution commands for the case when an error is detected in the global CORBA Naming Service and in the CTM daemon respectively, within the logical server.

- **When detecting an error in global CORBA Naming Service**
  The command specified in the `adminagent.naming.usr_cmd.abnormal_end` key is executed.
- **When detecting an error in CTM daemon**
  The command specified in the `adminagent.ctm.usr_cmd.abnormal_end` key is executed.

Moreover, an error is detected in either of the two processes (CTM daemon or global CORBA Naming Service) in the logical CTM, therefore, the log that reports the startup of the failure detection time commands in the logical server (CTM) will be output in a Management Server log.

3.3.2 Data acquisition settings using failure detection time commands (Systems for executing batch applications)

This subsection describes the settings to acquire the data for troubleshooting using the failure detection time commands. You can collect the material acquired by using the failure detection time commands as the snapshot log.

There are 2 types of failure detection time commands namely, system provided commands and user created commands. According to the default settings, when an error occurs in a logical server, the failure detection time commands provided by the system are executed and thread dumps and trace based performance analysis are acquired. The snapshot log is collected before terminating the logical server where the error occurs. For the information that can be collected by using the failure detection time command provided by system, see *2.3.2(1) Information that can be acquired by executing the failure detection time commands provided by the system*.

To change the operation settings for the failure detection time commands provided by the system, you must make environment settings in Management Server and Administration Agent. When using the failure detection time commands created by the user, you must make environment settings in Management Server and Administration Agent as well as create a command file for the failure detection time commands created by the user. For respective settings, see *3.3.1 Data acquisition settings using failure detection time commands (Systems for executing J2EE applications)*. Here, read "J2EE server" as "Batch server".

Important note

To collect the material acquired by the user created failure detection time commands as the snapshot log, you need to add the collection destination of that material to the snapshot log collection destination. For details about addition of the snapshot log collection destination, see *3.3.3(3) Customizing the snapshot log collection destination*.

3.3.3 Settings for collecting snapshot logs (Systems for executing J2EE applications)

This subsection describes the settings for collecting snapshot logs. You can change the settings of the files that are collected as snapshot log and storage destination of the collected snapshot logs.

(1) Data that you can collect for each snapshot log collection timing

The following table describes the settings to change the method to collect the snapshot logs according to the collection timing.

Table 3–8: Settings for changing the snapshot log collection (for a system on which J2EE applications runs)

Category	Collection timing	Settings required to change the default settings
Automatic collection#	Immediately before the automatic termination when the logical server fails	• Settings of failure detection time commands • Customization of the collection destination of snapshot logs • Timeout settings for snapshot log collection
	Immediately before the automatic restart when the J2EE server fails	
	Immediately before the J2EE server is restarted manually in a batch	• Customization of the collection destination of snapshot logs
Collect at the specified timing	When the management command (`mngsvrutil`) of the Management Server is executed to collect the snapshot log	

#

You can change the timing of collection of snapshot logs to before terminating the logical server or before restarting the J2EE server in the `com.cosminexus.mngsvr.snapshot.collect.point` key of `mserver.properties`. According to the default settings, the snapshot log is collected when the logical server stops.

The following points describe the data that you can collect according to the collection timing.

(a) Automatic collection

When the snapshot log is automatically collected at either of the following times, the Management Server executes the failure detection time commands, and the material such as thread dumps and trace based performance analysis are acquired:

- Immediately before automatic termination when the logical server fails (default settings)
- Immediately before the automatic restart when the J2EE server fails

You can collect the material collected using the failure detection time commands as the snapshot log. For details about changing operations and settings of the failure detection time commands, see *3.3.1 Data acquisition settings using failure detection time commands (Systems for executing J2EE applications)*.

(b) Any time collection

You can collect the snapshot log at any time only when thread dump files and user dumps (in Windows) or core dumps (in UNIX) are output. When collecting the snapshot log at any time, thread dump files and user dumps or core dumps are not output when the command is executed.

To collect the snapshot log at any time by executing the operation management commands (`mngsvrutil`) of the Management Server, you can specify the type of collection destination (type 1 or type 2) when executing the command

to collect the snapshot log. When collecting at other times, all files defined as collection targets of type 1 and type 2 are collected.

(2) Files that can be collected in a snapshot log

The troubleshooting data is classified into primary delivery data and secondary delivery data according to the time at which the data is sent to the maintenance personnel. You can collect the primary and secondary delivery data in snapshot log. For each type of data, see *2.3.3(2) Data that can be collected as snapshot log*.

Specify the file to be collected as primary delivery data in the snapshot log collection definition file (`snapshotlog.conf`) of primary delivery data. When collecting the snapshot log, if you specify the argument snapshot 1 in the `mngsvrutil` command, files specified in `snapshotlog.conf` are collected. Specify the files to be collected as secondary material in the snapshot log collection target definition file (snapshotlog.2.conf) of secondary delivery data. When collecting the snapshot log, if you specify the argument snapshot 2 in the `mngsvrutil` command, files specified in `snapshotlog.conf` and `snapshotlog.2.conf` are collected.

When you want to collect the files that are not defined as the snapshot log collection destination by default as the snapshot log, add the output destination of those files to the snapshot log collection target definition file. The following settings may also be required depending on the material:

- **Settings to acquire the material before starting an operation**
  You need to make the settings for collecting the material before starting the operation.
- **Acquiring the material using commands**
  Before collecting as the snapshot log, you need to execute commands to acquire the material targeted for collection. For details about data acquisition, see *4. Output Destinations and Output Methods of Data Required for Troubleshooting*.

(Examples)

- In Windows
  Before starting the operation, you need to set the environment variable (`CJMEMDUMP_PATH`) for user dumps. If you set the environment variable (`CJMEMDUMP_PATH`), you need to change the collection destination of the snapshot log.
- In UNIX
  Before starting the operation, you need to set the core file size for core dumps. We recommend that you specify the settings for collection before starting the operation, and then obtain core dumps when an error is detected by using the user-created failure detection time commands, and collect the core dumps as the snapshot log. The default output destination of core dumps is specified in the default snapshot log collection target definition file.

For details about settings of the collection to be implemented before starting the operation, see *3.3.15 Settings for Collecting a User Dump* or *3.3.16 Settings for Acquiring a Core Dump*. For details about the failure detection time commands created by user, see *3.3.1 Data acquisition settings using failure detection time commands (Systems for executing J2EE applications)*.

For specifying `snapshotlog.conf` and `snapshotlog.2.conf`, see *3.3.3(3) Customizing the snapshot log collection destination*. For details about the data to be collected as the primary delivery data and the secondary delivery data as per the default settings, see *Appendix A. List of Snapshot Logs to Be Collected*.

(3) Customizing the snapshot log collection destination

You can customize the snapshot log collection target definition file as the snapshot log collection destination. You can also use `adminagent.properties` to specify the number of files of the snapshot log for each logical server.

For details about the file, see *10.2.1 Definition file for snapshot log collection* in the *uCosminexus Application Server Definition Reference Guide*.

(a) Specifying the snapshot log collection destination

Edit the snapshot log collection target definition file and specify the collection destination for the snapshot log. The storage location of the file listing the files targeted as the snapshot log collection target definition file is as follows:

- **In Windows**
  Cosminexus-installation-directory`\manager\config\snapshotlog.conf`
  Cosminexus-installation-directory`\manager\config\snapshotlog.2.conf`
- **In UNIX**
  `/opt/Cosminexus/manager/config/snapshotlog.conf`
  `/opt/Cosminexus/manager/config/snapshotlog.2.conf`

Specify the destination directory to be collected files as the Primary delivery data in snapshotlog.conf. Also, specify the destination directory to be collected files as the secondary delivery data in snapshotlog.2.conf.

In snapshot log collection target definition file, you can use a variable for the collection path. For example, if "`${cosminexus.home}/manager/log/.+`" (period (.) represents optional character, and plus (+) represents more than 1 time) is specified for the snapshot log collection target definition file by using "`${cosminexus.home}`" variable that represents Cosminexus installation directory, all files under *Cosminexus-installation-directory*`/manager\log` (in Windows) or `/opt/Cosminexus/manager/`*log directory* (in UNIX) are collected. Do not include the dollar sign (`$`) in the variable value of the snapshot log collection target definition file.

(b) Specifying the number of snapshot log files

You can use the following keys of `adminagent.properties` to change the number of snapshot log files for each logical server. Default setting is 10. For `adminagent.properties`, see *8.2.1 adminagent.properties (Administration Agent property file)* in the *uCosminexus Application Server Definition Reference Guide*.

- `adminagent.snapshotlog.num_snapshots`
  Specify the number of snapshot log files to be collected as the primary delivery data for each logical server.
- `adminagent.snapshotlog.listfile.2.num_snapshots`
  Specify the number of snapshot log files to be collected as the secondary delivery data for each logical server.

(c) Specifying storage destination for the snapshot log

You can change the storage destination of the information that is automatically collected by snapshot log collection by using the `adminagent.snapshotlog.log_dir` key of `adminagent.properties`. By default, the information is stored at the following locations:

- **In Windows**
  Log-output-folder-of-Manager`\snapshot`
- **In UNIX**
  Log-output-directory-of-Manager`/snapshot`

For `adminagent.properties`, see *8.2.1 adminagent.properties (Administration Agent property file)* in the *uCosminexus Application Server Definition Reference Guide*.

(4) Setting snapshot log collection timeout

You can set the timeout for the processes that are executed when collecting the snapshot log. Set the timeout in `mserver.properties`. The following table shows the processes that are executed when collecting snapshot logs and their timeouts.

Table 3–9: Processes that are executed when collecting snapshot logs and timeout settings

Process	Timeout settings (`mserver.properties` key)	Explanation
Execution of the failure detection time command provided by the system	`com.cosminexus.mngsvr.sys_cmd.abnormal_end.timeout`	Shows the time that the program waits till each of the following processes ends: • Failure detection time command provided by the system • Collection of trace based performance analysis • If the command or collection of trace based performance analysis does not end even after the specified time has elapsed, ignore the executed command or collection of trace based performance analysis and continue the process.
Execution of the failure detection time command created by the user	`com.cosminexus.mngsvr.usr_cmd.abnormal_end.timeout`	Shows the time the program waits till the user created failure detection time command ends. If the command does not end even after the specified time has elapsed, ignore the executed command and continue the process.
snapshot log (primary delivery data) collection	`com.cosminexus.mngsvr.snapshot.auto_collect.timeout`	Shows the time the program waits till the collection of primary and secondary delivery data ends. If the collection does not end even after the specified time has elapsed, a service request for canceling the snap log collection is sent from Management Server to Administration Agent, and a KEOS20052-E message is output in the Management Server log.
snapshot log (secondary delivery data) collection		

For `mserver.properties`, see *8.2.6 mserver.properties (Management Server environment settings file)* in the *uCosminexus Application Server Definition Reference Guide*.

3.3.4 Settings for collecting snapshot log (Systems for executing batch applications)

This subsection describes the settings for collecting the snapshot log when using a batch server. You can change the settings of the files that are collected as snapshot log and storage destination of the collected snapshot logs. The following table describes the settings required for changing the method for collecting snapshot logs according to the collection timing. Other details are same as that of a J2EE server. For details, see *3.3.3 Settings for collecting snapshot logs (Systems for executing J2EE applications)*.

Table 3–10: Settings for changing the snapshot log collection (for the system on which batch applications run)

Category	Collection timing	Settings required to change the default settings
Automatic collection#	Immediately before the automatic termination when the logical server fails	• Settings of failure detection time commands

Category	Collection timing	Settings required to change the default settings
	Immediately before the automatic restart when the batch server fails.	• Customization of the collection destination of snapshot logs • Setting snapshot log collection timeout
	Immediately before the batch server is restarted manually in a batch.	• Customization of the collection destination of snapshot logs
Collect at the specified timing	When the management command (`mngsvrutil`) of the Management Server is executed to collect the snapshot log	

#

You can change the timing of collection of snapshot logs to before terminating the logical server or before restarting the batch server in the `com.cosminexus.mngsvr.snapshot.collect.point` key of `mserver.properties`. According to the default settings, the snapshot log is collected when the logical server stops.

3.3.5 Settings for acquiring the Management Server log

This subsection describes the settings for acquiring logs output by the Management Server.

The following is the output destination directory of the Management Server. You can change the output destination of the Management Server by using `manager.cfg` (Manager log settings file).

- In Windows
  Cosminexus-installation-directory`\manager\log`
- In UNIX
  `/opt/Cosminexus/manager/log`

Specify the following key in `mserver.properties` (Management Server environment setup file) to change the output level or the number of log files of the Management Server log.

- `com.cosminexus.mngsvr.log.level`
  Specify the output level of the Management Server log.
- `com.cosminexus.mngsvr.log.rotate`
  Specify the number of files of the Management Server log.
- `com.cosminexus.mngsvr.log.size`
  Specify the file size of the Management Server log.

For details about `mserver.properties` and the keys, see *8.2.6 mserver.properties (Management Server environment settings file)* in the *uCosminexus Application Server Definition Reference Guide*.

Important note

In UNIX, the Manager log output directory is created automatically by specifying 777 (rwxrwxrwx) in the access permission. To specify a previously manually-created directory in the log output destination, set up 777 (rwxrwxrwx) in the directory access permission. If 777 (rwxrwxrwx) is not set up in the access permission of the directory specified in the log output destination, the log might not be output during the execution of the command.

3.3.6 Settings for Acquiring the J2EE Server Log

This subsection describes the items that you can set up for acquiring J2EE server logs.

You can change the log output destination, log size, the log level, the switching method of the log output destination files, and the switching time of the log output destination files for the J2EE server logs. The following table describes the items that you can change and the corresponding parameters of the Easy Setup definition file.

Table 3–11: Settings for acquiring J2EE server log

Items	Corresponding parameter of the Easy Setup definition file
Log output destination	`ejb.server.log.directory` in the `<configuration>` tag on a logical J2EE server (j2ee-server)
Log size	**Number of log files** `ejbserver.logger.channels.define.`*channel-name*`.filenum` in the `<configuration>` tag on a logical J2EE server (j2ee-server) **Maximum size for each log file** `ejbserver.logger.channels.define.`*channel-name*`.filesize` in the `<configuration>` tag on a logical J2EE server (j2ee-server)
Log level	`ejbserver.logger.enabled.*` in the `<configuration>` tag on a logical J2EE server (j2ee-server)
Switching method of the log output destination files	`ejbserver.logger.rotationStyle` in the `<configuration>` tag of the logical J2EE server (`j2ee-server`)
Switching time of the log output destination files	`ejbserver.logger.rotationTime` in the `<configuration>` tag of the logical J2EE server (`j2ee-server`)

If you specify `true` (default value) in the `ejbserver.logger.systemlog.enabled` parameter in the `<configuration>` tag of logical J2EE server (j2ee-server) or if the specification of this parameter is omitted, the message indicating startup, termination, and abnormal termination of the J2EE server will output to an event log (in UNIX, `syslog`).

Notes (in UNIX)

To output messages related to J2EE server start, stop, and abnormal termination, to syslog, it is necessary to set the priority for the facility daemon to info or debug in the syslog settings. Moreover, the log output destination and log file name of `syslog` depend on the settings of `syslog`.

For details about the syslog and its settings, see the description on `syslogd` or `syslog.conf` in the manual provided with OS.

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*.

(1) Changing log output destination

Specify the log output destination directory in the Easy Setup definition file to change the output destination of the J2EE server log.

Changing log output destination

The default log output destination is as follows:

- In Windows
  working-directory`\ejb\`*server-name*`\logs`

- In UNIX

  working-directory/ejb/*server-name*/logs

Note that the default directory path of the working directory is *Cosminexus-installation-directory*\CC\server\public (in Windows) or /opt/Cosminexus/CC/server/public (in UNIX).

You can change the output destination of the working directory and J2EE server log if you specify following parameters in the <configuration> tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file:

- ejb.public.directory

  Specify the path of the working directory of the J2EE server.

- ejb.server.log.directory

  Specify the output destination directory of the J2EE server log.

Setup example (For definition of the physical tier)

- In Windows

```
<configuration>
 <logical-server-type>j2ee-server</logical-server-type>
 <param>
  <param-name>ejb.server.log.directory</param-name>
  <param-value>C:\CClogs\server\MyServer</param-value>
 </param>
 :
</configuration>
```

- In UNIX

```
<configuration>
 <logical-server-type>j2ee-server</logical-server-type>
 <param>
  <param-name>ejb.server.log.directory</param-name>
  <param-value>/CClogs/server/MyServer</param-value>
 </param>
 :
</configuration>
```

Current directory

The following is the current directory where the log output destination is specified by a relative path:

- In Windows

  working-directory\ejb*server-name*

- In UNIX

  working-directory/ejb/*server-name*

Notes

- After changing the log output destination, create a log output destination directory before starting the J2EE server.

  If there is no log output destination directory after the change, the message KDJE40024-E will output and an abnormal termination will occur when starting the J2EE server. Moreover, the messages KDJE37209-E, KDJE37210-E, and KDJE37211-E will output and an abnormal termination will occur when executing the Management Server.

- For starting multiple J2EE servers on the same host, keep the directory name unique for each server including the server name in the directory, so that the log output destination does not have the same directory. When specifying the same directory for a parameter value, the operation is not guaranteed.

- After changing the log output destination, if you want to output the log to a directory other than the working directory, you cannot delete the log file if the server is not set up. If you want to delete the log file, delete it manually.
- Note that if you set up the JavaVM maintenance information and the output destination of the GC log in the JavaVM startup parameter, the settings of JavaVM startup parameter are given priority, even when the log output destination is set up in the Easy Setup definition file.

  A JavaVM startup parameter is defined in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file. The specified contents of the JavaVM startup parameter are as follows:

`<param-name>` tag

`add.jvm.arg`

`<param-value>` tag

`XX:HitachiJavaLog:`*<JavaVM-Maintenance-information* and *GC-log-output-destination>*

When you specify the JavaVM startup parameter, the JavaVM maintenance information and the log file of GC is output to the directory set up for the JavaVM maintenance information and for the output destination of GC log respectively.

- You cannot specify the path including UNC name in the log output destination.

(2) Change log size

Set up the number of log files and the maximum size for each log file in the Easy Setup definition file to change the log size of the J2EE server.

Changing the number of log files

Specify the number of J2EE server log files in the `ejbserver.logger.channels.define.`*channel-name*`.filenum` parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server).

Setup example (For definition of the physical tier)

```
<configuration>
 <logical-server-type>j2ee-server</logical-server-type>
 <param>
  <param-name>ejbserver.logger.channels.define.MessageLogFile.filenum</
param-name>
  <param-value>3</param-value>
 </param>
 ...
</configuration>
```

Changing the maximum file size for each log file

Specify the maximum file size (unit: bytes) for each file of the J2EE server log files in the `ejbserver.logger.channels.define.`*channel-name*`.filesize` parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server).

Setup example (For definition of physical tier)

```
<configuration>
 <logical-server-type>j2ee-server</logical-server-type>
 <param>
  <param-name>ejbserver.logger.channels.define.MessageLogFile.filesize<
/param-name>
  <param-value>2097152</param-value>
 </param>
```

```
 ...
</configuration>
```

(3) Changing the log level

A J2EE server log level indicates the importance of a log. In log levels, there are four levels; *Error*, *Warning*, *Information*, and *Debug*. If you set up a log level, the log of the level that is set up will be output. By default, you can acquire only an Error level log and use this log as it is.

Specify the log level in the `ejbserver.logger.enabled.*` parameter in the `<configuration>` tag of the logical J2EE server in the Easy Setup definition file. Set up the level names *Error*, *Warning*, *Information*, and *Debug* as one or in multiple character strings in the `<param-value>` tag of `ejbserver.logger.enabled.*`. When you set up multiple log levels, the character string of the level name will be demarcated using comma (,).

Setup example (For definition of the physical tier)

1.

```
<configuration>
 <logical-server-type>j2ee-server</logical-server-type>
 <param>
  <param-name>ejbserver.logger.enabled.*</param-name>
  <param-value>Error</param-value>
 </param>
 ...
</configuration>
```

2.

```
<configuration>
 <logical-server-type>j2ee-server</logical-server-type>
 <param>
  <param-name>ejbserver.logger.enabled.*</param-name>
  <param-value>Error,Warning</param-value>
 </param>
 ...
</configuration>
```

3.

```
<configuration>
 <logical-server-type>j2ee-server</logical-server-type>
 <param>
  <param-name>ejbserver.logger.enabled.*</param-name>
  <param-value>Error,Warning,Information</param-value>
 </param>
 ...
</configuration>
```

4.

```
<configuration>
 <logical-server-type>j2ee-server</logical-server-type>
 <param>
  <param-name>ejbserver.logger.enabled.*</param-name>
  <param-value>Error,Warning,Information,Debug</param-value>
 </param>
```

```
  ...
</configuration>
```

Notes

- The number of logs that you can acquire increases in the order of 1, 2, 3, 4 as described in examples. If you acquire the log after setting up multiple log levels, the performance will decrease and the switching of log file will occur frequently.
- If you set up a blank value or a character string other than *Error*, *Warning*, *Information*, and *Debug* for the level name, the message `KDJE90009-W` will output and the `Error` level log will be acquired.

Hitachi recommended settings for log level

Hitachi recommends following settings for a log level:

- For normal operation
  Specify *Error* in the level name.
- For normal operation (verbose)
  Specify *Error, Warning* in the level name to acquire more detailed information than that of the normal operation.
- For test
  Specify *Error, Warning, Information* in the level name.
- At failure detection time
  Specify *Failure, Warning, Information, Debug* in the level name.

3.3.7 Settings for Acquiring the Batch Server Log

You can change the log output destination, log size, and log level for batch server log. The following table describes the items that can be changed and the parameters of the Easy Setup definition file corresponding to these items. Furthermore, set up these parameters in the user property for the batch server in the Easy Setup definition file.

Table 3–12: Settings for acquiring batch server log

Items	Corresponding parameters of the Easy Setup definition file
Log output destination	`ejb.server.log.directory` in the `<configuration>` tag on a logical J2EE server (j2ee-server)
Log size	**Number of log files** `ejbserver.logger.channels.define.`*channel-name*`.filenum` in the `<configuration>` tag on a logical J2EE server (j2ee-server) **Maximum size for each log file** `ejbserver.logger.channels.define.`*channel-name*`.filesize` in the `<configuration>` tag on a logical J2EE server (j2ee-server)
Log level	`ejbserver.logger.enabled.*` in the `<configuration>` tag on a logical J2EE server (j2ee-server)

If you specify `true` (default value) in the `ejbserver.logger.systemlog.enabled` parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server) or if you skip specification of this parameter, the message indicating startup, termination, and abnormal termination of a batch server is output to the event log (in UNIX, `syslog`).

For details about the above-mentioned respective settings, see *3.3.6 Settings for Acquiring the J2EE Server Log*. At this time, substitute *J2EE server* to *batch server*.

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

Notes (in UNIX)

To output messages related to batch server start, stop, and abnormal termination, to syslog, it is necessary to set the priority for the facility daemon to info or debug in the syslog settings. Moreover, the log output destination and log file name of `syslog` depend on the setting of `syslog`.

For details about the `syslog` and its settings, see the description of `syslogd` or `syslog.conf` in the manual provided with OS.

3.3.8 Settings for Acquiring the Web Server Log

This subsection describes the items that you can set up for acquiring Web server logs. The Web server set up for acquiring the logs is Cosminexus HTTP Server.

The logs output in the Web server are the Error logs, Access logs and the Request logs. For details about the output logs, see the *uCosminexus Application Server HTTP Server User Guide*. Among the output logs of the Web server, you can change the output destination or the output method of logs in the Easy Setup definition file for the Error log, Access log, and Request log.

The following table describes the settings of the items that you can change for acquiring the Web server log, and the corresponding parameters of the Easy Setup definition file.

Table 3–13: Settings for acquiring Web server log

Log	Items	Corresponding parameters of the Easy Setup definition file
Error log	Level of error log to be output	`LogLevel` in the `<configuration>` tag on a logical Web server (web-server)
	Output method of the error log	`HttpsdErrorMethod` in the `<configuration>` tag on a logical Web server (web-server)
	Output destination directory for error log	`HttpsdErrorLogFileDir` in the `<configuration>` tag on a logical Web server (web-server)
Access log	Output method of the access log	`HttpsdCustomMethod` in the `<configuration>` tag on a logical Web server (web-server)
	Output destination directory for access log	`HttpsdCustomLogFileDir` in the `<configuration>` tag on a logical Web server (web-server)
	Format of the access log to be output	`HttpsdCustomlogFormat` in the `<configuration>` tag on a logical Web server (web-server)
Request log	Trace extraction availability	`HWSRequestLogLevel` in the `<configuration>` tag on a logical Web server (web-server)
	Output method of the request log	`HttpsdRequestMethod` in the `<configuration>` tag on a logical Web server (web-server)
	Output destination directory of request log	`HttpsdRequestLogFileDir` in the `<configuration>` tag on a logical Web server (web-server)

Moreover, you can specify the unit of time and the output time for Error log, Access log, and Request log in the `HWSLogTimeVerbose` parameter in the `<configuration>` tag of logical Web server (web-server).

Notes

When checking the Web server operation by using Management Server, to output the operation check log apart from the usual log (access log), you must make settings in the Easy Setup definition file. Set the `AppendDirectives` and `HttpsdCustomlogFormat` parameters by specifying `item` in the `SetBy` parameter in the `<configuration>` tag of the logical Web server (web-server).

The `AppendDirectives` and `HttpsdCustomlogFormat` parameters are set as explained in the examples below. The example explains how to collect the operation check log by using the wraparound method. Change the description of the `CustomLog` directive in the `AppendDirectives` parameter according to the log output method.

Example of AppendDirectives parameter settings

In Windows

```
<param>
  <param-name>AppendDirectives</param-name>
  <param-value>
<![CDATA[
SetEnvIf Remote_Addr ^127\.0\.0\.1$ Env_ManagerHealthCheck
CustomLog "|\"\"Cosminexus installation directory/httpsd/sbin/rotatelog
s2.exe\" \"Cosminexus-installation-directory/httpsd/servers/HWS_ actual
-server-name-of-the-logical-Web-server/logs/access_manager\" 8192 5\""
hws_std env=Env_ManagerHealthCheck
]]>
  </param-value>
</param>
```

In UNIX

```
<param>
  <param-name>AppendDirectives</param-name>
  <param-value>
<![CDATA[
SetEnvIf Remote_Addr ^127\.0\.0\.1$ Env_ManagerHealthCheck
CustomLog "|/opt/hitachi/httpsd/sbin/rotatelogs2 \"/opt/hitachi/httpsd/
servers/HWS_actual-server-name-of-the-
logical-Web-server/logs/access_manager\" 8192 5" hws_std env=Env_Manage
rHealthCheck
]]>
  </param-value>
</param>
```

Example of HttpsdCustomlogFormat parameter settings

```
<param>
  <param-name>HttpsdCustomlogFormat</param-name>
  <param-value>hws_std env=!Env_ManagerHealthCheck</param-value>
</param>
```

3.3.9 Settings for acquiring the NIO HTTP server log

This subsection describes the items that can be set up for acquiring the NIO HTTP server log.

The following table describes the settings that you can change for acquiring the NIO HTTP server log and parameters of the Easy Setup definition file corresponding to the items.

Table 3–14: Settings for acquiring the NIO HTTP server log

Log or trace	Items	Corresponding parameters of the Easy Setup definition file
Access log	Form of the access log	`ejbserver.logger.access_log.nio_http.format` in the `<configuration>` tag on a logical J2EE server (j2ee-server)#
	File size of the access log	`ejbserver.logger.channels.define.NIOHTTPAccessLogFile.filesize` in the `<configuration>` tag on a logical J2EE server (j2ee-server)
	Number of files of access log	`ejbserver.logger.channels.define.NIOHTTPAccessLogFile.filenum` 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 about acquiring the trace based performance analysis files, see *7. Performance Analysis by Using Trace Based Performance Analysis*.

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 NIO HTTP server, see *6.11.2 Customizing the access log of the NIO HTTP server* in the *uCosminexus Application Server Web Container Functionality Guide*.

3.3.10 Settings for Acquiring the Cosminexus Manager Log

In addition to separately acquiring the logs of the Administration Agent, Management Agent, and Management Server you can collectively acquire them as an integrated log. You can acquire integrated message log files, integrated trace log files, or command maintenance log files as an integrated log. For details about the logs that you can acquire as an integrated log, see *4.3 Application Server log (Systems for executing J2EE applications)*.

This subsection describes the changes in settings of integrated logs. Set up the integrated log in `manager.cfg`. The location of the `manager.cfg` file is as follows:

- **In Windows**
  Cosminexus-installation-directory`\manager\config\manager.cfg`
- **In UNIX**
  `/opt/Cosminexus/manager/config/manager.cfg`

Set the following keys in `manager.cfg` to change the settings of the integrated log.

- `com.cosminexus.manager.log.dir`
  Specify the output destination of the integrated log. For default settings, output to the following location:
  - In Windows
    Cosminexus-installation-directory`\manager\log`
  - In UNIX
    `/opt/Cosminexus/manager/log`
- `com.cosminexus.manager.messagelog.size`
  Specify the maximum size for each integrated message log file.

- `com.cosminexus.manager.messagelog.fnum`

  Specify the number of integrated message log files.
- `com.cosminexus.manager.messagelog.style`

  Specify how to switch the output destination of the integrated message log file. If `SHIFT` (shift mode) is specified, `com.cosminexus.manager.messagelog.time` is enabled.
- `com.cosminexus.manager.messagelog.time`

  Specify the time at which the output destination of the integrated message log file will be switched.
- `com.cosminexus.manager.tracelog.size`

  Specify the maximum size for each integrated trace log file.
- `com.cosminexus.manager.tracelog.fnum`

  Specify the number of integrated trace log files.
- `com.cosminexus.manager.tracelog.style`

  Specify how to switch the output destination of the integrated trace log file. If `SHIFT` (shift mode) is specified, `com.cosminexus.manager.tracelog.time` is enabled.
- `com.cosminexus.manager.tracelog.time`

  Specify the time at which the output destination of the integrated trace log file will be switched.
- `com.cosminexus.manager.cmdtracelog.size`

  Specify the maximum size for each file of command maintenance log files.
- `com.cosminexus.manager.cmdtracelog.fnum`

  Specify the number of command maintenance log files.
- `com.cosminexus.manager.log.compatible`

  Specify whether to output the Administration Agent and Management Server logs separately. According to the default settings, individual logs are output concurrently with the integrated log. Specify false if you do not want to output individual logs.

3.3.11 Settings for Acquiring the Resource Adapter Logs

This subsection describes the settings to acquire the resource adapter logs. The following two settings are required to acquire resource adapter logs.

- **Settings for application of the log output**

  Set up the availability of output of the log for each resource adaptor by the server management commands. Acquire Hitachi Connector Property file by the `cjgetrarprop` command and specify `true` for LogEnabled in the `<property>` tag. After editing the file, use the `cjsetresprop` command to apply the edited contents. Note that for defining the property before deploying the resource adapter, use the `cjgetrarprop` and `cjsetrarprop` command.

  For details about Hitachi Connector Property file, see *4.1 HITACHI Connector Property file* in the *uCosminexus Application Server Application and Resource Definition Reference Guide*. For details about the above commands, see the *uCosminexus Application Server Command Reference Guide*. For details about the operations of the server management commands, see *3. Basic Operations of Server Management Commands* in the *uCosminexus Application Server Application Setup Guide*.
- **Setting the size, number, and the level of logs**

  Set up the size and number of resource adapter logs in the following parameter in the `<configuration>` tag of logical J2EE server (j2ee-server) in the Easy Setup definition file.

- `ejbserver.connector.logwriter.filesize`

  Set up the maximum size for each resource adapter log file. (unit: byte).

- `ejbserver.connector.logwriter.filenum`

  Specify the number of resource adapter log files.

Moreover, the log output level of the J2EE server (log level) is considered as the log level of the resource adapter. For details about settings of the log level of the J2EE server, see *3.3.6(3) Changing the log level*.

Important note

- For the CJMSP resource adapters, methods to set up log size, number of files, and log levels differ from the other resource adapters. For details, see the subsection *3.3.13 Settings for collecting Cosminexus JMS Provider logs*.
- If the sum of the length of the log output destination directory of the resource adapter and the length of the resource adapter display name exceeds the path length restrictions for the OS, the initialization of the resource adapter log fails, the KDJE90002-E message is output, and then the server stops. This occurs in the cases such as when the log file is locked by another process.

  Use one of the following steps to ensure that the resource adapter log does not exceed the maximum path length. For details, see the *uCosminexus Application Server System Setup and Operation Guide*.

  - Change the length of the resource adapter display name

  - Change the log output destination directory or work directory

  Also, do not lock the log file with another process.
- If the initialization of the resource adapter log fails, the KDJE90002-E message is output and then the J2EE server stops. This occurs when the sum of the path length of the log output destination directory of the resource adapter and the length of the resource adapter display name exceeds the path length restrictions for the OS, or in the cases such as when the log file is locked by another process.

  Change the length of the resource adapter display name, or change the log output destination directory or work directory to ensure that the resource adapter log does not exceed the maximum path length. Also, do not lock the log file with another process.

3.3.12 Settings for Acquiring the Cosminexus TPBroker Log

This subsection describes the changes in the output destination of trace files, the number of files, and entries of Cosminexus TPBroker.

The default output destination of the trace files of Cosminexus TPBroker is as follows:

For trace information of a J2EE server

- In Windows

  working-directory`\ejb\`*server-name*`\logs\TPB\logj`
- In UNIX

  working-directory`/ejb/`*server-name*`/logs/TPB/logj`

For trace information of server management commands

- In Windows

  Cosminexus-installation-directory`\CC\admin\logs\TPB\logj`
- In UNIX

/opt/Cosminexus/CC/admin/logs/TPB/logj

For the trace information of the EJB client applications

- In Windows

  EJB-client-log-output-directory#\system\TPB\logj

- In UNIX

  EJB-client-log-output-directory#/system/TPB/logj

#: For details on the output destination of the EJB client log output directory, see *4.5.2 Output Destination of the EJB Client Application System Log*.

For the CTM trace information

- In Windows

  Product-installation-directory\TPB\log

  Product-installation-directory\TPB\logj

- In UNIX

  /opt/Cosminexus/TPB/log

  /opt/Cosminexus/TPB/logj

Command used in batch applications

- In Windows

  Product-installation-directory\TPB\log

- In UNIX

  /opt/Cosminexus/TPB/log

The following table describes the items that can be changed, the corresponding parameters of the Easy Setup definition file or the corresponding user definition files and keys.

Table 3–15: Settings for acquiring Cosminexus TPBroker log

Items	Category	Corresponding parameters of the Easy Setup definition file or user definition files and keys
Output destination of trace file	J2EE server	ejb.server.log.directory in the <configuration> tag on a logical J2EE server (j2ee-server) or vbroker.orb.htc.tracePath in the <configuration> tag on a logical J2EE server (j2ee-server)
	Server management commands	-Dejbserver.log.directory option of the USRCONF_JVM_ARGS key of usrconf.bat (in Windows), or usrconf (in UNIX) or vbroker.orb.htc.tracePath key# of usrconf.properties for server management commands
	EJB client applications	• For the cjclstartap command, the vbroker.orb.htc.tracePath key in usrconf.properties for the Java applications • For the vbj command, the vbroker.orb.htc.tracePath key in the JavaVM system properties specified with the vbj command

Items	Category	Corresponding parameters of the Easy Setup definition file or user definition files and keys
	CTM	• For the global CORBA Naming Service, set up the environment variable HVI_TRACEPATH at the location where the logical server type of adminagent.xml (Administration Agent configuration file) is Naming Service. • For the CTM domain manager, set up the environment variable HVI_TRACEPATH in user.env.variable in the <configuration> tag of the logical CTM domain manager (ctm-domain-manager). • For CTM, set up the environment variable HVI_TRACEPATH in user.env.variable in the <configuration> tag of the logical CTM (component-transaction-monitor).
	Command used in batch applications	Environment variable HVI_TRACEPATH
Number of trace files	J2EE server	vbroker.orb.htc.comt.fileCount in the <configuration> tag on a logical J2EE server (j2ee-server)
	EJB client applications	• For the cjclstartap command, the vbroker.orb.htc.comt.fileCount key in usrconf.properties for the Java applications • For the vbj command, the vbroker.orb.htc.comt.fileCount key in the JavaVM system properties specified with the vbj command
	CTM	• For the global CORBA Naming Service, set up the environment variable HVI_COMTFILECOUNT at the location where the logical server type of adminagent.xml (Administration Agent configuration file) is Naming Service. • For the CTM domain manager, set up the environment variable HVI_COMTFILECOUNT in user.env.variable in the <configuration> tag of the logical CTM domain manager (ctm-domain-manager). • For CTM, set up the environment variable HVI_COMTFILECOUNT in user.env.variable in the <configuration> tag of the logical CTM (component-transaction-monitor).
Number of trace file entries	J2EE server	vbroker.orb.htc.comt.entryCount in the <configuration> tag on a logical J2EE server (j2ee-server)
	EJB client applications	• For the cjclstartap command, the vbroker.orb.htc.comt.entryCount key in usrconf.properties for the Java applications • For the vbj command, the vbroker.orb.htc.comt.entryCount key in the JavaVM system properties specified with the vbj command
	CTM	• For the global CORBA Naming Service, set up the environment variable HVI_COMTENTRYCOUNT at the location where the logical server type of adminagent.xml (Administration Agent setup file) is Naming Service. • For the CTM domain manager, set up the environment variable HVI_COMTENTRYCOUNT in user.env.variable in the <configuration> tag of the logical CTM domain manager (ctm-domain-manager). • For CTM, set up the environment variable HVI_COMTENTRYCOUNT in user.env.variable in the

Items	Category	Corresponding parameters of the Easy Setup definition file or user definition files and keys
		<configuration> tag of the logical CTM (component-transaction-monitor).
	Command used in batch applications	Environment variable HVI_COMTENTRYCOUNT

#

For details about the usrconf.bat file (in Windows) or usrconf file (in UNIX), and usrconf.properties, see *3.3 Customizing operation settings of server management commands* in the *uCosminexus Application Server Application Setup Guide*. For details about each file and key for the server management commands see the following sections in the *uCosminexus Application Server Definition Reference Guide*:

- *5.1 List of files used in server management commands*
- *5.2.1 usrconf (Option definition file for server management commands for UNIX)*
- *5.2.2 usrconf.bat (Option definition file for server management commands for Windows)*
- *5.2.3 usrconf.properties (System property file for server management commands)*

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

For details on the usrconf.properties file and keys for the Java applications, see *12.2.2 usrconf.properties (User property file for Java applications)* in the *uCosminexus Application Server Definition Reference Guide*.

For details on the JavaVM system properties specified with the vbj command, see *12.2.3 System properties specified in the Java application* in the *uCosminexus Application Server Definition Reference Guide.*

For details on adminagent.xml (Administration Agent setup file), see *8.2.4 adminagent.xml (Administration Agent settings file)* in the *uCosminexus Application Server Definition Reference Guide*.

For details on the communication trace, see the *TPBroker Operation Guide*.

Notes

- To change the output destination of a trace file, you need to create comtrc and mdltrc in advance as the subdirectory of the changed output destination directory of the trace file. If you change the output destination, the trace file is output to comtrc and mdltrc under the changed log output destination directory.
- If you specify both the ejb.server.log.directory and vbroker.orb.htc.tracePath for <param-name> in the <configuration> tag on a logical J2EE server (j2ee-server) in the Easy Setup definition file, the settings of the vbroker.orb.htc.tracePath are given priority.
- If you specify both the -Dejbserver.log.directory option of the USRCONF_JVM_ARGS key of usrconf.bat (in Windows) or usrconf (in UNIX) and the vbroker.orb.htc.tracePath key of usrconf.properties for the server management commands, the settings of the vbroker.orb.htc.tracePath key of usrconf.properties is given priority.
- You cannot change the output destination of the server management commands when operating from the Management Server Remote Management.
- If you specify the number of trace files and entries with the CTM domain manager and CTM, these values are valid not only in the CTM domain manager and the CTM daemon, but also in the processes of the CTM regulator and the ctmstart command. When you increase the number of trace files and entries, note the increase in the disk usage.
- When you increase the number of trace file entries of the communication trace file, note the increase in the memory usage.

3.3.13 Settings for collecting Cosminexus JMS Provider logs

This subsection describes the settings for changing output levels, number of files, and file size of the logs of CJMSP Broker, management command (cjmsicmd), and CJMSP resource adapters that are used by Cosminexus JMS Provider.

With Cosminexus JMS Provider, the following three types of logs are output:

- CJMSP Broker log
- Management command (`cjmsicmd`) log
- CJMSP resource adapter log

The following table describes the default output destination for each of the above logs.

Table 3–16: Default output destination for logs of Cosminexus JMS Provider

Log type	Default output destination

CJMSP Broker log	**In Windows**
	`<CJMSP_HOME>`[#1]`\var\instances\instanceName\log`
	In UNIX
	`<CJMSP_HOME>`[#1]`/var/instances/instanceName/log`

Management command (`cjmsicmd`) log	**In Windows**
	`<CJMSP_HOME>`[#1]`\var\admin\log`
	In UNIX
	`<CJMSP_HOME>`[#1]`/var/admin/log`

CJMSP resource adapter log	**In Windows** *J2EE-server-log-output-directory* `(ejb.server.log.directory)`[#2]`\cjms\Cosminexus_JMS_Provider_RA` **In UNIX** *J2EE-server-log-output-directory* `(ejb.server.log.directory)`[#2]`/cjms/Cosminexus_JMS_Provider_RA`

#1

`<CJMSP_HOME>` indicates the following directory:

In Windows

Cosminexus-installation-directory`\CC\cjmsp`

In UNIX

`/opt/Cosminexus/CC/cjmsp`

#2

J2EE-server-log-output-directory `(ejb.server.log.directory)` is the directory specified in the J2EE server option definition. By default, the following directory is used:

In Windows

Directory-specified-in-ejb.public.directory`\ejb\`*J2EE-server-name*`\logs`

In UNIX

Directory-specified-in-ejb.public.directory`/ejb/`*J2EE-server-name*`/logs`

Note that if the default output destination does not exist, a directory is created when the log is output.

(1) Settings for collecting the CJMSP Broker logs

Among the log acquisition settings for CJMSP Broker, you can change the log output level, number of files, and file size.

The following table describes how to change the settings.

Table 3–17: How to change the log collection settings for CJMSP Broker

Item	Changing method
Log output level	Specify the changes in the following properties of `commonconfig.properties` or `config.properties`: • `broker.logger.MessageLogFile.trace.level`
Number of files	Specify the changes in the following properties of `commonconfig.properties` or `config.properties`: • `broker.logger.MessageLogFile.filenum` • `broker.logger.ExceptionLogFile.filenum`
File size	Specify the changes in the following properties of `commonconfig.properties` or `config.properties`: • `broker.logger.MessageLogFile.filesize` • `broker.logger.ExceptionLogFile.filesize`

Note:

With CJMSP Broker, you can change the log output destination with the `-varhome` option of the `cjmsbroker` command. However, you can only change the log output destination for the `<CJMSP_HOME>\var` directory (in Windows) or `<CJMSP_HOME>/var` directory (in UNIX).

Note that if the specified directory does not exist, the default settings are applied.

For details about the properties, see *6.2.2 commonconfig.properties (CJMSP broker common properties file)* and *6.2.3 config.properties (CJMSP broker individual properties file)* in the *uCosminexus Application Server Definition Reference Guide*.

(2) Settings for collecting the management command (cjmsicmd) log

Among the log collection settings for the management command (`cjmsicmd`), you can change the log output level, output destination, number of files, and file size.

The following table describes how to change the settings.

Table 3–18: How to change the log collection settings for management command (cjmsicmd)

Item	Changing method
Log output level	Specify the changes in the following property of `admin.properties`: • `admin.logger.MessageLogFile.trace.level`
Output destination	Specify the changes in the following properties of `admin.properties`: • `admin.logger.MessageLogFile.filepath` • `admin.logger.ExceptionLogFile.filepath`
Number of files	Specify the changes in the following properties of `admin.properties`: • `admin.logger.MessageLogFile.filenum` • `admin.logger.ExceptionLogFile.filenum`
File size	Specify the changes in the following properties of `admin.properties`: • `admin.logger.MessageLogFile.filesize` • `admin.logger.ExceptionLogFile.filesize`

Note that if the specified directory does not exist, the default settings are applied.

For details about the properties, see *6.2.1 admin.properties (Management command properties file)* in the *uCosminexus Application Server Definition Reference Guide*.

(3) Settings for collecting the CJMSP resource adapter log

Among the log collection settings for the CJMSP resource adapter, you can change the log output level, number of files, and file size.

The following table describes how to change the settings.

Table 3–19: How to change the log collection settings for CJMSP resource adapter

Item	Changing method
Log output level	Specify the following property in <config-property> below <resourceadapter> in the Connector property file: • MsgLogLevel
Number of files	Specify the following properties in <config-property> below <resourceadapter> in the Connector property file: • MsgLogFileNum • ExpLogFileNum
File size	Specify the following properties in <config-property> below <resourceadapter> in the Connector property file: • MsgLogFileSize • ExpLogFileSize

For details about the Connector property file, see *4.1 HITACHI Connector Property file* in the *uCosminexus Application Server Application and Resource Definition Reference Guide*.

3.3.14 Settings for Collecting the OS Statistical Information

This section describes the settings to acquire the OS statistical information when the OS is Windows.

You can acquire the performance data of the system resources using the Windows system monitor. When an error occurs or signs of an error appear, collection of performance data by Windows system monitor begins and after the error occurs the performance data log is stored. For details about the operation of the system monitor, see the manual provided with the OS.

Extract the system monitor logs described in the following table at an interval of 60 seconds. For details about how to set up, see the manual provided with the OS.

Table 3–20: Setting contents of the system monitor

Performance object	Instance	Item name	Description
processor	--	%Processor Time	CPU usage rate (total value excluding thread in non-idle state)
		%Privileged Time	CPU usage rate (kernel mode part)
		%User Time	CPU usage rate (user mode part)
memory	--	Cache Bytes	Number of bytes currently used by file system cache
		Cache Faults/sec	Frequency of extracting the memory from another location or extracting from the disk per second

Performance object	Instance	Item name	Description
		Page Faults/sec	Number of page faults per second
		Transition Faults/sec	Number of faults per second
process	_Total	Handle Count	Total number of handles currently open
		Page Faults/sec	Occurrence rate of page faults
		Private Bytes	Memory Usage (bytes)
		Virtual Bytes	Virtual memory usage (bytes)
		Working Set Bytes	Actual memory usage (bytes)
	cjstartsv	%Processor Time	CPU usage rate (total value excluding thread in non-idle state)
		%Privileged Time	CPU usage rate (kernel mode part)
		%User Time	CPU usage rate (user mode part)
		Page Faults/sec	Occurrence rate of page faults
		Thread Count	Number of threads
		Private Bytes	Memory Usage (bytes)
		Virtual Bytes	Virtual memory usage (bytes)
		Working Set Bytes	Actual memory usage (bytes)

Legend:

--: Not applicable

3.3.15 Settings for Collecting a User Dump

This section describes the settings to acquire user dumps in Windows.

(1) When the task manager or the Windows debug tool is used

When the product hangs up, the user dump becomes necessary as the data required for troubleshooting. To obtain the user dump, use the task manager or the Windows debug tool. For details, see the Microsoft website.

Note that if you want to instantly obtain the user dump when JavaVM terminates abnormally, specify the following settings in the registry before you start the product. The registry settings affect the entire system, so take adequate precautions when you specify the settings. However, `cjstartsv.exe`, `cjstartweb.exe`, `cjclstartap.exe`, and `adminagent.exe` are automatically set up when the product is installed.

- Registry key
  `\\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Windows Error Reporting\LocalDumps`
- Registry values
  - `DumpCount:`*number-of-stored-dumps*
  - `DumpType:2`

(2) Using the cjstopsv command

When acquiring user dumps using the `-fd` option of the `cjstopsv` command, you must set up the output destination directory of the user dump in the environment variable `CJMEMDUMP_PATH`. File name of the user dump is `cjmemdump.dmp`.

The following is a setup example of the environment variable `CJMEMDUMP_PATH`:

Setup example of the environment variable CJMEMDUMP_PATH

```
set CJMEMDUMP_PATH=C:\temp
```

In the above-mentioned example, `cjmemdump.dmp` is created under *C:\temp*.

When specifying the environment variable `CJMEMDUMP_PATH`, note the followings:

- Confirm that there is enough free disk space in the disk where the data is to be stored since the file size of the user dump is more than the required size of real memory of the J2EE server.
- Do not specify the directory that includes Japanese multibyte characters. The output of the user dump might fail.
- Specify the existing directory in the output destination directory of the user dump.

(3) When the user dump is obtained for the forced termination of a logical server

When you use Management Server to set up the system, specify the output destination directory of the user dump in the environment variable `CJMEMDUMP_PATH` to obtain the user dump for the forceful termination of the logical server. If you specify this environment variable, the user dump can be collected beneath the directory specified in this environment variable when the logical J2EE server is terminated forcefully. The user dump file name is `cjmemdump.dmp`. For details, see *4.1.11 Setting environment variables of the system* in the *uCosminexus Application Server System Setup and Operation Guide*.

3.3.16 Settings for Acquiring a Core Dump

This section describes the settings for acquiring core dumps in UNIX.

Important note

Depending on the Linux specifications, the size information in the core file might be invalid.

(1) Setting the maximum size of core files

The maximum size of core files may be `0` depending on the operation environment of the system. In such cases, core dumps of processes cannot be acquired. Therefore, you need to set in advance the maximum size of core files to infinite. To set the maximum size of core files to infinite, specify the option in the JavaVM startup parameter in the Easy Setup definition file or the option definition file, or execute shell commands.

Note that the larger the memory pool size specified in the JavaVM startup parameter in the Easy Setup definition file or option definition file, the larger the core file size will be, so secure sufficient free disk space.

- **Setting up the option in the JavaVM startup parameter in the Easy Setup definition file**

Define the JavaVM startup parameter in the <configuration> tag on a logical J2EE server (j2ee-server) in the Easy Setup definition file. The setting contents of a JavaVM startup parameter are as follows:

<param-name> tag

add.jvm.arg

<param-value> tag

-XX:+HitachiFullCore

Setup example (For definition of the physical tier)

```
<configuration>
 <logical-server-type>j2ee-server</logical-server-type>
 <param>
  <param-name>add.jvm.arg</param-name>
  <param-value>-XX:+HitachiFullCore</param-value>
 </param>
 :
</configuration>
```

- **Setting up the option in the JavaVM startup parameter in the option definition file**

  Define the JavaVM startup parameter in the option definition file.

  Setup example (For the option definition file for Management Server)

  ```
  add.jvm.arg=-XX:+HitachiFullCore
  ```

- **To execute shell commands**

  Execute the shell command and specify the maximum size of the core file as infinite.

  Execution example of Csh (C shell)

  ```
  limit coredumpsize unlimit
  ```

  Execution example of sh (standard shell)

  ```
  ulimit -c unlimited
  ```

Also, in addition to these settings, we recommend that you execute the shell commands to set the maximum file size to unlimited.

- **Example of execution for csh (C shell)**

  ```
  limit filesize unlimit
  ```

- **Example of execution for sh (standard shell)**

  ```
  ulimit -f unlimited
  ```

Reference note

Formula for estimating the core file size

The core file size, generated when a JavaVM process is down, is equal to the amount of virtual memory used. For details on the formula for the virtual memory usage, see the following manuals. The disk, which contains the current directory of the JavaVM process where the core file is generated, must always have free space greater than this core file size.

- For the J2EE application execution platform
  5.3 Estimating memory used for each process in the *uCosminexus Application Server System Design Guide*
- For the batch application execution platform
  6.3 Estimating virtual memory usage in the *uCosminexus Application Server System Design Guide*
- For Management Server and Administration Agent
  5.3 Estimating memory used for each process in the *uCosminexus Application Server System Design Guide* and the standard memory requirements used by Cosminexus Component Container in the *Release Notes*.

(2) Setting the maximum number of files for core files

- You can define the upper limit for the number of core files of a logical J2EE server in the `ejb.server.corefilenum` parameter in the `<configuration>` tag of `j2ee-server` in the Easy Setup definition file. Define the `ejb.server.corefilenum` parameter in the extension parameter of the J2EE server.
  If the total number of core dump files output to *working-directory*`/ejb/`*server-name*`/` during the restart of the `cjstartsv` process exceeds the specified maximum number, the files are deleted in the order of output date, starting from the oldest file.
- You can define the upper limit for the number of core files of Management Server in the `ejb.server.corefilenum` parameter in the option definition file for Management Server.
  If the total number of core dump files output to *Application-Server-installation-directory*`/manager/containers/m/ejb/`*server-name-of-Management-Server*`/` during the restart of the Management Server process exceeds the specified maximum number, the files are deleted in the order of output date, starting from the oldest file.

3.3.17 Settings for Acquiring the JavaVM Material

This section describes the settings for acquiring the following JavaVM material:

- Thread dumps of JavaVM
- JavaVM log (JavaVM log file)
- Event log of the Explicit Memory Management functionality

For details about the JavaVM startup options, see *14. Options for Invoking JavaVM* in the *uCosminexus Application Server Definition Reference Guide*.

By default, the thread dumps of JavaVM are output under the following directory:

- In Windows
  working-directory`\ejb\`*server-name*
- In UNIX
  working-directory`/ejb/`*server-name*

By default, the JavaVM log is output under the following directory:

- In Windows
  working-directory`\ejb\`*server-name*`\logs`
- In UNIX

working-directory/ejb/*server-name*/logs

When you want to change the output destination, you must change the log output directory in the ejb.server.log.directory parameter in the <configuration> tag of a logical J2EE server (j2ee-server) in the Easy Setup definition file before starting JavaVM or J2EE server.

The respective settings for acquiring the thread dumps of JavaVM, JavaVM log (JavaVM log file), and event log of the Explicit Memory Management functionality is as follows:

(1) Settings for Acquiring Thread Dumps of JavaVM

This subsection describes the settings for acquiring thread dumps of JavaVM.

The contents output to the thread dump of JavaVM differs depending on the JavaVM startup option specified in the JavaVM startup parameter in the Easy Setup definition file. Define the JavaVM startup option in the JavaVM startup parameter in <configuration> tag of a logical J2EE server (j2ee-server). The setting contents of a JavaVM startup parameter are as follows:

<param-name> tag

add.jvm.arg

<param-value> tag

JavaVM-startup-option

The following table describes the options to be specified for JavaVM startup options. For details about the options, see *14. Options for Invoking JavaVM* in the *uCosminexus Application Server Definition Reference Guide*.

Table 3–21: Options to be specified for JavaVM startup option (Settings for acquiring the thread dumps of JavaVM)

Option	Description
-XX:+HitachiThreadDump	Output the extended thread dump. By default, the thread dump is output.
-XX:+HitachiThreadDumpToStdout	The extended thread dump is output to the standard output. By default, the thread dump is output.
-XX:+HitachiThreadDumpWithHashCode	The hash code of the thread is output to the thread information. By default, the thread dump is output.
-XX:+HitachiThreadDumpWithCpuTime	The user CPU time and kernel CPU time after the thread starts are output to the thread information. By default, the thread dump is output.
-XX:+HitachiThreadDumpWithBlockCount	The number of times the process was blocked by the thread and the number of times the process is in the pending state is output to the thread information. By default, the thread dump is output.
-XX:+HitachiOutOfMemoryAbort -XX:+HitachiOutOfMemoryAbortThreadDump	In either of the settings, the thread dump is output when forced termination is performed due to OutOfMemoryError. If the C heap in the J2SE class library is insufficient, and the C heap in the processing of JavaVM is insufficient, the thread dump is not output
-XX:+HitachiOutOfMemoryAbortThreadDumpWithJHeapProf	A class-wise statistical information is output in the thread dump that is output when forced termination is performed due to OutOfMemoryError. Nothing is output by default.

To change the output destination of the thread dump file, specify the output destination in the environment variable JAVACOREDIR. For details on the environment variable JAVACOREDIR, see *14.7 Details of environment variables used in JavaVM* in the *uCosminexus Application Server Definition Reference Guide*.

(2) Settings for acquiring JavaVM logs

This subsection describes the settings for acquiring JavaVM logs.

The JavaVM log is a log that you can acquire with the extension option added by Hitachi in the standard JavaVM. You can acquire more troubleshooting information as compared to the information acquired from the standard JavaVM. This log file is called *JavaVM log file*. The JavaVM GC log is also output to this file.

To acquire the JavaVM log file, specify a JavaVM startup option in the JavaVM startup parameter in the Easy Setup definition file. Define the JavaVM startup option in the JavaVM startup parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server). The following are the setting contents of the JavaVM startup parameter:

`<param-name>` tag

`add.jvm.arg`

`<param-value>` tag

JavaVM-startup-option

The following table describes the options to be specified for the JavaVM startup option. For details about the options, see *14. Options for Invoking JavaVM* in the *uCosminexus Application Server Definition Reference Guide*.

Table 3–22: Options to be specified for JavaVM startup option (Settings for acquiring JavaVM log)

Option	Description

`-XX:+HitachiOutOfMemoryStackTrace`
`-XX:+HitachiVerboseGC`
`-XX:+HitachiOutOfMemoryHandling`
`-XX:+HitachiJavaClassLibTrace`
`-XX:+JITCompilerContinuation`
	If any of the following options is specified, the JavaVM log file is output:
	• If `-XX:+HitachiOutOfMemoryStackTrace` is specified, the exception information and the stack trace are output to the JavaVM log file. If you specify this option, `-XX:+HitachiOutOfMemorySize` and `-XX:+HitachiOutOfMemoryCause` are also specified concurrently.
	• If `-XX:+HitachiVerboseGC` is specified, the extended verboseGC information is output to the JavaVM log file, when the GC occurs.
	• If `-XX:+HitachiOutOfMemoryHandling` is specified, information related to the frequency of occurrence of OutOfMemory is output to the JavaVM log file when OutOfMemory occurs due to Java heap insufficiency or insufficient metaspace area.
	• If `-XX:+HitachiJavaClassLibTrace` is specified, the stack trace of the class library is output to the JavaVM log file.
	• If `-XX:+JITCompilerContinuation` is specified, the JIT compiler continuation functionality is enabled, therefore, if the JIT compilation fails due to a logical inconsistency in a method configuring the application, the JIT compiler continuation functionality log is output to the JavaVM log file.

Specify the following extension options based on the requirements and set the output methods and the output contents of the JavaVM log file:

- File size and the number of JavaVM log files
- Extended verboseGC function option
- Extension function option when OutOfMemoryError occurs
- Class library trace function option
- Local variable information output function option
- Asynchronous log file output function option, etc.

The settings to output the extended verboseGC information to the maintenance information of JavaVM (Java heap information) and the GC log are described below. With the options described in the following table, apply the output of extended verboseGC information and specify the output format of the extended verboseGC information.

Table 3–23: Options to be specified for output of the extended verbosegc information

Option	Description
`-XX:+HitachiVerboseGC`	The extended verboseGC information is output to the JavaVM log file.
`-XX:+HitachiVerboseGCPrintDate`	The date is output to each line of the extended verboseGC information.
`-XX:+HitachiVerboseGCCpuTime`	The CPU usage time of the thread executing the GC from the start until the termination of the GC is output. The CPU usage time is divided into the CPU time spent in the user mode and the CPU time spent in the kernel mode and then output.
`-XX:HitachiVerboseGCIntervalTime =` *Time-intervals (seconds)*	Specify the output intervals of the extended verboseGC information.
`-XX:+HitachiVerboseGCPrintCause`	The cause for occurrence of the GC is output to the extended verboseGC information.
`-XX:+HitachiOutputMilliTime`	Output the date (up to milliseconds) to each line of the extended verboseGC information.
`-XX:+HitachiCommaVerboseGC`	The extended verboseGC information is output in the CSV format.
`-XX:+HitachiVerboseGCPrintTenuringDistribution`	The tenuring distribution information of the Survivor area is output. For details about the output format and output information, see *9.11 Tenuring distribution information output functionality of the Survivor area*.
`-XX:+HitachiVerboseGCPrintJVMInternalMemory`	Heap information managed in JavaVM is output to the JavaVM log file.
`-XX:+HitachiVerboseGCPrintThreadCount`	To monitor the number of Java threads, the number of Java threads is output to the JavaVM log file.
`-XX:+HitachiVerboseGCPrintDeleteOnExit`	The cumulative heap size secured by JavaVM by invoking `java.io.File.deleteOnExit()` and the number of times the method is invoked are output to the JavaVM log file.

You can acquire the information to estimate the Java heap area size and the metaspace area size required by the server from the extended verbosegc information.

(3) Settings for acquiring the event log of Explicit Memory Management functionality

This point describes the settings for acquiring the *Event log of Explicit Memory Management functionality*. This point also describes the relation of the event log of the Explicit Memory Management functionality with the JavaVM log file.

The settings required for acquiring the event log of Explicit Memory Management functionality are as follows:

- **Settings for log output level**

  Set up the log output level to output a log depending on the purpose. You can specify the four log output levels; *none*, *normal*, *verbose*, and *debug*. Default level of J2EE server is *normal*. The log details are given in the order of *none*<*normal*<*verbose*<*debug* and the output volume increases.

  For example, you can perform an operation, where you can specify *normal* for normal and *verbose* if an error occurs and can acquire a detailed log.

- **Settings for log output file**

The log of Explicit Memory Management functionality is output to a separate file other than the JavaVM log. Set up the file for the output destination, file size, and number of files according to the assumed operation.

For a J2EE server or batch server, you can set up the above-mentioned settings using JavaVM startup option specified in the JavaVM startup parameter in the Easy Setup definition file. Set up the JavaVM startup option in the JavaVM startup parameter in the `<configuration>` tag of a logical J2EE server (j2ee-server). The settings of the JavaVM startup parameter are as follows:

`<param-name>` tag

`add.jvm.arg`

`<param-value>` tag

JavaVM- startup-option

The following table describes the options to be specified for the JavaVM startup option. For details about the options, see *14. Options for Invoking JavaVM* in the *uCosminexus Application Server Definition Reference Guide*.

Table 3–24: Option to be specified for JavaVM startup option (Settings for acquiring the event log of the Explicit Memory Management functionality)

Setting contents	Option	Description
Setting of output level	-XX:HitachiExplicitMemoryLogLevel:*character-string*	Specify log output level. • none Event log of the Explicit Memory Management functionality is not output. • normal Specify for normal operation. The status of the Explicit heap is output when an event that causes a significant change in the size of the Explicit heap occurs or when GC occurs. • verbose Specify if a detailed log is required when error occurs. • debug Specify when a detailed log is required more than verbose. Performance of the system is lowered.
Settings for file to be output	-XX:HitachiExplicitMemoryJavaLog:*character-string*	Specify the file name of file that outputs event log.
	-XX:HitachiExplicitMemoryJavaLogFileSize=*positive-integer*	specify the maximum file size for each file.
	-XX:HitachiExplicitMemoryJavaLogNumberOfFile=*positive-integer*	Specify a maximum number of log files to be created. When a specified value exceeds, a wraparound is performed, and the log is output to the file that was created first.

Note that the event log of the Explicit Memory Management functionality is output to a file that differs from the JavaVM log file. However, the values set up in the JavaVM log file are inherited for some options.

The following table describes the items that inherit the settings of the JavaVM log file.

Table 3–25: Items that inherit the settings of the JavaVM log file

Option name	Meaning	Default value
`-XX:+HitachiJavaLogNoMoreOutput`	An option to specify behavior when an attempt to output the log file fails.	Enabled
`-XX:+HitachiOutputMilliTime`	An option to specify whether to set millisecond as the unit of time output to the log file.	Disabled

For details about the JavaVM extension options, see *14.2 Details of JavaVM extension options* in the *uCosminexus Application Server Definition Reference Guide*.

3.3.18 Settings for acquiring the WebSocket container log

This subsection describes the items that you can set up for acquiring the WebSocket container log.

The following table describes the settings of the items that you can change for acquiring the WebSocket container log and the corresponding parameters of the Easy Setup definition file.

Table 3–26: Settings for acquiring WebSocket container logs

Log or trace	Items	Corresponding parameters of the Easy Setup definition file
Access log	Whether the access log is output	`ejbserver.logger.access_log.websocket.enabled` in the `<configuration>` tag on a logical J2EE server (j2ee-server) (By default, the access log is not output.)
	Format when the access log is output	`ejbserver.logger.access_log.websocket.format` in the `<configuration>` tag on a logical J2EE server (j2ee-server)#
	File size of the access log	`ejbserver.logger.channels.define.WebSocketAccessLogFile.filesize` in the `<configuration>` tag on a logical J2EE server (j2ee-server)
	Number of files of access log	`ejbserver.logger.channels.define.WebSocketAccessLogFile.filenum` 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 about acquiring the trace based performance analysis files, see *7. Performance Analysis by Using Trace Based Performance Analysis*.

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 NIO HTTP server, see *6.11.2 Customizing the access log of the NIO HTTP server* in the *uCosminexus Application Server Web Container Functionality Guide*.

4 Output Destinations and Output Methods of Data Required for Troubleshooting

You can acquire the data that is used in the troubleshooting at any time for each data. This chapter describes how to seperately output the data, such as logs and thread dumps, and the default output destination for the data. This chapter also describes the directories to be used for troubleshooting.

4.1 Organization of this chapter

This chapter describes the output destination and the output method of the data used for troubleshooting.

There are the respective default output destinations for the data used for troubleshooting. Additionally, you can separately acquire the data required for troubleshooting, such as the information without using the snapshot log.

The following table describes the organization of this chapter.

Table 4–1: Organization of this chapter (Output destination and output method of the data used for troubleshooting)

Category	Title	Reference
Explanation	Types of data used for troubleshooting (When snapshot log is not used)	*4.2*
	Application Server log (Systems for executing J2EE applications)	*4.3*
	Application Server log (Systems for executing batch applications)	*4.4*
	EJB Client Application System Log	*4.5*
	Trace based performance analysis	*4.6*
	JavaVM thread dump	*4.7*
	JavaVM GC Log	*4.8*
	Memory Dump	*4.9*
	JavaVM log (JavaVM log file)	*4.10*
	JavaVM Output Message Logs (Standard Output or Error Report File)	*4.11*
	OS Status Information and OS Logs	*4.12*
	OS Statistical Information	*4.13*
	Application Server definition information	*4.14*
	Contents of J2EE server or batch server working directory	*4.15*
	Application Server Resource Setting Information	*4.16*
	Web Server Logs	*4.17*
	JavaVM stack trace information	*4.18*
	Event log of the Explicit Memory Management functionality	*4.19*
	Information on the execution of the Component Container Administrator setup command (In UNIX)	*4.20*

For an overview of troubleshooting, how to output the data automatically, setup related to data acquisition and data output, and the output contents of the data, see the following respective chapters:

- Overview of troubleshooting and how to output the data automatically
  2. Troubleshooting
- Setup related to data acquisition and data output
  3. Preparing for Troubleshooting
- Contents output in data
  5. Problem Analysis

4.2 Types of data used for troubleshooting (When snapshot log is not used)

Materials, such as log outputs by the application server configuration software, can also be acquired separately without using the snapshot log. This section describes the methods for acquiring the required data separately when a trouble occurs. For details about acquiring the data using the snapshot log, see *2.3.3 Collecting the Snapshot Log*.

The following table describes the reference sections for the required data acquired separately when the snapshot log is not used.

Table 4–2: Required data that is to be acquired (When snapshot log is not used)

Data to be acquired	Reference
Application Server log (Systems for executing J2EE applications)	*4.3*
Application Server log (Systems for executing batch applications)	*4.4*
System log of EJB client application#	*4.5*
Trace based performance analysis	*4.6*
JavaVM thread dump	*4.7*
JavaVM GC log	*4.8*
Memory dump	*4.9*
JavaVM log (JavaVM log file)	*4.10*
JavaVM output message logs (Standard output or error report file)	*4.11*
OS status information and OS logs	*4.12*
OS statistical information	*4.13*
Application Server definition information	*4.14*
Contents of J2EE server or batch server working directory	*4.15*
Application Server resource setting information	*4.16*
Web server logs	*4.17*
JavaVM stack trace information	*4.18*
Event log of the Explicit Memory Management functionality	*4.19*
Information on the execution of the Component Container Administrator setup command (In UNIX)	*4.20*

#

Acquire for the system executing an EJB client application.

4.3 Application Server log (Systems for executing J2EE applications)

This section describes how to acquire logs output by the component software of the Application Server manually in the systems for executing J2EE applications. Further, when the application server log is already collected as a snapshot log, you need not perform the tasks explained here.

This section describes how to acquire the following logs:

- Cosminexus Component Container logs
- Cosminexus Performance Tracer logs
- Cosminexus Component Transaction Monitor logs
- Logs output in audit log
- Application user logs

> **Reference note**
>
> The method of collecting Cosminexus HTTP Server log is described in *4.17 Web Server Logs*.

4.3.1 Acquiring the Cosminexus Component Container Logs

This section describes the types and output destinations of Cosminexus Component Container logs. The Cosminexus Component Container logs include the following logs:

- J2EE server, server management command logs
- Administration agent, Management agent, Management Server logs
- Internal setup tool of the virtual server manager and Server Communication Agent logs
- Integrated user management logs
- Cosminexus JMS Provider logs

The output destination of each of the log is described below.

(1) Acquiring the J2EE server, server management command logs

This section describes the method for acquiring the J2EE server, server management command logs.

Also, in Cosminexus Component Container, the migration command logs are output besides these logs. When the resource depletion monitoring function is used, the resource depletion monitoring log is output.

- The five types of J2EE server logs are the message log, user log, exception log, access log, and maintenance log. Note that in addition to these logs, event log or syslog are output in the case of starting, stopping, and abnormal termination of a J2EE server.
- The three types of server management command logs are message log, exception log, and maintenance log.
- In the resource adapter version up command (`cjrarupdate`) log, there are three types of logs, message log, exception log, and maintenance log.
- The three types of migration command logs are message log, exception log, and maintenance log.

Each of these logs is described below:

Message log

The operation status of a J2EE server, server management command, and migration command is output. Message log is used as the operation monitoring information for various types of servers and commands.

User log

The information of the standard output and the standard error output in the application is output in the user log. Use this log to check the operation when developing the application. Note that if you specify the `java.security.debug` property and start the server, the standard output and standard error output information is not output to the user log. This log also includes the JavaVM memory related logs.

Exception log

The exception information of Cosminexus Component Container is output when a problem occurs in the system. Note that you need not monitor the exception log in daily operations. Use this log to reference exception information if a message is output to the log.

Access log

The processing results of requests to Web applications and the communication history of WebSocket are output.

Maintenance log

The error maintenance information of Cosminexus Component Container is output when a problem occurs in the system. Maintenance personnel use this log to analyze the errors that occur in Cosminexus Component Container.

Event log (in Windows)

The information indicating start, stop, or abnormal termination of a J2EE server is output to this log. The output destination differs depending on the Windows event log settings.

Note that the event log is not output depending on how the J2EE server has been stopped. In the following cases, there are times when the log is not output correctly:

- When a problem occurs in JavaVM itself when the J2EE server is running
- When the J2EE server process is stopped externally by TerminateProcess
- When the J2EE server terminates abnormally due to insufficient memory when the `-XX:+HitachiOutOfMemoryAbort` option is specified to start the JavaVM.

  Note that the `-XX:+HitachiOutOfMemoryAbort` option is set by default.

syslog (in UNIX)

The information indicating start, stop, or abnormal termination of a J2EE server is output to this log. The output destination differs depending on the settings in the UNIX syslog.

Note that the syslog is not output depending on how the J2EE server has been stopped. In the following cases, there are times when the log is not output correctly:

- When a problem occurs in JavaVM itself when the J2EE server is running
- When the J2EE server process is stopped externally by the SIGKILL signal (such as `kill -9`)
- When the J2EE server terminates abnormally due to insufficient memory when the `-XX:+HitachiOutOfMemoryAbort` option is specified to start the JavaVM.

  Note that the `-XX:+HitachiOutOfMemoryAbort` option is set by default.

Resource depletion monitoring log

When using the resource depletion monitoring function, the resource depletion monitoring information about the resources being monitored is output. Use it for investigating the cause if the resource usage or the used resource quantity exceeds the threshold value.

The log is recorded in an order starting from the log file attached with the smallest file number. When one log file size attains the maximum size for one file, the log is recorded in the log file attached with the next file number. When the last log file (log file attached with the file count number) size attains the maximum size for one file, the log file of file number

1 is made empty and the log is recorded in that file. Thereafter, the log is recorded in the log files in the order of the file number by emptying those log files before recording a new log.

The following table describes the default of log output destination. The Cosminexus Component Container log can be acquired according to the server or command.

The *working-directory* shown in the log output destination indicates a directory specified in the `ejb.public.directory` parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server) of the Easy Setup definition file. The default value is *Cosminexus-installation-directory*`\CC\server\public` (in Windows) or `/opt/Cosminexus/CC/server/public` (in UNIX).

(a) J2EE server log

Table 4–3: Output destination of the J2EE server log (Default)

Category	Contents	Log output destination and log file name[#1]	Default size × number of files	Channel name

Message log	Operation log	• In Windows
		`ejb.server.log.directory`[#2]`\cjmessage[`*n*`].log`
		• In UNIX
		`ejb.server.log.directory`[#2]`/cjmessage[`*n*`].log`
			1MB × 2	MessageLogFile

	Log operation log[#3]	• In Windows
		`<ejb.server.log.directory>`[#2]`\cjlogger.log`
		• In UNIX
		`<ejb.server.log.directory>`[#2]`/cjlogger.log`
			1MB × 2	--

	Operation log of resource adapter that is deployed and used as a J2EE resource adapter[#4]	• In Windows
		(Resource adapter of Connector 1.0 specifications)
		`ejb.server.log.directory`[#2]`\connectors\resource-adapter-display-name[`*n*`].log`
		(Resource adapter of Connector 1.5 specifications)
		`<ejb.server.log.directory>`[#2]`\connectors\resource-adapte-display-name_connection-defintion-arrangement-order_[`*n*`].log`
		• In UNIX
		(Resource adapter of Connector 1.0 specification)
		`ejb.server.log.directory`[#2]`/connectors/resource-adapter-display-name[`*n*`].log`
		(Resource adapter of Connector 1.5 specification)
		`<ejb.server.log.directory>`[#2]`/connectors/resource-adapter-display-name_connetion-definition-arrangement-order_[`*n*`].log`
			2 MB × 4	--

Category	Contents	Log output destination and log file name[#1]	Default size × number of files	Channel name

	Operation log of resource adapter used in a J2EE application[#4]	• In Windows
		(Resource adapter of normal mode/ Connector 1.0 specifications)
		`<ejb.server.log.directory>`[#2]`\connectors\J2EE application-name\resource-adapter-display-name[n].log`
		(Resource adapter of test mode/Connector 1.0 specifications)
		`<ejb.server.log.directory>`[#2]`\connectors\test#J2EE-application-name\resource-adapter-display-name[n].log`
		(Resource adapter of `normal mode/ Connector 1.5` specifications)
		`<ejb.server.log.directory>`[#2]`\connectors\J2EE-application-name\resource-adapter-display-name_connection-defintion-arrangement-order_[n].log`
		(Resource adapter of test mode/Connector 1.5 specifications)
		`<ejb.server.log.directory`[#2]`>\connectors\test#J2EE-application-name\resource-adapter-display name_connection-defintion-arrangement-order_[n].log`
		• In UNIX
		(Resource adapter of normal mode/ Connector 1.0 specifications)
		`<ejb.server.log.directory>`[#2]`/connectors/J2EE-application-name/resource-adapter-display-name[n].log`
		(Resource adapter of test mode/Connector 1.0 specifications)
		`<ejb.server.log.directory>`[#2]`/connectors/test#J2EE-application-name/resource-adapter-display-name [n].log`
		(Resource adapter normal mode/Connector 1.5 specifications)
		`<ejb.server.log.directory>`[#2]`/connectors/J2EE-application-name/resource-adapter-display-name_connection-defintion-arrangement-order_[n].log`
		(Resource adapter of test mode/Connector 1.5 specifications)
		`<ejb.server.log.directory>`[#2]`/connectors/test#J2EE-application-name/resource-adapter-display-`
			2 MB × 4	--

Category	Contents	Log output destination and log file name[#1]	Default size × number of files	Channel name
		name _connection-definition-arrangement-order_[*n*].log		
User log	Web servlet log[#5]	• In Windows ejb.server.log.directory[#2]\web_servlet[*n*].log • In UNIX ejb.server.log.directory[#2]/web_servlet[*n*].log	4MB × 4	WebServletLogFile
	User output log	• In Windows ejb.server.log.directory[#2]\user_out[*n*].log • In UNIX ejb.server.log.directory[#2]/user_out[*n*].log	1MB × 2	UserOutLogFile
	User error log	• In Windows ejb.server.log.directory[#2]\user_err[*n*].log • In UNIX ejb.server.log.directory[#2]/user_err[*n*].log	1MB × 2	UserErrLogFile
	JavaVM maintenance information and GC log	• In Windows ejb.server.log.directory[#2]\javalog[*nn*].log • In UNIX ejb.server.log.directory[#2]/javalog[*nn*].log	4MB × 4	--
	Event log of Explicit Memory Management functionality	• In Windows ejb.server.log.directory[#2]\ehjavalog[*nn*].log • In UNIX ejb.server.log.directory[#2]/ehjavalog[*nn*].log	4MB × 4	--
Exception log	Exception information when an error occurs	• In Windows ejb.server.log.directory[#2]\cjexception[*n*].log • In UNIX ejb.server.log.directory[#2]/cjexception[*n*].log	1MB × 2	ExceptionLogFile
Maintenance log	Maintenance information	• In Windows <ejb.server.log.directory>[#2]\CC\maintenance\cjmaintenance[*n*].log • In UNIX ejb.server.log.director[#2]/CC/maintenance/cjmaintenance[*n*].log	16MB × 4	MaintenanceLogFile
	Console message	• In Windows	1MB × 2	ConsoleLogFile

Category	Contents	Log output destination and log file name[#1]	Default size × number of files	Channel name
		<ejb.server.log.directory>[#2]\CC\maintenance\cjconsole[*n*].log • In UNIX ejb.server.log.directory[#2]/CC/maintenance/cjconsole[*n*].log		
	EJB container maintenance information	• In Windows <ejb.server.log.directory>[#2]\CC\maintenance\cjejbcontainer[*n*].log • In UNIX ejb.server.log.directory[#2]/CC/maintenance/cjejbcontainer[*n*].log	1MB × 2	EJBContainerLogFile
	Web container maintenance information	• In Windows <ejb.server.log.directory>[#2]\CC\maintenance\cjwebcontainer[*n*].log • In UNIX ejb.server.log.directory[#2]/CC/maintenance/cjwebcontainer[*n*].log	1MB × 2	WebContainerLogFile
	Start process standard output information[#6]	• In Windows <ejb.server.log.directory>[#2]\CC\maintenance\cjstdout.log • In UNIX ejb.server.log.directory[#2]/CC/maintenance/cjstdout.log	--	--
	Start process standard error information[#6]	• In Windows <ejb.server.log.directory>[#2]\CC\maintenance\cjstderr.log • In UNIX ejb.server.log.directory[#2]/CC/maintenance/cjstderr.log	--	--
	Termination process information	• In Windows <ejb.server.log.directory>[#2]\CC\maintenance\cj_shutdown[*n*].log • In UNIX ejb.server.log.directory[#2]/CC/maintenance/cj_shutdown[*n*].log	4KB × 2[#7]	--
	Trace log for Web container maintenance	• In Windows <ejb.server.log.directory>[#2]\CC\maintenance\cjweb_access[*n*].log • In UNIX	4 MB × 16	WebAccessLogFile

Category	Contents	Log output destination and log file name[#1]	Default size × number of files	Channel name
		<ejb.server.log.directory>[#2]/CC/maintenance/cjweb_access[*n*].log		
	RMI communication log of J2EE server	• In Windows <ejb.server.log.directory>[#2]\CC\rmi\cjrmi[n].log • In UNIX <ejb.server.log.directory>[#2]/CC/rmi/cjrmi[n].log	1MB × 4	--
Event log	Log showing J2EE server start, stop or abnormal termination	Application log of Windows event viewer[#8]	--	--
syslog	Log showing J2EE server start, stop or abnormal termination	Depends on UNIX syslog settings.[#9]	--	--
Debug log	Log for checking development	• In Windows <ejb.server.log.directory>[#2]\cjdevelopment[*n*].log • In UNIX <ejb.server.log.directory>[#2]/cjdevelopment[*n*].log	1 MB × 4	DevelopmentLog File
Access log	Processing results of HTTP communication	• In Windows <ejb.server.log.directory>[#2]\cj_access_niohttp[*n*].log • In UNIX <ejb.server.log.directory>[#2]/cj_access_niohttp[*n*].log	4 MB × 16	NIOHTTPAccess LogFile
	Processing results of WebSocket communication	• In Windows <ejb.server.log.directory>[#2]\cj_access_websocket[*n*].log • In UNIX <ejb.server.log.directory>[#2]/cj_access_websocket[*n*].log	4 MB × 16	WebSocketAccess LogFile

Legend:

--: Not applicable

Note:

Channel name is the name to identify the output destination of the log. Use it as a key value when changing log attributes (size, number of files).

#1

In the part of the log file name [*n*], the file number (number of files from 1 (maximum 16)) is added.

However, when using the sub directory shared mode of an EJB client application, the maximum number of files is 64.

Moreover, in the [nn] part, a serial number from 01 to 99 is added.

#2

<ejb.server.log.directory> indicates a directory specified in the ejb.server.log.directory parameter in the <configuration> tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file. The default value is *Cosminexus-installation-directory*\CC\server\public\ejb*server-name*\logs.

For details about the ejb.server.log.directory parameter in the Easy Setup definition file, see *4.11.3 Parameters used for setting up the option definitions for the J2EE server* and *2.2.2 usrconf.cfg (Option definition file for J2EE servers)* in the *uCosminexus Application Server Definition Reference Guide*.

#3

Check the file contents when the file is output. If the maximum size is exceeded while checking, rename the `cjlogger.log` file with the backup file name (`cjlogger_save.log`).

#4

Decision about acquiring the resource adapter log depends on the contents specified in server management commands. Moreover, you can use the simple setup definition file to change the size and number of files for a resource adapter log. For details about the settings for acquiring the resource adapter logs, see *3.3.11 Settings for Acquiring the Resource Adapter Logs*.

#5

Stack trace for the exception that occurred in a servlet and JSP is also output.

#6

It is a log in which only the start process information is acquired. As it is output mainly while starting and terminating J2EE server, this log is almost not output online. When the file size reaches the upper limit, it is saved in `cjstdout_save.log` or `cjstderr_save.log` under *working-directory*`\ejb\`*server-name*`\logs` (in Windows) or under *working-directory*`/ejb/`*server-name*`/logs` (in UNIX). If `cjstdout_save.log` or `cjstderr_save.log` already exists, it is overwritten.

#7

The size and number of files cannot be changed.

#8

The output destination of the log file differs depending on the Windows event log settings.

#9

To output messages related to J2EE server start, stop, and abnormal termination, to syslog, it is necessary to set the priority for the facility daemon to info or debug in the syslog settings. For details about the syslog settings, see the manual provided with the OS.

Reference note

When the session failover function is used, the log for session failover is output as the log of the J2EE server using the session failover function.

(b) Server management command log

Table 4–4: Output destination of the server management command log (Default)

Category	Contents	Log output destination and log file name#1	Default size × number of files	Channel name
Message log	Operation log#2, #3	• In Windows *Cosminexus-installation-directory*`\CC\admin\logs\cjmessage[`*n*`].log` • In UNIX `/opt/Cosminexus/CC/admin/logs/cjmessage[n].log`	1024KB × 3	MessageLogFile
	Log operation log#2	• In Windows *Cosminexus-installation-directory*`\CC\admin\logs\cjlogger.log` • In UNIX `/opt/Cosminexus/CC/admin/logs/ cjlogger.log`	1024KB × 2	--
Exception log	Exception information when an error occurs#2, #3	• In Windows *Cosminexus-installation-directory*`\CC\admin\logs\cjexception[`*n*`].log` • In UNIX	1024KB × 6	ExceptionLogFile

Category	Contents	Log output destination and log file name#1	Default size × number of files	Channel name
		/opt/Cosminexus/CC/admin/logs/cjexception[*n*].log		
Maintenance log	Maintenance information#2	• In Windows *Cosminexus-installation-directory*\CC\admin\logs\CC\maintenance\cjmaintenance[*n*].log • In UNIX /opt/Cosminexus/CC/admin/logs/CC/maintenance/cjmaintenance[*n*].log	1024KB × 3	MaintenanceLog File
	Console message#2	• In Windows *Cosminexus-installation-directory*\CC\admin\logs\CC\maintenance\cjconsole[*n*].log • In UNIX /opt/Cosminexus/CC/admin/logs/CC/maintenance/cjconsole[*n*].log	32KB × 3	ConsoleLogFile
	Maintenance information of the server management command#2	• In Windows *Cosminexus-installation-directory*\CC\admin\logs\CC\maintenance\cjserveradmin[*n*].log • In UNIX /opt/Cosminexus/CC/admin/logs/CC/maintenance/cjserveradmin[*n*].log	32KB × 3	ServerAdminLog File

Legend:

--: Not applicable

Note:

Channel name is the name to identify the output destination of the log.

#1

In the [*n*] part of the log file name, add the file number (from 1 to the maximum number of files for each log).

#2

The command name is displayed in the output message (application identification name) of the Hitachi Trace Common Library format. For details on the log in the Trace Common Library format, see *5.2 Application Server Log*.

#3

For the compatibility mode, the output destination of the operation log and the exception information when an error occurs will differ from the standard mode. For the compatibility mode, the output destination, the default size, and the number of files are as follows:

Table 4–5: Output destination of server management commands log (Compatibility mode)

Contents	Log output destination and log file name#	Default size × number of files
Operation log	• In Windows *Cosminexus-installation-directory*\CC\admin\logs\<command-name>message[*n*].log • In UNIX /opt/Cosminexus/CC/admin/logs/command-name/message[*n*].log	128KB × 2

Contents	Log output destination and log file name#	Default size × number of files
Exception information when an error occurs	• In Windows *Cosminexus-installation-directory*\CC\admin\logs\<command-name>exception[*n*].log • In UNIX /opt/Cosminexus/CC/admin/logs/command-name/exception[*n*].log	256KB × 2

#

In the [*n*] part of the log file name, add the file number (from 1 to the maximum number of files for each log).

In the messages output to the message log of the server management commands, there are the cases when the message ID field is blank and message ID (such as KDJEnnnnn-Y) is included in the message text field. This is an additional information of the messages output before or after the messages are output at the server side.

(c) Resource adapter version-up command (cjrarupdate) log

Table 4–6: Output destination of the resource adapter version-up command (cjrarupdate) log

Category	Contents	Log output destination and log file name#	Default size × number of files
Message log	Operation log	• In Windows *Cosminexus-installation-directory*\CC\logs\cjrarupdatemessage[*n*].log#1 • In UNIX /opt/Cosminexus/CC/logs/cjrarupdatemessage[*n*].log#1	1MB × 2
Exception log	Exception information when an error occurs	• In Windows *Cosminexus-installation-directory*\CC\logs\cjrarupdateexception[*n*].log#1 • In UNIX /opt/Cosminexus/CC/logs/cjrarupdateexception[*n*].log#1	1MB × 2
Maintenance log	Maintenance information	• In Windows *Cosminexus-installation-directory*\CC\logs\cjrarupdatemaintenance[*n*].log#2 • In UNIX /opt/Cosminexus/CC/logs/cjrarupdatemaintenance[*n*].log#2	16 MB × 4

#1

In the [*n*] part of the log file name, add the file number (1 or 2).

#2

In the [*n*] part of the log file name, add the file number (from 1 to 4).

(d) Migration command (cjenvupdate) log

Table 4–7: Output destination of the migration command (cjenvupdate) log

Category	Contents	Log output destination and log file name#	Default size × number of files
Message log	Operation log of the cjenvupdate command	• In Windows *Cosminexus-installation-directory*\CC\logs\cjenvupdatemessage[*n*].log • In UNIX /opt/Cosminexus/CC/logs/cjenvupdatemessage[*n*].log	4MB × 4
Exception log	Exception information of the cjenvupdate command	• In Windows *Cosminexus-installation-directory*\CC\logs\cjenvupdateexception[*n*].log • In UNIX /opt/Cosminexus/CC/logs/cjenvupdateexception[*n*].log	4MB × 4
Maintenance log	Maintenance information of the cjenvupdate command	• In Windows *Cosminexus-installation-directory*\CC\logs\cjenvupdatemaintenance[*n*].log • In UNIX /opt/Cosminexus/CC/logs/cjenvupdatemaintenance[*n*].log	4MB × 4

#

In [*n*], the file number (from 1 to 4) is attached.

(e) Resource depletion monitoring log

Table 4–8: Output destination of resource depletion monitoring log

Monitored Resources	Log acquisition location and log file name#1	Default size × number of files	Channel name
Memory	• In Windows <ejb.server.log.directory>#2\watch\cjmemorywatch[*n*].log • In UNIX ejb.server.log.directory#2/watch/cjmemorywatch[*n*].log	1MB × 2	MemoryWatchLogFile
File descriptors	• In UNIX#3 ejb.server.log.directory#2/watch/cjfiledescriptorwatch[*n*].log	1MB × 2	FileDescriptorWatchLogFile
Threads	• In Windows <ejb.server.log.directory>#2\watch\cjthreadwatch[*n*].log • In UNIX ejb.server.log.directory#2/watch/cjthreadwatch[*n*].log	1MB × 2	ThreadWatchLogFile
Thread dump	• In Windows <ejb.server.log.directory>#2\watch\cjthreaddumpwatch[*n*].log	1MB × 2	ThreaddumpWatchLogFile

Monitored Resources	Log acquisition location and log file name#1	Default size × number of files	Channel name
	• In UNIX `ejb.server.log.directory`#2`/watch/cjthreaddumpwatch[`*n*`].log`		

HTTP requests pending queue	• In Windows
	`<ejb.server.log.directory>`#2`\watch\cjrequestqueuewatch[`*n*`].log`
	• In UNIX
	`ejb.server.log.directory`#2`/watch/cjrequestqueuewatch[`*n*`].log`
		1MB × 2	RequestQueueWatchLogFile

HTTP session numbers	• In Windows
	`<ejb.server.log.directory>`#2`\watch\cjhttpsessionwatch[`*n*`].log`
	• In UNIX
	`ejb.server.log.directory`#2`/watch/cjhttpsessionwatch[`*n*`].log`
		1MB × 2	HttpSessionWatchLogFile

Connection pool	• In Windows
	`<ejb.server.log.directory>`#2`\watch\cjconnectionpoolwatch[`*n*`].log`
	• In UNIX
	`ejb.server.log.directory`#2`/watch/cjconnectionpoolwatch[`*n*`].log`
		1MB × 2	ConnectionPoolWatchLogFile

Note:

Channel name is the name to identify the output destination of the log. Use it as a key value when changing log attributes (size, number of files).

#1

In [*n*], the file number (number of files from 1 (maximum 16)) is attached.

#2

`<ejb.server.log.directory>` indicates the directory specified in the `ejb.server.log.directory` parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server) of the Easy Setup definition file. The default value is *Cosminexus-installation-directory*`\CC\server\public\ejb\`*server-name*`\logs`.

For details about the `ejb.server.log.directory` parameter in the Easy Setup definition file, see *4.11.3 Parameters used for setting up the option definitions for the J2EE server* and *2.2.2 usrconf.cfg (Option definition file for J2EE servers)* in the *uCosminexus Application Server Definition Reference Guide*.

#3

The file descriptor cannot be monitored in Windows and AIX.

For details about the information output to the resource depletion monitoring log file and for the output format of the log file, see *4.3 Resource depletion monitoring functionality and output of resource depletion monitoring information* in the *uCosminexus Application Server Operation, Monitoring, and Linkage Guide*.

(f) User definition file to set output destination of the log

When the output destination of the J2EE server and server management command log output destination is changed, reference the user definition file in which the log output destination is set, described in the following table and confirm the output destination. Note that if the output destination of a log is changed, that log is not collected when the snapshot log is collected in a batch. Change the collection destination of the snapshot log as and when required.

Table 4–9: User definition file in which the output destination of log is set

Category	User definition file
J2EE server	The `ejb.server.log.directory` parameter is specified in the `<configuration>` tag of the logical J2EE server (j2ee-server) of the Easy Setup definition file. The default is *Cosminexus-installation-directory*`\CC\server\public\ejb\`*server-name*`\logs` (in Windows), or `/opt/Cosminexus/CC/server/public/ejb/`*server-name*`/logs` (in UNIX).
Server management commands	`ejbserver.log.directory` key of the server management command usrconf.bat (in Windows) or usrconf (in UNIX) The default key is *Cosminexus-installation-directory*\CC\admin\logs (in Windows) or /opt/Cosminexus/CC/admin/logs (in UNIX). You cannot change the output destination of the server management command log when operating from the Management Server remote management functionality.

For details about the settings of the data acquisition for troubleshooting, such as how to change the output destination of the logs, see the chapter *3. Preparing for Troubleshooting*.

(2) Acquiring the log of Administration agent, Management agent, and Management Server

This points describes the output destination of the Administration Agent, Management Agent, and the Management Server log.

In the Administration agent, Management agent, and Management Server logs, besides acquiring separately, you can also acquire these logs by compiling as a Integrated log. Following are the types of integrated logs:

- Integrated message log
  The message logs of the Manager are integrated and output.
- Integrated trace log
  The trace logs of the Manager are integrated and output.
- Command maintenance log#
  The management commands, the commands used with Smart Composer functionality, and the trace log of the snapshotlog command are integrated and output.

For details about the commands used with the Smart Composer functionality, see *8. Commands Used with the Smart Composer Functionality* in the *uCosminexus Application Server Command Reference Guide*.

The following table describes the output destination of integrated logs.

Table 4–10: Output destination of the integrated log (In Windows)

File name	Contents	Output destination directory	Default size × number of files
mngmessage[*n*].#1log	Integrated message log	*Log-output-directory-of-Manager*\message#2	256KB × 4
mngtrace[*n*]#1.log	Integrated trace log	*Log-output-directory-of-Manager*\trace#2	1MB × 4
mngcmd[*n*]#1.log	Command maintenance log	*Log-output-directory-of-Manager*\maintenance#2	16MB × 4

#1
In the part of the log file name [*n*], the file number (number of files from 1 (maximum 64)) is added.

#2

Log-output-directory-of-Manager indicates the directory specified in manager.cfg (Manager log settings file). The default value is *Cosminexus-installation-directory*\manager\log. For details about manager.cfg, see *8.2.9 manager.cfg (Manager settings file)* in the *uCosminexus Application Server Definition Reference Guide*.

Table 4–11: Output destination of the integrated log (in UNIX)

File name	Contents	Output destination directory	Default size × number of files
mngmessage[*n*]#1.log	Integrated message log	*Log-output-directory-of-Manager*/message#2	256KB × 4
mngtrace[*n*]#1.log	Integrated trace log	*Log-output-directory-of-Manager*/trace#2	1MB × 4
mngcmd[*n*]#1.log	Command maintenance log	*Log-output-directory-of-Manager*/maintenance#2	16MB × 4

#1

In the part of the log file name [*n*], the file number (number of files from 1 (maximum 64)) is added.

#2

Log-output-directory-of-Manager indicates the directory specified in manager.cfg (Manager log settings file). The default value is /opt/Cosminexus/manager/log. For details about manager.cfg, see *8.2.9 manager.cfg (Manager settings file)* in the *uCosminexus Application Server Definition Reference Guide*.

Note that the log output in the integrated log is output separately by default.

Additionally, the logs output in the integrated log might not output separately. For details about the settings to output integrated logs, see *3.3.10 Settings for Acquiring the Cosminexus Manager Log*.

The following table describes the Administration agent, Management agent, and Management Server log output destination as well as the possibility of output to integrated message log and integrated trace log when acquiring separately.

Table 4–12: Output destination when acquiring the Administration agent, Management agent, and Management Server log separately (in Windows)

Category	File name	Contents	Output destination directory	Default size × number of files	Integrated message log/ integrated trace log
Administration agent	adminagent.err.[1-16].log#1	Standard error output of Administration agent	*Log-output-directory-of-the-Manager*	64KB × 4	N
	adminagent.out.[1-16].log#1	Standard output of Administration agent		64KB × 4	N
	adminagent.err	Standard command line error output of Administration agent		--	N
	adminagent[1-16].log	Administration agent log		64KB × 4	Y
	adminagentctl.exe.[1-2].log	Administration agent start, stop command log		64KB × 2	N
	adminagent[*n*]#2.log	Administration agent maintenance log	*Log-output-directory-of-Manager*\maintenance	16MB × 4	N

Category	File name	Contents	Output destination directory	Default size × number of files	Integrated message log/ integrated trace log
	mngrmi[*n*][#2].log	Maintenance log in RMI processing executed by Administration Agent		16MB × 8	N
	processConsole[*n*][#2].log	Console log	*Log-output-directory-of-the-Manager*	64KB × 4	N
	adminagentsv.exe.[1-16]log[#1]	Administration agent service log		64KB × 2	N
	adminagentsv.exe.out[#1]	Administration agent service standard output		--	N
	adminagentsv.exe.err[#1]	Administration agent service standard error output		--	N
	adminagent.javalog[01-04].log	JavaVM log file of Administration Agent		256KB × 4	N
Management agent	mngagent-*domain-name-agent-name*.[*n*][#2].log[#3]	• Management agent log and trace • System JP1 event and user JP1 event log[#4] for J2EE server • Management event published log[#5]		64KB × 4	N
Management Server	mngsvr.exe.[1-2].log	Management Server service log		64KB × 2	N
	mngsvr.exe.err[#1]	Management Server service standard error output		--	N
	mngsvr.exe.out[#1]	Management Server service standard output		--	N
	mngsvrctl.exe.[1-2].log	Management Server service start, stop command log		64KB × 2	N
	mngsvr[*n*][#2].log	• Management Server log • Management server[#6] system JP1 event log[#4]		64KB × 4	Y
	mngsvr[*n*][#2].log	Management Server maintenance log	*Log-output-directory-of-Manager*\maintenance	16MB × 2	N
	cjmessage[*n*].log	Operation log	*Manager-log-output-directory*	1 MB ×2	N
	cjexception[*n*].log	Exception information for error	*Manager-log-output-directory*\maintenance	1 MB ×2	N

Category	File name	Contents	Output destination directory	Default size × number of files	Integrated message log/ integrated trace log
	cjmaintenance[*n*].log	Maintenance information		16 MB ×4	N
	cjconsole[*n*].log	Console message		1 MB ×2	N
	cjejbcontainer[*n*].log	EJB container maintenance information		1 MB ×2	N
	web_servlet[*n*].log	Web servlet log		4 MB ×4	N
	user_out[*n*].log	User output log		1 MB ×2	N
	user_err[*n*].log	User error log		1 MB ×2	N
	cjlogger.log	Log operation log		1 MB ×2	N
	javalog[*nn*].log	JavaVM maintenance information and GC log		4 MB ×4	N
	ehjavalog[*nn*].log	Event log of the Explicit Memory Management functionality		4 MB ×4	N
	cjwebcontainer[*n*].log	Web container maintenance information		1 MB ×2	N
	cjstdout.log	Standard output information for the start process	*Manager-log-output-directory*\maintenance/CC/maintenance	--	N
	cjstderr.log	Standard error information for the start process		--	N
	cj_shutdown[*n*].log	End process information		4 KB ×4	N
	cjrmi[*n*].log	RMI communication log of the J2EE server for the RMI processing executed by the Management Server	*Manager-log-output-directory*\maintenance\CC\rmi	1 MB ×4	N
	cjhttp_thr.*time-information*.inprocess_http.mm	Thread trace information	*Manager-log-output-directory*\maintenance\http\maintenance\thr	About 3.2 MB ×16	N
	cjhttp_comm.*time-information*.inprocess_http.mm	Communication trace information	*Manager-log-output-directory*\maintenance\http\maintenance\comm	About 16.6 MB ×16	N
	cjmemorywatch[*n*].log	Resource depletion monitoring log (Memory)	*Manager-log-output-directory*\maintenance\watch	1 MB ×2	N
	cjthreadwatch[*n*].log	Resource depletion monitoring log (Thread)		1 MB ×2	N

Category	File name	Contents	Output destination directory	Default size × number of files	Integrated message log/ integrated trace log
	`cjthreaddumpwatch[n].log`	Resource depletion monitoring log (Thread dump)		1 MB ×2	N
	`cjrequestqueuewatch[n].log`	Resource depletion monitoring log (HTTP request pending queue)		1 MB ×2	N
	`cjhttpsessionwatch[n].log`	Resource depletion monitoring log (Number of HTTP sessions)		1 MB ×2	N

Legend:

Y: Output to an integrated log

N: No output to an integrated log

[1-n]: Shows that a serial number in the range from 1 to *n* is used for the number of log files.

--: Not applicable

Note:

`<Manager-log-output-directory>` shows the directory specified in `manager.cfg` (Manager log setup file). The default is *Cosminexus-installation-directory*`\manager\log` (in Windows) or `/opt/Cosminexus/manager/log` (in UNIX). For details on `manager.cfg`, see *8.2.9 manager.cfg (Manager settings file)* in the *uCosminexus Application Server Definition Reference Guide*.

#1

The log is output in a format different than Hitachi Trace Common Library format. For details on the log in the Trace Common Library format, see *5.2 Application Server Log*.

#2

In the part of file name [*n*], the serial number from 1 to total number of specified log files is added.

#3

You can change the output destination of Management agent log trace. If the output destination of Management agent log and trace is changed, see the mngagent.log.filename key value of the `mngagent.properties` file (Management agent property file).

#4

This log is output if a system built with the application server by integrating with JP1 is to be operated.

#5

This log is output when a Management event is used.

#6

This log is for the Management Server of the Application Server.

Table 4–13: Output destination when acquiring the Administration agent, Management agent, and Management Server log separately (in UNIX)

Category	File name	Contents	Output destination directory	Default size × number of files	Integrated message log/ integrated trace log
Administration agent	`adminagent.err.[1-16].log`#1	Standard error output of Administration agent	*Log-output-directory-of-the-Manager*	64KB × 4	N
	`adminagent.out.[1-16].log`#1	Standard output of Administration agent		64KB × 4	N
	`adminagent.err`	Standard command line error output of Administration agent		--	N

Category	File name	Contents	Output destination directory	Default size × number of files	Integrated message log/ integrated trace log
	`adminagent[1-16].log`	Administration agent log		64KB × 4	Y
	`adminagentctl.[1-16].log`	Administration agent start, stop command log		64KB × 2	N
	`adminagent[`*n*`]`[#2]`.log`	Administration agent maintenance log	*Log-output-directory-of-Manager*`/maintenance`	16MB × 4	N
	`mngrmi[`*n*`]`[#2]`.log`	Maintenance log in RMI processing executed by Administration Agent		16MB × 8	N
	`processConsole[`*n*`]`[#2]`.log`	Console log	*Log-output-directory-of-the-Manager*	64KB × 4	N
	`adminagent.javalog[01-04].log`	JavaVM log file of Administration Agent		256KB × 4	N
Management agent	`mngagent-`*domain-name-agent-name*`.[`*n*`]`[#2]`.log`[#3]	• Management agent log and trace • System JP1 event and user JP1 event log[#4] for J2EE server • Management event published log[#5]		64KB × 4	N
Management Server	`mngsvrctlstart.[1-2].log`	Management Server start command		64KB × 2	N
	`mngsvrctlstop.[1-2].log`	Management Server stop command		64KB × 2	N
	`mngsvrctlsetup.[1-2].log`	Management Server set up command		64KB × 2	N
	`mngsvr[`*n*`].log`	• Management Server log • Management server[#6] system JP1 event log[#4]		64KB × 4	Y
	`mngenvsetup.[1-2].log`	Execution log of the `mngenvsetup` command	*Log-output-directory-of-Manager*`/maintenance`	512KB × 2	N
	`mngsvr[`*n*`]`[#2]`.log`	Management Server maintenance log	*Log-output-directory-of-Manager*`/maintenance`	16MB × 2	N
	`cjmessage[`*n*`].log`	Operation log	*Manager-log-output-directory*	1 MB ×2	N
	`cjexception[`*n*`].log`	Exception information for error	*Manager-log-output-directory*`/maintenance`	1 MB ×2	N
	`cjmaintenance[`*n*`].log`	Maintenance information		16 MB ×4	N
	`cjconsole[`*n*`].log`	Console message		1 MB ×2	N

Category	File name	Contents	Output destination directory	Default size × number of files	Integrated message log/ integrated trace log
	cjejbcontainer[*n*].log	EJB container maintenance information		1 MB ×2	N
	web_servlet[*n*].log	Web servlet log		4 MB ×4	N
	user_out[*n*].log	User output log		1 MB ×2	N
	user_err[*n*].log	User error log		1 MB ×2	N
	cjlogger.log	Log operation log		1 MB ×2	N
	javalog[*nn*].log	JavaVM maintenance information and GC log		4 MB ×4	N
	ehjavalog[*nn*].log	Event log of the Explicit Memory Management functionality		4 MB ×4	N
	cjwebcontainer[*n*].log	Web container maintenance information		1 MB ×2	N
	cjstdout.log	Standard output information for the start process	*Manager-log-output-directory*/maintenance/CC/maintenance	--	N
	cjstderr.log	Standard error information for the start process		--	N
	cj_shutdown[*n*].log	End process information		4 KB ×4	N
	cjrmi[*n*].log	RMI communication log of the J2EE server for the RMI processing executed by the Management Server	*Manager-log-output-directory*/maintenance/CC/rmi	1 MB ×4	N
	cjhttp_thr.*time-information*.inprocess_http.mm	Thread trace information	*Manager-log-output-directory*/maintenance/http/maintenance/thr	About 3.2 MB ×16	N
	cjhttp_comm.*time-information*.inprocess_http.mm	Communication trace information	*Manager-log-output-directory*/maintenance/http/maintenance/comm	About 16.6 MB ×16	N
	cjmemorywatch[*n*].log	Resource depletion monitoring log (Memory)	*Manager-log-output-directory*/maintenance/watch	1 MB ×2	N
	cjfiledescriptorwatch[*n*].log	Resource depletion monitoring log (File descriptor)		1 MB ×2	N
	cjthreadwatch[*n*].log	Resource depletion monitoring log (Thread)		1 MB ×2	N

Category	File name	Contents	Output destination directory	Default size × number of files	Integrated message log/ integrated trace log
	`cjthreaddumpwatch[n].log`	Resource depletion monitoring log (Thread dump)		1 MB ×2	N
	`cjrequestqueuewatch[n].log`	Resource depletion monitoring log (HTTP request pending queue)		1 MB ×2	N
	`cjhttpsessionwatch[n].log`	Resource depletion monitoring log (Number of HTTP sessions)		1 MB ×2	N

Legend:

Y: Output to an integrated log

N: No output to an integrated log

[1-n]: Shows that a serial number in the range from 1 to *n* is used for the number of log files.

--: Not applicable

#1

The log is output in a format different than Hitachi Trace Common Library format. For details on the log in the Trace Common Library format, see *5.2 Application Server Log*.

#2

In the part of file name [*n*], the serial number from 1 to total number of specified log files is added.

#3

You can change the output destination of Management agent log trace. If the output destination of Management agent log and trace is changed, see the mngagent.log.filename key value of the mngagent.properties file (Management agent property file).

#4

This log is output if a system built with the application server by integrating with JP1 is to be operated.

#5

This log is output when a Management event is used.

#6

This log is for the Management Server of the Application Server.

Important note

In the console log, the standard output and the standard error output of the server process started by the Administration agent is output. The precautions related to console log are as follows.

- In Windows, the console log is not output for the following processes:
  Logical performance tracer
  Logical Web server
  Logical CTM domain manager
  Logical CTM
  Logical user server started indirectly
  For the startup types of the logical user server, see *8.2.19 Logical user server definition file* in the *uCosminexus Application Server Definition Reference Guide*.
- When multiple lines of information to be output to the console log are output at the same time, console log displays the information by consolidating into a single line.

- When the number of characters of the information to be output in console information exceeds 2039 characters, the information after the 2039th character is split and output to the following line.

(3) Acquiring the logs of the internal setup tool of the virtual server manager and Server Communication Agent

This subsection describes the output destination for logs of the internal setup tool of the virtual server manager and Server Communication Agent.

The following table describes the output destination for logs of the internal setup tool of the virtual server manager and Server Communication Agent, and the presence or absence of output to the integrated message log or integrated trace log.

Table 4–14: Output destination for logs of the internal setup tool of the virtual server manager and Server Communication Agent

Category	File name	Contents	Output destination directory	Default size × number of files	Integrated message log/ Integrated trace log
Internal setup tool of the virtual server manager	`rasetup[`*n*`]`[#2]`.log`	Logs for the internal setup tool of the virtual server manager	In Windows *Cosminexus-installation-directory*`\manager\setup\log` In UNIX `/opt/Cosminexus/manager/setup/log`	262144 bytes × 4	N
	`rasetup[`*n*[#2]`.log`	Maintenance logs for the internal setup tool of the virtual server manager	In Windows *Cosminexus-installation-directory*`\manager\setup\log\maintenance` In UNIX `/opt/Cosminexus/manager/setup/log/maintenance`	16777216 bytes × 4	N
Server Communication Agent	`sinaviagent[`*n*`]`[#2]`.log`	Server Communication Agent logs	*Log-output-directory-of-Server-Communication-Agent*[#1]	524288 bytes × 4	N
	`sinaviagentsv[`*n*`]`[#2]`.log`	Service logs of the Server Communication Agent		65536 bytes × 4	N
	`snactl[`*n*`]`[#2]`.log`	Start and stop command logs of the Server Communication Agent		65536 bytes × 4	N
	`sinaviagent.err`	Standard error output of the Server Communication Agent		65536 bytes × 1	N
	`sinaviagent.out`	Standard output of the Server		65536 bytes × 1	N

Category	File name	Contents	Output destination directory	Default size × number of files	Integrated message log/ Integrated trace log
		Communication Agent			
	processConsole[*n*] #2.log	Console logs		65536 bytes × 4	N
	sinaviagent[*n*] #2.log	Maintenance logs of the Server Communication Agent	In Windows *Cosminexus-installation-directory*\sinagent\log\maintenance In UNIX /opt/Cosminexus/sinagent/log/maintenance	1048576 bytes × 4	N
	sinaviagentsv[*n*] #2.log	Maintenance logs for the Server Communication Agent services		65536 bytes × 4	N
	snactl[*n*] #2.log	Maintenance logs for the start and stop commands of the Server Communication Agent		65536 bytes × 4	N
	sinaviagent.java log[*n*]. #2log	JavaVM log file of the Server Communication Agent		256 KB × 4	N

Legend:

N: No output to an integrated log.

[n]: Shows that a serial number in the range from 1 to *n* is used for the number of log files.

#1

Log-output-directory-of-Server-Communication-Agent indicates the directory specified in sinaviagent.cfg (option definition file for Server Communication Agent). The default values are *Cosminexus-installation-directory*\sinagent\log (in Windows), or /opt/Cosminexus/sinagent/log (in UNIX).

#2

In the [*n*] part of the file name, a serial number is added sequentially from 1 up to the specified number of log files.

(4) Acquiring the integrated user management log

The integrated user management trace file is output according to the settings in the com.cosminexus.admin.auth.trace.prefix option of the ua.conf file (integrated user management configuration file). For details about the ua.conf file, see *14.2.2 ua.conf (integrated user management configuration file)* in the *uCosminexus Application Server Security Management Guide*.

(5) Collecting Cosminexus JMS Provider logs

This subsection describes the collection of Cosminexus JMS Provider logs. With Cosminexus JMS Provider, you can collect CJMSP Broker logs, management command (cjmsicmd) logs, and CJMSP resource adapter logs. The following table describes the default log output destinations.

Table 4–15: Output destination of Cosminexus JMS Provider logs (Default)

Log type	Category	Default output destination	Default size ×Number of files
CJMSP Broker log	Message log	• In Windows <CJMSP_HOME>[#1]\var\instances*instanceName*\log\cjmsbroker_msg[*n*].log • In UNIX <CJMSP_HOME>[#1]/var/instances/*instanceName*/log/cjmsbroker_msg[*n*].log	1 MB ×2
	Error log	• In Windows <CJMSP_HOME>[#1]\var\instances*instanceName*\log\cjmsbroker_err[*n*].log • In UNIX <CJMSP_HOME>[#1]/var/instances/*instanceName*/log/cjmsbroker_err[*n*].log	1 MB ×2
Management command (cjmsicmd) log[#2]	Message log	• In Windows <CJMSP_HOME>[#1]\var\admin\log\cjmsadmin_msg[*n*].log • In UNIX <CJMSP_HOME>[#1]/var/admin/log/cjmsadmin_msg[*n*].log	1 MB ×2
	Error log	• In Windows <CJMSP_HOME>[#1]\var\admin\log\cjmsadmin_err[*n*].log • In UNIX <CJMSP_HOME>[#1]/var/admin/log/cjmsadmin_err[*n*].log	1 MB ×2
CJMSP resource adapter log	Message log	• In Windows *J2EE-server-log-output-directory-(ejb.server.log.directory)*[#3]\cjms\Cosminexus_JMS_Provider_RA\cjmsra_msg[*n*].log • In UNIX *J2EE-server-log-output-directory-(ejb.server.log.directory)*[#3]/cjms/Cosminexus_JMS_Provider_RA/cjmsra_msg[*n*].log	1 MB ×2
	Error log	• In Windows *J2EE-server-log-output-directory-(ejb.server.log.directory)*[#3]\cjms\Cosminexus_JMS_Provider_RA\cjmsra_err[*n*].log • In UNIX *J2EE-server-log-output-directory-(ejb.server.log.directory)*[#3]	1 MB ×2

Log type	Category	Default output destination	Default size ×Number of files
		/cjms/Cosminexus_JMS_Provider_RA/ cjmsra_err[*n*].log	

Note

In [*n*], add the file number (number of files from 1 (maximum 16)).

#1

<CJMSP_HOME> indicates the following directory:

In Windows

Cosminexus-installation-directory\CC\cjmsp

In UNIX

/opt/Cosminexus/CC/cjmsp

#2

If the log output has reached the specified maximum file size, the log output destination switches to the next file. If the file count reaches the specified number of files, the output destination switches to the first file by the wraparound method and the original information is overwritten. Note that the file is initialized only during the creation and not during the wraparound. If the file size is extended, the file is initialized only for the extended part. Initialization means writing null characters (0x20) in the area from EOF (End Of File) to the specified file size. The existing data is not affected.

#3

J2EE-server-log-output-directory (ejb.server.log.directory) is the directory specified in the J2EE server option definition. By default, the following directory is specified:

In Windows

Directory-specified-in-ejb.public.directory\ejb*J2EE-server-name*\logs

In UNIX

Directory-specified-in-ejb.public.directory/ejb/*J2EE-server-name*/logs

(6) Required information to be acquired other than a log

This section describes the required information to be acquired other than a log.

When using the in-process transaction service

When using the in-process transaction service, it is necessary to acquire the status file of the in-process transaction service. Note that you also acquire the spare status file when the status file is duplicated.

The status file is stored in the path specified in the ejbserver.distributedtx.ots.status.directory1 parameter and the ejbserver.distributedtx.ots.status.directory2 parameter (during duplication) in the <configuration> tag of the logical J2EE server (j2ee-server) in the Easy setup definition file.

4.3.2 Acquiring the Cosminexus Performance Tracer Log

This section describes the Cosminexus Performance Tracer log type and output destination of the log.

(1) Cosminexus Performance Tracer log type

In Cosminexus Performance Tracer, output the PRF daemon, PRF command log for each PRF identifier. Furthermore, output the error analysis log (various maintenance information), used by the maintenance personnel for error analysis when a problem occurs in the system, to the environment variable PRFSPOOL settings directory.

When you want to monitor the daily operations, monitor the event log and syslog.

(2) Cosminexus Performance Tracer log output destination

The output destination of the Cosminexus Performance Tracer log is as follows:

Table 4–16: Output destination of the Cosminexus Performance Tracer log

Contents	Output destination directory#1
PRF daemon and PRF command log	• In Windows *environment-variable-PRFSPOOL-settings-directory*\log\PRF-identifier\ctmlog[*n*] and event log #2 • In UNIX $PRFSPOOL/log/*PRF-identifier*/ctmlog[*n*] and syslog #3
Module trace	• In Windows *environment-variable-PRFSPOOL-settings-directory*\utt\umt • In UNIX $PRFSPOOL/utt/umt
Structured exception occurrence log (In Windows)	*Environment-variable-PRFSPOOL-settings-directory*\oslttrc
Maintenance information	• In Windows *environment-variable-PRFSPOOL-settings-directory* • In UNIX $PRFSPOOL/

#1

01 or *02* is displayed in [*n*].

#2

The messages required for operations are output. The log file output destination varies according to the Windows event log settings.

#3

The messages required for operations are output. For details on the syslog settings, see the OS documentation.

4.3.3 Acquiring the Cosminexus Component Transaction Monitor Log

This section describes the output destination of the Cosminexus Component Transaction Monitor log.

(1) Types of Component Transaction Monitor logs

With Component Transaction Monitor, the CTM daemon and the CTM command logs are output for each CTM identifier. Also, different maintenance information is output to the environment variable CTMSPOOL settings directory for the maintenance personnel to analyze the errors when a problem occurs in the system. Execute the ctmlogcat command to check the output messages.

Monitor the event log and syslog when you want to monitor the daily operations.

(2) Output destination of the Component Transaction Monitor logs

The following table describes the output destination of the Component Transaction Monitor logs.

Table 4–17: Output destination of the Cosminexus Component Transaction Monitor log

Contents	Output destination directory#1
CTM daemon and CTM command log	• In Windows *environment-variable-CTMSPOOL-settings-directory*\log*CTM-daemon-identifier*\ctmlog[*n*] and event log #2 • In UNIX $CTMSPOOL/log/*CTM-daemon-identifier*/ctmlog[*n*] and syslog #3
CTM domain daemon log	• In Windows *Environment-variable-CTMSPOOL-settings-directory*\log\ctmdmlog[n] • In UNIX $CTMSPOOL/log/ctmdmlog[n]
Module trace	• In Windows *Environment-variable-CTMSPOOL-settings-directory*\utt\umt • In UNIX $CTMSPOOL/utt/umt
Structured exception occurrence log (In Windows)	*Environment-variable-PRFSPOOL-settings-directory*\oslttrc
Maintenance information	• In Windows *environment-variable-CTMSPOOL-settings-directory* • In UNIX $CTMSPOOL

#1

01 or 02 is displayed in [*n*].

#2

The messages required for operations are output. The log file output destination varies according to the Windows event log settings.

#3

The messages required for operations are output. For details on the syslog settings, see the OS documentation.

4.3.4 Acquiring the log output in audit log

This subsection describes the log file that is output when the audit log functionality is used.

The following table describes the types and output destinations of the log output in audit logs.

Table 4–18: Output destination of the logs output in the audit log (Default)

Category	Contents	Log output destination and log file name#	Default size × number of files
Message log	Message log of the audit log	• In Windows *Cosminexus-installation-directory*\auditlog\rasmessage[*n*].log • In UNIX /opt/Cosminexus/auditlog/rasmessage[*n*].log	1MB × 4

Category	Contents	Log output destination and log file name#	Default size × number of files
Exception log	Exception information of the audit log	• In Windows *Cosminexus-installation-directory*\auditlog\rasexception[*n*].log • In UNIX /opt/Cosminexus/auditlog/rasexception[*n*].log	1MB × 8

#
In the part of the log file name [*n*], the file number (number of files from 1 (maximum 64)) is added.

4.3.5 Acquiring the Application User Log

You can acquire the application log when the settings are set so that the application log is output in the Hitachi Trace Common Library format. The user logs are in the following two types:

- User log output in the J2EE application (J2EE component) running on the J2EE server.
- User log output in the EJB client application.

For details on the log in the Trace Common Library format, see *5.2 Application Server Log*.

(1) Acquiring the user logs of J2EE application

For the log output destination of a J2EE application, the log is output to a file, which has a name containing the prefix specified in the ejbserver.application.userlog.CJLogHandler.*handler-name*.path parameter in the configuration tag of the logical J2EE server (j2ee-server) in the simple setup definition file, in the directory below. Note that the handler name is specified in the handler-name to identify the key values.

- In Windows
  Log-output-destination-root-(-ejb.server.log.directory-value-)\user\ (default is *J2EE-server-work-directory*\ejb*J2EE-server-name*\logs\user)
- In UNIX
  Log-output-destination-root-(-ejb.server.log.directory-value-)/user/ (default is *J2EE-server-work-directory*/ejb/*J2EE-server-name* /logs/user)

For details on the settings to output user logs in J2EE applications, see *8.8 Setting the user log output of J2EE applications* in the *uCosminexus Application Server Expansion Guide*.

(2) Acquiring the EJB client application user log

For the user log output destination of an EJB client application, the log is output to the file with the name having the prefix specified in the ejbserver.application.userlog.CJLogHandler.*handler-name*.path key value of the EJB client application system property in the directory below. Note that the handler name is specified in the handler-name to identify the key values.

- In Windows
  Log-output-destination-root-(-ejb.server.log.directory-value-)\user\(default is *J2EE-server-work-directory*\ejb*J2EE-server-name*\logs\user)
- In UNIX

Log-output-destination-root-(-ejb.server.log.directory-value-)/`user`/ (default is *J2EE-server-work-directory*/`ejb`/*J2EE-server-name*/`logs`/`user`)

For the method of setting the system properties to output user logs in EJB client applications, see *8.10 Setting the user log output of EJB client applications (When using the cjclstartap command)* or *8.11 Implementing and setting the user log output of EJB client applications (When using the vbj command)* in the *uCosminexus Application Server Expansion Guide*.

4.4 Application Server log (Systems for executing batch applications)

This subsection describes how to acquire logs output by the component software of the Application Server manually in a system for executing batch applications. Further, when the application server log is already collected as a snapshot log, you need not perform the tasks explained here.

This section describes how to acquire the following logs:

- Cosminexus Component Container logs
- Application user logs

Additionally, when you acquire the logs output by the component software of the Application Server manually in a system for executing the batch applications, you also acquire the following logs:

- Cosminexus Performance Tracer log
- Logs output in audit log

For details about how to acquire the above-mentioned logs, see *4.3 Application Server log (Systems for executing J2EE applications)*. The respective reference details of these logs are as follows:

- **Cosminexus Performance Tracer log**
  4.3.2 Acquiring the Cosminexus Performance Tracer Log
- **Log output in audit log**
  4.3.4 Acquiring the log output in audit log

4.4.1 Acquiring the Cosminexus Component Container Logs (systems executing batch applications)

This section describes the types and output destinations of Cosminexus Component Container logs. There are two types of Cosminexus Component Container logs:

- Batch server, server management commands, batch application logs
- Administration agent, Management agent, Management Server logs

The output destination of each of the log is described below.

(1) Acquiring the batch server, server management commands, batch application logs

This point describes how to acquire the batch server, server management commands, and batch application logs.

Furthermore, in Cosminexus Component Container, the migration command logs are output in addition to these logs. When the resource depletion monitoring functionality is used, the resource depletion monitoring log is output.

- The four types of batch server logs are message log, user log, exception log, and maintenance log. Note that in addition to these logs, event log or syslog are output in the case of starting, stopping, and abnormal termination of a batch server.
- The three types of server management command logs are message log, exception log, and maintenance log.
- The one type of batch application log is message log.

- In the resource adapter version up command (`cjrarupdate`) log, there are three types of logs, message log, exception log, and maintenance log.
- The three types of migration command logs are message log, exception log, and maintenance log.

Each of these logs is described below:

Message log

The operation status of a batch server, server management command, and migration command is output. Message log is used as the operation monitoring information for various types of servers and commands.

User log

The information of the standard output and the standard error output in the application is output in the user log. Use this log to check the operation when developing the application. Note that if you specify the `java.security.debug` property and start the server, the standard output and standard error output information is not output to the user log. This log also includes the JavaVM memory related logs.

Exception log

The exception information of Cosminexus Component Container is output when a problem occurs in the system. Note that you need not monitor the exception log in daily operations. Use this log to reference exception information if a message is output to the log.

Maintenance log

The error maintenance information of Cosminexus Component Container is output when a problem occurs in the system. Maintenance personnel use this log to analyze the errors that occur in Cosminexus Component Container.

Event log (in Windows)

The information indicating start, stop, or abnormal termination of a batch server is output to this log. The output destination differs depending on the Windows event log settings.

Note that the event log is not output depending on how the batch server has been stopped. In the following cases, there are times when the log is not output correctly:

- When a problem occurs in JavaVM itself when the batch server is running
- When the batch server process is stopped externally by TerminateProcess
- When the batch server terminates abnormally due to insufficient memory when the `-XX:+HitachiOutOfMemoryAbort` option is specified to start the JavaVM.

  Note that the `-XX:+HitachiOutOfMemoryAbort` option is set by default.

syslog (in UNIX)

The information indicating start, stop, or abnormal termination of a batch server is output to this log. The output destination differs depending on the settings in the UNIX syslog.

Note that the event log is not output depending on how the batch server has been stopped. In the following cases, there are times when the log is not output correctly:

- When a problem occurs in JavaVM itself when the batch server is running
- When the batch server process is stopped externally by the SIGKILL signal (such as kill -9)
- When the batch server terminates abnormally due to insufficient memory when the `-XX:+HitachiOutOfMemoryAbort` option is specified to start the JavaVM.

  Note that the `-XX:+HitachiOutOfMemoryAbort` option is set by default.

Resource depletion monitoring log

When using the resource depletion monitoring function, the resource depletion monitoring information about the resources being monitored is output. Use it for investigating the cause if the resource usage or the used resource quantity exceeds the threshold value.

The log is recorded in an order starting from the log file attached with the smallest file number. When one log file size attains the maximum size for one file, the log is recorded in the log file attached with the next file number. When the last log file (log file attached with the file count number) size attains the maximum size for one file, the log file of file number 1 is made empty and the log is recorded in that file. Thereafter, the log is recorded in the log files in the order of the file number by emptying those log files before recording a new log.

The following table describes the default of log output destination. The Cosminexus Component Container log can be acquired according to the server or command.

The *working-directory* shown in the log output destination indicates a directory specified in the `ejb.public.directory` parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file. The default value is *Cosminexus-installation-directory*`\CC\server\public` (in Windows) or `/opt/Cosminexus/CC/server/public` (in UNIX).

(a) Batch server log

Table 4–19: Output destination of the batch server log (Default)

Category	Contents	Log output destination and log file name[#1]	Default size × number of files	Channel name

Message log	Operation log	• In Windows
		`ejb.server.log.directory`[#2]`\cjmessage[`*n*`].log`
		• In UNIX
		`ejb.server.log.directory`[#2]`/cjmessage[`*n*`].log`
			1MB × 2	MessageLogFile

	Log operation log[#3]	• In Windows
		`<ejb.server.log.directory>`[#2]`\cjlogger.log`
		• In UNIX
		`<ejb.server.log.directory>`[#2]`/cjlogger.log`
			1 MB × 2	--

	Operation log of resource adapter that is deployed and used in batch server[#4]	• In Windows
		`ejb.server.log.directory`[#2]`\connectors\resource-adapter-display-name[`*n*`].log`
		• In UNIX
		`ejb.server.log.directory`[#2]`/connectors/resource-adapter-display-name[`*n*`].log`
			1MB × 2	--

User log	User output log	• In Windows
		`ejb.server.log.directory`[#2]`\user_out[`*n*`].log`
		• In UNIX
		`ejb.server.log.directory`[#2]`/user_out[`*n*`].log`
			1MB × 2	UserOutLogFile

	User error log	• In Windows
		`ejb.server.log.directory`[#2]`\user_err[`*n*`].log`
		• In UNIX
		`ejb.server.log.directory`[#2]`/user_err[`*n*`].log`
			1MB × 2	UserErrLogFile

Category	Contents	Log output destination and log file name[#1]	Default size × number of files	Channel name

	JavaVM maintenance information and GC log	• In Windows
		`ejb.server.log.directory`[#2]`\javalog[`*nn*`].log`
		• In UNIX
		`ejb.server.log.directory`[#2]`/javalog[`*nn*`].log`
			4MB × 4	--

	Event log of the Explicit Memory Management functionality	• In Windows
		`ejb.server.log.directory`[#2]`\ehjavalog[`*nn*`].log`
		• In UNIX
		`ejb.server.log.directory`[#2]`/ehjavalog[`*nn*`].log`
			4MB × 4	--

Exception log	Exception information when an error occurs	• In Windows
		`ejb.server.log.directory`[#2]`\cjexception[`*n*`].log`
		• In UNIX
		`ejb.server.log.directory`[#2]`/cjexception[`*n*`].log`
			1MB × 2	ExceptionLogFile

Maintenance log	Maintenance information	• In Windows
		`<ejb.server.log.directory>`[#2]`\CC\maintenance\cjmaintenance[`*n*`].log`
		• In UNIX
		`ejb.server.log.director`[#2]`/CC/maintenance/cjmaintenance[`*n*`].log`
			16MB × 4	MaintenanceLog File

	Console message	• In Windows
		`<ejb.server.log.directory>`[#2]`\CC\maintenance\cjconsole[`*n*`].log`
		• In UNIX
		`ejb.server.log.directory`[#2]`/CC/maintenance/cjconsole[`*n*`].log`
			1MB × 2	ConsoleLogFile

	EJB container maintenance information	• In Windows
		`<ejb.server.log.directory>`[#2]`\CC\maintenance\cjejbcontainer[`*n*`].log`
		• In UNIX
		`ejb.server.log.directory`[#2]`/CC/maintenance/cjejbcontainer[`*n*`].log`
			1MB × 2	EJBContainerLog File

	Start process standard output information[#5]	• In Windows
		`<ejb.server.log.directory>`[#2]`\CC\maintenance\cjstdout.log`
		• In UNIX
		`ejb.server.log.directory`[#2]`/CC/maintenance/cjstdout.log`
			--	--

Category	Contents	Log output destination and log file name[#1]	Default size × number of files	Channel name

	Start process standard error information[#5]	• In Windows
		`<ejb.server.log.directory>`[#2]`\CC\maintenance\cjstderr.log`
		• In UNIX
		`ejb.server.log.directory`[#2]`/CC/maintenance/cjstderr.log`
			--	--

	Termination process information	• In Windows
		`<ejb.server.log.directory>`[#2]`\CC\maintenance\cj_shutdown[`*n*`].log`
		• In UNIX
		`ejb.server.log.directory`[#2]`/CC/maintenance/cj_shutdown[`*n*`].log`
			4KB × 2[#6]	--

Event log	Log showing batch server start, stop or abnormal termination	Application log of Windows event viewer[#7]	--	--
syslog	Log showing batch server start, stop or abnormal termination	Depends on UNIX syslog settings. [#8]	--	--

Legend:

--: Not applicable

Note:

Channel name is the name that identifies the log output destination.

#1

In the `[`*n*`]` part of the log file name, add the file number (from 1 to the maximum number of files for each log).

Moreover, in the [*nn*] part, a serial number from 01 to 99 is added.

#2

`<ejb.server.log.directory>` indicates a directory specified in the `ejb.server.log.directory` parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file. The default value is *Cosminexus-installation-directory*`\CC\server\public\ejb\`*server-name*`\logs`.

For details about the `ejb.server.log.directory` parameter, see *3.2.1 usrconf.cfg (Option definition file for batch servers)* in the *uCosminexus Application Server Definition Reference Guide*.

#3

Check the file size when the file is output. If the maximum size is exceeded when the file size is checked, change the name of the `cjlogger.log` file to the name of the backup file (`cjlogger_save.log`).

#4

Decision about acquiring the resource adapter log depends on the contents specified in server management commands. Moreover, you can use the simple setup definition file to change the size and number of files for a resource adapter log. For details about settings for acquiring resource adapter logs, see *3.3.11 Settings for Acquiring the Resource Adapter Logs*.

#5

It is a log in which only the start process information is acquired. As it is output mainly while starting and terminating the batch server, this log is almost not output online. When the file size reaches the upper limit, it is saved in `cjstdout_save.log` or `cjstderr_save.log` under *working-directory*\ejb*server-name*\logs (in Windows) or under *working-directory*/ejb/*server-name*/logs (in UNIX). If `cjstdout_save.log` or `cjstderr_save.log` already exists, it is overwritten.

#6

The size and number of files cannot be changed.

#7

The output destination of the log file differs depending on the Windows event log settings.

#8

To output messages related to batch server start, stop, and abnormal termination, to syslog, it is necessary to set the priority for the facility daemon to info or debug in the syslog settings. For details about the syslog settings, see the manual provided with the OS.

(b) Server management command log

Table 4–20: Output destination of server management command logs (Default)

Category	Contents	Log output destination and log file name#1	Default size × number of files	Channel name
Message log	Operation log#2, #3	• In Windows *Cosminexus-installation-directory*\CC\admin\logs\cjmessage[*n*].log • In UNIX /opt/Cosminexus/CC/admin/logs/cjmessage[*n*].log	1024KB × 3	MessageLogFile
	Log operation log#2	• In Windows *Cosminexus-installation-directory*\CC\admin\logs\cjlogger.log • In UNIX /opt/Cosminexus/CC/admin/logs/ cjlogger.log	1024KB × 2	--
Exception log	Exception information when an error occurs#2, #3	• In Windows *Cosminexus-installation-directory*\CC\admin\logs\cjexception[*n*].log • In UNIX /opt/Cosminexus/CC/admin/logs/cjexception[*n*].log	1024KB × 6	ExceptionLogFile
Maintenance log	Maintenance information#2	• In Windows *Cosminexus-installation-directory*\CC\admin\logs\CC\maintenance\cjmaintenance[*n*].log • In UNIX /opt/Cosminexus/CC/admin/logs/CC/maintenance/cjmaintenance[*n*].log	1024KB × 3	MaintenanceLog File
	Console message#2	• In Windows *Cosminexus-installation-directory*\CC\admin\logs\CC\maintenance\cjconsole[*n*].log • In UNIX /opt/Cosminexus/CC/admin/logs/CC/maintenance/cjconsole[*n*].log	32KB × 3	ConsoleLogFile
	Maintenance information of the server management command#2	• In Windows *Cosminexus-installation-directory*\CC\admin\logs\CC\maintenance\cjserveradmin[*n*].log • In UNIX	32KB × 3	ServerAdminLog File

Category	Contents	Log output destination and log file name#1	Default size × number of files	Channel name
		/opt/Cosminexus/CC/admin/logs/CC/maintenance/cjserveradmin[*n*].log		

Legend:

--: Not applicable

Note:

Channel name is the name that identifies the log output destination of the log. Use it as a key value when changing log attributes (size, number of files).

#1

In the [*n*] part of the log file name, add the file number (from 1 to the maximum number of files for each log).

#2

The command name is displayed in the output message (application identification name) of the Hitachi Trace Common Library format. For details on the log in the Trace Common Library format, see *5.2 Application Server Log*.

#3

For the compatibility mode, the output destination of the operation log and the exception information when an error occurs will differ from the standard mode. For the compatibility mode, the output destination, default size, and the number of files are as follows:

Table 4–21: Output destination of server management command log (Compatibility mode)

Contents	Log output destination and log file name#	Default size × number of files
Operation log	• In Windows *Cosminexus-installation-directory*\CC\admin\logs*command-name*message[*n*].log • In UNIX /opt/Cosminexus/CC/admin/logs/*command-name*/message[*n*].log	128KB × 2
Exception information when an error occurs	• In Windows *Cosminexus-installation-directory*\CC\admin\logs*command-name*exception[*n*].log • In UNIX /opt/Cosminexus/CC/admin/logs/*command-name*/exception[*n*].log	256KB × 2

#

In the part of the log file name [*n*], the file number (number of files from 1 (maximum 16)) is added.

In the messages output to the message log of the server management commands, there are the cases when the message ID field is blank and message ID (such as KDJEnnnnn-Y) is included in the message text field. This is an additional information of the messages output before or after the message is output at the server side.

(c) Batch application log

Table 4–22: Output destination of the Batch application log (Default)

Category	Contents	Log output destination and log file name#1	Default size × number of files
Message log	Operation log of the cjexecjob command, the cjkilljob command, and the	• In Windows <batch.log.directory>#2\cjmessage[*n*].log • In UNIX batch.log.directory#2/cjmessage[*n*].log	1MB × 2

Category	Contents	Log output destination and log file name#1	Default size × number of files
	`cjlistjob` command		

#1

In the part of the log file name [*n*], the file number (number of files from 1 (maximum 16)) is added.

#2

The <`batch.log.directory`> indicates a directory specified in the `batch.log.directory` of option definition file (`usrconf.cfg` file) for batch applications. The default values are as follows:

In Windows

Cosminexus-installation-directory`\CC\batch\logs`

In UNIX

`/opt/Cosminexus/CC/admin/logs`

(d) Resource adapter version-up command (cjrarupdate) log

Table 4–23: Output destination of the resource adapter version-up command (cjrarupdate) log

Category	Contents	Log output destination and log file name#	Default size × number of files
Message log	Operation log	• In Windows *Cosminexus-installation-directory*`\CC\logs\cjrarupdatemessage[`*n*`].log` • In UNIX `/opt/Cosminexus/CC/logs/cjrarupdatemessage[`*n*`].log`	1MB × 2
Exception log	Exception information when an error occurs	• In Windows *Cosminexus-installation-directory*`\CC\logs\cjrarupdateexception[`*n*`].log` • In UNIX `/opt/Cosminexus/CC/logs/cjrarupdateexception[`*n*`].log`	1MB × 2
Maintenance log	Maintenance information	• In Windows *Cosminexus-installation-directory*`\CC\logs\cjrarupdatemaintenance[`*n*`].log` • In UNIX `/opt/Cosminexus/CC/logs/cjrarupdatemaintenance[`*n*`].log`	1MB × 2

#

In the part of the log file name [*n*], the file number (number of files from 1 (maximum 16)) is added.

(e) Migration command (cjenvupdate) log

Table 4–24: Output destination of the migration command (cjenvupdate) log

Category	Contents	Log output destination and log file name#	Default size × number of files
Message log	Operation log of the cjenvupdate command	• In Windows *Cosminexus-installation-directory*`\CC\logs\cjenvupdatemessage[`*n*`].log` • In UNIX	4MB × 4

Category	Contents	Log output destination and log file name#	Default size × number of files
		/opt/Cosminexus/CC/logs/cjenvupdatemessage[*n*].log	
Exception log	Exception information of the cjenvupdate command	• In Windows *Cosminexus-installation-directory*\CC\logs\cjenvupdateexception[*n*].log • In UNIX /opt/Cosminexus/CC/logs/cjenvupdateexception[*n*].log	4MB × 4
Maintenance log	Maintenance information of the cjenvupdate command	• In Windows *Cosminexus-installation-directory*\CC\logs\cjenvupdatemaintenance[*n*].log • In UNIX /opt/Cosminexus/CC/logs/cjenvupdatemaintenance[*n*].log	4MB × 4

#

In [*n*], the file number (number of files from 1 (maximum 16)) is attached.

(f) Resource depletion monitoring log

Table 4–25: Output destination of resource depletion monitoring log

Monitored Resources	Log acquisition location and log file name#1	Default size × number of files	Channel name
Memory	• In Windows <ejb.server.log.directory>#2\watch\cjmemorywatch[*n*].log • In UNIX ejb.server.log.directory#2/watch/cjmemorywatch[*n*].log	1MB × 2	MemoryWatchLogFile
File descriptors	• In UNIX#3 ejb.server.log.directory#2/watch/cjfiledescriptorwatch[*n*].log	1MB × 2	FileDescriptorWatchLogFile
Threads	• In Windows <ejb.server.log.directory>#2\watch\cjthreadwatch[*n*].log • In UNIX ejb.server.log.directory#2/watch/cjthreadwatch[*n*].log	1MB × 2	ThreadWatchLogFile
Thread dump	• In Windows <ejb.server.log.directory>#2\watch\cjthreaddumpwatch[*n*].log • In UNIX ejb.server.log.directory#2/watch/cjthreaddumpwatch[*n*].log	1MB × 2	ThreaddumpWatchLogFile
HTTP requests pending queue	• In Windows <ejb.server.log.directory>#2\watch\cjrequestqueuewatch[*n*].log	1MB × 2	RequestQueueWatchLogFile

Monitored Resources	Log acquisition location and log file name#1	Default size × number of files	Channel name
	• In UNIX ejb.server.log.directory#2/watch/cjrequestqueuewatch[*n*].log		
HTTP session numbers	• In Windows <ejb.server.log.directory>#2\watch\cjhttpsessionwatch[*n*].log • In UNIX ejb.server.log.directory#2/watch/cjhttpsessionwatch[*n*].log	1MB × 2	HttpSessionWatchLogFile
Connection pool	• In Windows <ejb.server.log.directory>#2\watch\cjconnectionpoolwatch[*n*].log • In UNIX ejb.server.log.directory#2/watch/cjconnectionpoolwatch[*n*].log	1MB × 2	ConnectionPoolWatchLogFile

Note:

Channel name is the name to identify the output destination of the log. Use it as a key value when changing log attributes (size, number of files).

#1

In [*n*], the file number (number of files from 1 (maximum 16)) is attached.

#2

<ejb.server.log.directory> indicates a directory specified in the ejb.server.log.directory parameter in the <configuration> tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file. The default value is *Cosminexus-installation-directory*\CC\server\public\ejb*server-name*\logs.

For details about the ejb.server.log.directory parameter, see *3.2.1 usrconf.cfg (Option definition file for batch servers)* in the *uCosminexus Application Server Definition Reference Guide*.

#3

The file descriptor cannot be monitored in Windows and AIX.

For details about the information output to the resource depletion monitoring log file and the output format of the log file, see *4.3 Resource depletion monitoring functionality and output of resource depletion monitoring information* in the *uCosminexus Application Server Operation, Monitoring, and Linkage Guide*.

(g) User definition file to set output destination of the log

When the batch server and server management command log output destination is changed, reference the user definition file in which the log output destination is set, described in the following table and confirm the output destination. Note that if the output destination of a log is changed, that log is not collected when the snapshot log is collected in a batch. Change the collection destination of the snapshot log as and when required.

Table 4–26: User definition file in which the output destination of log is set

Category	Location specified for user definition file
Batch server	The ejb.server.log.directory parameter specified in the <configuration> tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file The default is *Cosminexus-installation-directory*\CC\server\public\ejb*server name*\logs (in Windows) or /opt/Cosminexus/CC/server/public/ejb/*server-name*/logs (in UNIX).
Server management commands	ejbserver.log.directory key of the server management command usrconf.bat (in Windows) or usrconf (in UNIX)

Category	Location specified for user definition file
	The default key is *Cosminexus-installation-directory*\CC\admin\logs (in Windows) or /opt/Cosminexus/CC/admin/logs (in UNIX). You cannot change the output destination of the server management command logs when operating from the Management Server Remote Management functionality.

For details about settings of the data acquisition for troubleshooting, such as how to change the output destination of log, see *3. Preparing for Troubleshooting*.

(2) Acquiring the log of Administration agent, Management agent, and Management Server

This point describes the output destination of Administration Agent, Management Agent, and the Management Server log.

For the Administration Agent, Management Agent, and the Management Server logs, besides acquiring separately, you can also acquire these logs by compiling as a **integrated log**.

For details about the output destination when acquiring the Administration Agent, Management Agent, and the Management Server log by compiling as an integrated log or separately, see *4.3.1(2) Acquiring the log of Administration agent, Management agent, and Management Server*.

(3) Required information to be acquired other than a log

For details about the required information to be acquired other than a log, see *4.3.1(6) Required information to be acquired other than a log*.

4.4.2 Acquiring the Application User Log (systems executing batch applications)

You can acquire the user log of the batch application when the settings are set so that the user log is output in the Hitachi Trace Common Library format. For details on the log in the Trace Common Library format, see *5.2 Application Server Log*.

For the *output destination of a batch application user log*, the log is output to a file that has a name containing the prefix specified in the ejbserver.application.userlog.CJLogHandler.*handler-name*.path parameter within the <configuration> tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file, in the following directory. Note that the handler name is specified in the handler-name to identify the key values.

- In Windows

  Log-output-destination-root-(-ejb.server.log.directory-)\user\ (default is *batch-server-work-directory*\ejb*batch-server-name*\logs\user)

- In UNIX

  log-output-destination-root-(-ejb.server.log.directory-value-)/user/(default is *batch-server-work-directory*/ejb/*batch-server-name*/logs/user)

For details about settings to output batch application user logs, see *2.3.5 Log output of a batch application* in the *uCosminexus Application Server Expansion Guide*.

4.5 EJB Client Application System Log

This section describes the system log types and output destination of the EJB client application.

Note that for the precautions to be taken when working with the EJB client application system log, see *2.6.1 Precautions Related to the System Log of an EJB Client Application*.

Important note

When using the uCosminexus Client, read the *Cosminexus-installation-directory*\CC of the EJB client application log save directory as *Cosminexus-installation-directory*\CCL.

4.5.1 Types of EJB Client Application System Logs

In an EJB client application, output the system log for every EJB client application process. The system log types of an EJB client application are as follows:

Table 4–27: Types of EJB client application system logs

Types	Log contents	File name	Default size × number of files	Channel name
Message log	Operation log	cjclmessage[*n*].log[#1]	1MB × 2	ClientMessageLogFile
	cjclstartap command operation log	cjclstartap[*n*].log[#2]	1MB × 2	--
	cjcldellog command operation log	cjcldellog.log	1MB × 2[#3]	--
User log	User output log	user_out[*n*].log[#1]	1MB × 2	UserOutLogFile
	User error log	user_err[*n*].log[#1]	1MB × 2	UserErrLogFile
Java log	JavaVM maintenance information, GC log	javalog[*n*].log[#1]	256KB × 4	--
Exception log	Exception information when an error occurs	cjclexception[*n*].log[#1]	1MB × 2	ClientExceptionLogFile
Maintenance log[#4]	Maintenance information	cjclmaintenance[*n*].log[#1]	1MB × 2	ClientMaintenanceLogFile
	EJB container maintenance information	cjejbcontainer[*n*].log[#1]	1MB × 2	EJBContainerLogFile
	Start process standard output information	cjstdout[*n*].log[#2]	1MB × 2	--
	Start process standard error information	cjstderr[*n*].log[#2]	1MB × 2	--
	Log operation information	cjlogger.log	1MB × 2[#3]	--

Legend:

--: Not applicable

#1

In the part of file name [*n*], the serial number from 1 to total number of specified log files is added.

#2

In the part of file name [*n*], number (1 or 2) of the number of specified log files is attached.

#3

When the size of cjcldellog.log exceeds 1 MB, it is renamed to backup log file called cjcldellog_save.log.

#4

Collect this log if it needs to be sent to maintenance personnel.

4.5.2 Output Destination of the EJB Client Application System Log

The EJB client application system log is output to the directory specified in the following keys:

- Directory specified in the `ejb.client.log.directory` key and `ejb.client.ejb.log` key of option definition file (`usrconf.cfg`) for Java applications
- Directory specified in the `ejbserver.client.log.directory` key and `ejbserver.client.ejb.log` key in system properties

When different values are specified in both, the Java application option definition file (`usrconf.cfg`) and system properties, system property settings are valid.

Note that in system properties, you can set the output destination for the following logs only:

- Operation log (cjclmessage[*n*].log)
- Exception information when an error occurs (cjclexception[*n*].log)
- Maintenance information (cjclmaintenance[*n*].log)
- Log operation information (cjlogger.log)

The cjcldellog command operation log (cjcldellog.log) and cjclstartap command operation log (cjclstartap[*n*].log) are output directly under *Cosminexus-installation-directory*`\CC\client\logs` (in Windows) or `/opt/Cosminexus/CC/client/logs` (in UNIX).

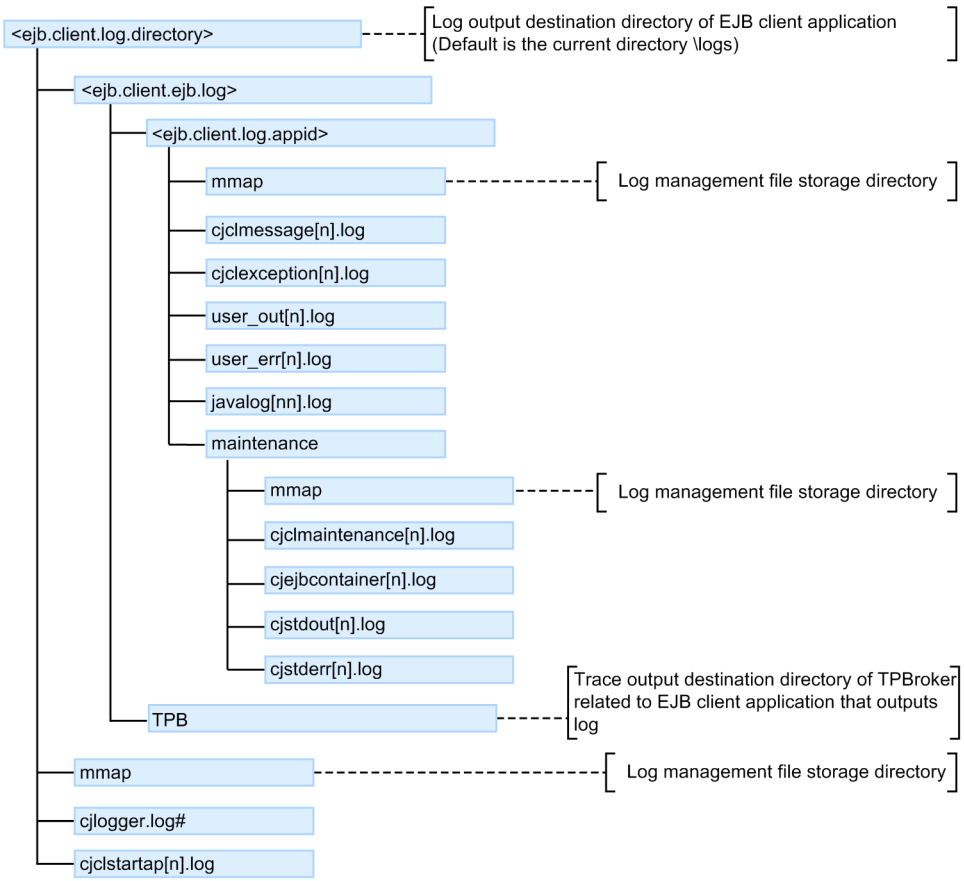
Important note

- The standard output and the standard error contents of an EJB client application are not output to the log file. To output them to the log file, either uses the user log function or redirect.

Logs for multiple processes are output under a single directory (the default is the `ejbcl` directory). Note that, if you want to have a separate log output destination for each process (each application), specify the respective output destinations in the `ejb.client.log.directory` key of `usrconf.cfg`.

The following figure shows the output destination of the EJB client application system log.

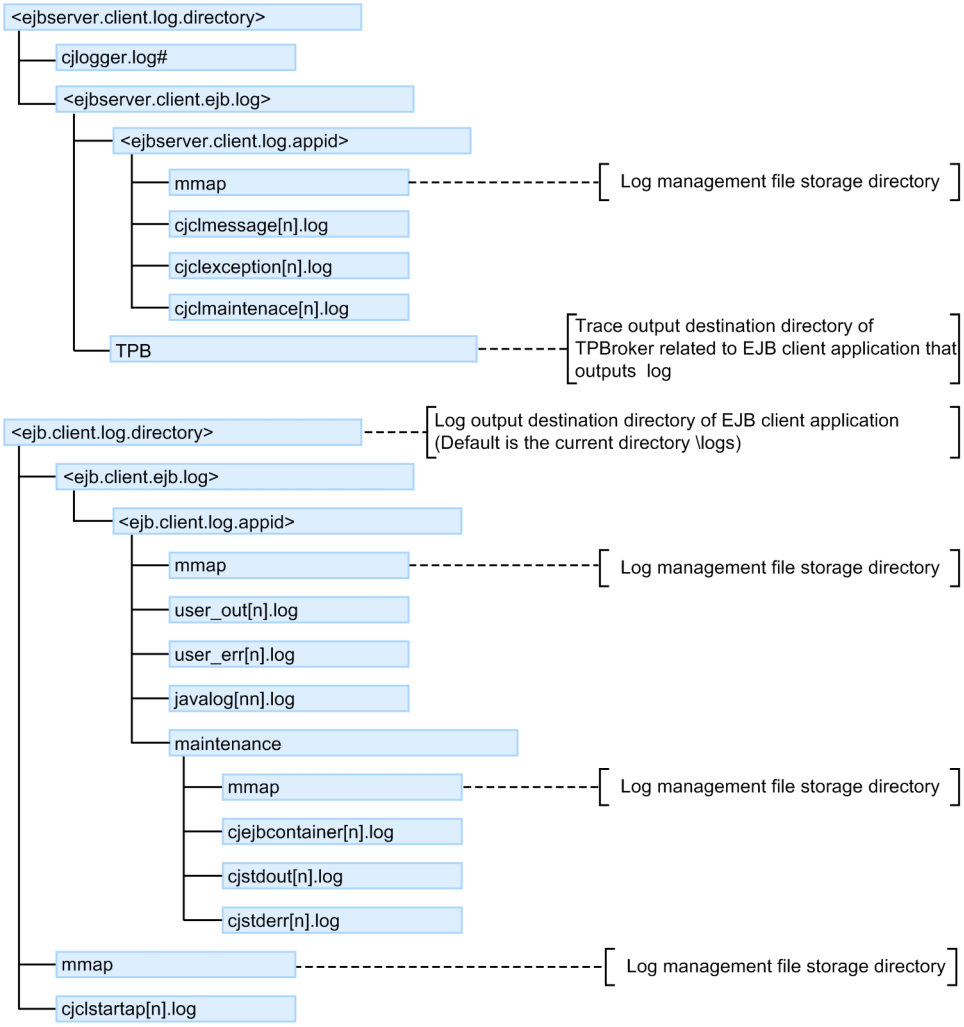
Figure 4–1: Output destination of the EJB client application system log (When specified in the option definition file for Java applications)



Note
[n] of the file name becomes the serial number of 1 ~ (Number of files of the specified log).

#
cjlogger.log is output only if there is an error in the settings of the properties of the EJB client application.

Figure 4–2: Output destination of the EJB client application system log (When specified in system property)



Note
[n]of the file name becomes the serial number of 1 ~ (Number of files of the specified log).
#
cjlogger.log is output only if there is an error in the settings of the properties of the EJB client application.

4.6 Trace based performance analysis

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 about how to collect the trace based performance analysis, see *7. Performance Analysis by Using Trace Based Performance Analysis*. The session trace may also be output to the trace based performance analysis.

4.7 JavaVM thread dump

This section describes the method of acquiring the JavaVM thread dump.

You can use the following method to acquire the JavaVM thread dump:

- Acquire a thread dump of J2EE server, CORBA naming service, and CTM using the management command (`mngsvrutil`).
- Acquire a thread dump of J2EE server, CORBA naming service, and EJB client application using separate commands.
- Acquire the class-wise statistical information or Explicit heap detail information in an extended thread dump using the JavaVM commands.

Important note

Obtain the thread dump after the output of the previous thread dump ends.

Each method is described below:

4.7.1 When using the management command

To acquire a thread dump of JavaVM using the management command (`mngsvrutil`), specify server in the argument of the sub command dump of the command mngsvrutil and execute the command.

The following thread dump can be acquired:

- **J2EE server (including cluster)**
- **CORBA naming service and CTM**

For the `mngsvrutil` command, see *mngsvrutil (Management Server management command)* in the *uCosminexus Application Server Command Reference Guide*. For the sub commands of the `mngsvrutil` command, see *7.3 Details of subcommands of the mngsvrutil command* in the *uCosminexus Application Server Command Reference Guide*.

The execution format, an execution example, and output destination for `mngsvrutil` command are as follows:

Execution format

```
mngsvrutil -m Management-Server-host-name[:port-number]-u management-user
-ID -p management-password -t logical-server-name dump server
```

Execution example

```
mngsvrutil -m mnghost -u user01 -p pw1 -t myserver dump server
```

Output destination

When the target is a J2EE server

- In Windows

  working-directory#`\ejb\`*server-name*`\javacore*.txt`
- In UNIX

working-directory#/ejb/*server-name*/javacore*.txt

The *working-directory* indicates a directory specified in the user definition of a J2EE server (ejb.public.directory in the usrconf.cfg file). The default values are as follows:

- In Windows
  Cosminexus-installation-directory\CC\server\public
- In UNIX
  /opt/Cosminexus/CC/server/public

When the target is a CORBA naming service and CMT

- In Windows
  Cosminexus-installation-directory\TPB\logj\javacore*.txt
- In UNIX
  /opt/Cosminexus/TPB/logj/javacore*.txt

4.7.2 When using separate commands

The method to output a thread dump differs depending on the JavaVM start option in which the specified J2EE server is running.

- When -XX:+HitachiThreadDump is set, you can acquire the *extended thread dump*. This option is set by default.
- If -XX:+HitachiThreadDumpToStdout is set, a thread dump is also output to the standard output. This option is not set by default. Set this option if required.
- When the following startup options are set, if -XX:+HitachiOutOfMemoryAbortThreadDump is set, a thread dump is output when forced termination is performed using OutOfMemoryError
  - -XX:+HitachiOutOfMemoryAbort
  - -XX:+HitachiThreadDump

  Except in the following cases:
  - When C heap is insufficient in the J2SE class library
  - When C heap is insufficient in JavaVM processing
- When the following startup options are set, if -XX:+HitachiOutOfMemoryAbortThreadDumpWithJHeapProf is set, a class wise statistical information is output in the thread dump when forced termination is performed using OutOfMemoryError. For details about the class-wise statistical information, see *9.3 Class-wise statistical functionality*. The class-wise statistical information, which is output while setting this above-mentioned option, is same as the information acquired when executing the jheapprof command.
  - -XX:+HitachiOutOfMemoryAbort
  - -XX:+HitachiOutOfMemoryAbortThreadDump
  - -XX:+HitachiThreadDump

(1) Acquiring J2EE server thread dump

If the J2EE server process (cjstartsv) exists, the J2EE server thread dump is acquired by executing the cjdumpsv command. An example of executing the cjdumpsv command is described below. As the transition status of each thread

is to be confirmed over a period of time, execute the cjdumpsv command multiple times. Execute about 10 times every 3 seconds.

- In Windows

```
Cosminexus-installation-directory\CC\server\bin\cjdumpsv J2EE-server-name
```

- In UNIX

```
/opt/Cosminexus/CC/server/bin/cjdumpsv J2EE-server-name
```

When you execute the cjdumpsv command, the JavaVM thread dump is output to the following files:

- Server standard output log
- *working-directory*\ejb*server-name*\javacore*process-number*.*command-execution-date-and-time*.txt (in Windows)
- *working-directory*/ejb/*server-name*/javacore*process-number*.*command-execution-date-and-time*.txt (in UNIX)

The default output destination of the server standard output log is ejb.server.log.directory\CC\maintenance\cjstdout.log (in Windows) or ejb.server.log.directory/CC/maintenance/cjstdout.log (in UNIX). For details about changing the output destination, see *4.3 Application Server log (Systems for executing J2EE applications)* or see *4.4 Application Server log (Systems for executing batch applications)*. Note that the default directory path of the working directory is *Cosminexus-installation-directory*\CC\server\public (in Windows) or /opt/Cosminexus/CC/server/public (in UNIX).

Furthermore, the output destination of the javacore *process-number*.*command-execution-date-and-time*.txt file can be changed with the environmental variable JAVACOREDIR. However, when writing to a specified directory fails, javacore *process-number*.*command-execution-date-and-time*.txt file is output to the default output destination. When it is not possible to output even to this directory, it is output only to the standard error output.

For details about the cjdumpsv command, see *cjdumpsv (get thread dump of J2EE server)* in the *uCosminexus Application Server Command Reference Guide*.

Reference note

When a thread dump is output, the following message is output to the standard output and the execution of the java program is continued. This message is output irrespective of the -XX:+HitachiThreadDumpToStdout settings.

```
Writing Java core to full-file-path-name...OK
```

(2) Acquiring the CORBA naming service thread dump

In Windows, when the CORBA naming service process (nameserv) exists, press the Ctrl+Break key on the command prompt from where the CORBA naming service is started. As the transition status of each thread is to be confirmed over a period of time, execute it multiple times. Execute about 10 times every 3 seconds. Note that you cannot acquire a thread dump if the CORBA naming service is monitored from the Management Server.

In UNIX, if the CORBA naming service process (java) exists, execute the `kill` command, and acquire a thread dump of the CORBA naming service. Note that you cannot acquire a thread dump if the CORBA naming service is monitored from the Management Server.

To acquire a thread dump of the CORBA naming service in UNIX:

1. Acquire the process ID of the CORBA naming service.
   The method of acquiring the process ID of the CORBA naming service differs depending on the following cases:

   When no other java process is started

   ```
   ps -ef | grep java
   ```

   When multiple java processes are started

   If the shell script for starting the CORBA naming service is used, you can output the process ID of the CORBA naming service in the `namesv_pid` file generated in the current working directory.

   An example of the shell script for starting the CORBA naming service is as follows:

   ```
   #!/bin/sh
   export VBROKER_ADM=/opt/Cosminexus/TPB/adm
   export SHLIB_PATH="${SHLIB_PATH}:/opt/Cosminexus/TPB/lib"

   # start name server process
   exec /opt/Cosminexus/TPB/bin/nameserv \
   -J-Dvbroker.agent.enableLocator=false \
   -J-Djava.security.policy==/opt/Cosminexus/CC/server/sysconf/cli.polic
   y \
   -J-Dvbroker.se.iiop_tp.scm.iiop_tp.listener.port=900 &

   # save background java process pid
   echo $! > ./namesv_pid
   ```

2. Specify the acquired process ID, and execute the kill command.

   ```
   kill -3 `cat namesv_pid`
   ```

(3) Acquiring a thread dump of EJB client applications

Acquire a thread dump of EJB client applications by executing the `cjcldumpap` command.

When the `cjcldumpap` command is executed, a thread dump of the EJB client application started by executing the `cjclstartap` command is output. Moreover, it is also possible to output a thread dump for a specific process. For details about the `cjcldumpap` command, see *cjcldumpap (get thread dump of Java application)* in the *uCosminexus Application Server Command Reference Guide*.

The execution format, execution example, and thread dump output destination of the `cjcldumpap` command are as follows:

Execution format

To output a thread dump of an EJB client application started with the `cjclstartap` command

```
cjcldumpap
```

To output a thread dump for a specific process

```
cjcldumpap process-ID
```

Execution example

To output a thread dump of an EJB client application started with the `cjclstartap` command

```
cjcldumpap
```

To output a thread dump for a specific process

```
cjcldumpap 3264
```

Output destination

Current directory from where the `cjclstartap` command is executed

4.7.3 When using JavaVM commands

When using the JavaVM commands, you can acquire the class-wise statistical information or Explicit heap detail information with the extended thread dump.

(1) Acquiring an extended thread dump of class-wise statistical information

Execute the `jheapprof` command to acquire an extended thread dump of the class-wise statistical information. The class-wise statistical information is output using the class-wise statistic functionality, and this information is used for changing Java objects and checking the Java object reference depending on the GC. For details about the class-wise statistical functionality, see the section *9.3 Class-wise statistical functionality* and for details about how to output the class-wise statistical information, see the subsection *9.3.3 Outputting Statistic Information for Each Class*. For the `jheapprof` command, see *jheapprof(Output of extended thread dump containing Hitachi class-wise statistical information)* in the *uCosminexus Application Server Command Reference Guide*.

(2) Acquiring an extended thread dump of Explicit heap detail information

Among the Explicit heap detail information, if the Explicit Memory Management functionality is valid, acquire the following information in the extended thread dump:

- Explicit heap information
- Explicit memory block information

Acquire the object statistical information of the Explicit heap detail information and the release rate information of the Explicit memory block with the extended thread dump by executing the `eheapprof` command. The object statistical information is the detailed information in Explicit memory blocks. The release rate information is the ratio of objects released in the automatic release processing of the Explicit memory block. This information is used for debugging and for error analysis of the system that uses the Explicit Memory Management functionality.

For details about the Explicit Memory Management functionality, see *7. Suppression of Full GC by Using the Explicit Memory Management Functionality* in the *uCosminexus Application Server Expansion Guide*. For details about the `eheapprof` command, see *eheapprof (Output of extended thread dump containing the Explicit heap detailed information)* in the *uCosminexus Application Server Command Reference Guide*.

Execution format

In Windows

```
eheapprof [-f|-i] [-freeratio] -p process-ID
```

In UNIX

```
eheapprof [-f|-i] [-freeratio] [-force] -p process-ID
```

Execution example

Here, the class-wise statistical information of Java process with process ID 2463 is output.

1. In the `-p` option, specify the process ID of Java process to output class-wise statistical information and then execute the `eheapprof` command.

   ```
   % eheapprof -p 2463
   ```

   When the `-f` option is omitted using the `eheapprof` command, the following confirmation message is displayed:

   In Windows

   The confirmation message whether to output an extended thread dump with Hitachi class-wise statistical information is displayed in the following format:

   ```
   Force VM to output ExplicitHeapProf: ? (y/n)
   ```

   In UNIX

   Confirmation message of process ID is displayed in the following format:

   ```
   send SIGQUIT to 2463: ? (y/n)
   ```

2. Enter y

   If the extended thread dump with Hitachi class-wise statistics is output, the following message is output in the running java program:

   ```
   Writing Java core to javacore2463.030806215140.txt... OK
   ```

   The running java program creates an extended thread dump with Hitachi class-wise statistics (`javacore<process ID>.<date-time>.txt`) in the current directory to continue the program.

4.7.4 Precautions to be taken when class-wise statistical information is output in the thread dump

If `-XX:+HitachiOutOfMemoryAbortThreadDumpWithJHeapProf` is set up in the JavaVM startup option, a class-wise statistical information is output in the thread dump when forced termination is performed using OutOfMemoryError.

When you use a large amount of heap (Java heap, Explicit heap, or Metaspace area), it takes time to output the class-wise statistical information in the thread dump. Therefore, when OutOfMemoryError occurs, a JavaVM might not be able to be recovered at once if terminated forcibly.

Important note

If the following startup options of JavaVM are not set, the class-wise statistical information is not output in the thread dump:

- `-XX:+HitachiOutOfMemoryAbort`
- `-XX:+HitachiOutOfMemoryAbortThreadDump`
- `-XX:+HitachiThreadDump`

4.8 JavaVM GC Log

The JavaVM GC log can be acquired only when the log output destination is set before starting JavaVM or the J2EE server.

The output destination of the GC log is specified in the `ejb.server.log.directory` parameter in the `<configuration>` tag of logical J2EE server (j2ee-server), in the Easy Setup definition file.

Also, when you want to perform a Full GC for the running Java processes, execute the `javagc` command. The `javagc` command can also be used for measuring the memory used for one transaction, investigating memory leak, and application debugging other than for investigating the cause when a problem occurs.

The execution format of the `javagc` command when performing a Full GC by specifying a running Java process is as follows. Note that for other options that can be specified, see *javagc (forcibly perform GC)* in the *uCosminexus Application Server Command Reference Guide*.

Execution format

```
javagc -p process-ID
```

The following log is output as the execution result. Note that in this example, `-XX:+HitachiVerboseGCPrintCause` has been specified as the option.

```
[VGC]<Wed Mar 17 00:42:30 2004>(Skip Full:0,Copy:0)[ Full GC 149K->149K(19
84K),0.0786038
secs] [ DefNew::Eden:264K->0K(512K)] [ DefNew::Survivor:0K->63K(64K)] [ Te
nured:
85K->149K(1408K)] [Metaspace: 3634K(4492K, 4492K)->3634K(4492K, 4492K)][cl
ass space: 356K(388K, 388K)->356K(388K, 388K)] [ cause:JavaGC Command]
```

4.9 Memory Dump

When restarting a J2EE server or CORBA naming service, acquire the following files as memory dump:

- User dump (in Windows)
- J2EE server memory dump
- CORBA naming service memory dump
- Management Server memory dump
- Administration Agent memory dump

These files are used by maintenance personnel for analyzing errors if a problem occurs in the system.

4.9.1 Acquiring a User Dump (In Windows)

This subsection describes the collection of user dumps.

Use the task manager or a Windows debug tool to obtain the user dump. For details, see the Microsoft website.

Alternatively, use the `-fd` option of the `cjstopsv` command to obtain the following files:

User-dump-output-destination-directory`\cjmemdump.dmp`

Specify the *user-dump-output-destination-directory* in the environment variable `CJMEMDUMP_PATH`.

4.9.2 Acquiring J2EE Server Memory Dump

This section describes how to acquire a J2EE server memory dump for each OS.

(1) In Windows

If the J2EE server is running (if the `cjstartsv` process exists), collect the memory dump from the task manager#.

If the J2EE server is running and you want to forcibly stop it and acquire the memory dump, execute the `cjstopsv` command with the `-fd` option specified.

If the J2EE server is down, collect the memory dump from the Windows debug tool#.

#
For details, see the Microsoft website.

To acquire the memory dump when the J2EE server is down, you need to specify settings in advance. For details on how to specify the settings, see *3.3.15 Settings for Collecting a User Dump*.

(2) In UNIX

When the cjstartsv process is down, acquire a core dump output in the *working-directory*`/ejb/`*server-name*.

When restarting the `cjstartsv` process, the names of the core dump files are renamed in `core.`*output-date-time*# (in AIX) or `core.`*process-ID*`.`*output-date-and-time*# (in Linux). The core dumps are not saved by overwriting when re-starting the `cjstartsv` process, therefore, you can save the core dumps generated when errors occur.

#

The output date and time is output in the *YYMMDDhhmmss* format.
YY: Western calendar year (Last 2 digits) *MM*: Month (2 digits) *DD*: Day (2 digits)
hh: Hour (2 digits in 24 hour notation) mm: Minute (2 digits) ss: Seconds (2 digits)

Note that you can set the upper limit value for the core dumps to be saved. In Windows, when restarting the `cjstartsv` process and executing the `javacore` command, starting from the oldest, the core dump is deleted in the order of output date and time. In UNIX, when the total size of the core dump files output to the *working-directory*`/ejb/`*server-name* exceeds the upper limit value, the core dump files are deleted in the order of output date and time starting from the oldest. The upper limit value is specified in `ejb.server.corefilenum` of the J2EE server extension parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file. Note that the files are deleted during restart of the `cjstartsv` process. For details about setting the upper limit value for number of core files, see *3.3.16 Settings for Acquiring a Core Dump*.

After acquiring a core dump, execute the javatrace command to acquire only the stack trace information from the core dump. The stack trace information is the information required for investigating the cause of abnormal termination of JavaVM. For details about how to acquire the stack trace information, see the section *4.18 JavaVM stack trace information*.

You can acquire a core dump in the following cases. How to acquire a core dump for each case is described below:

- Acquiring a core dump when the cjstartsv process is running

  Acquiring a core dump when the cjstartsv process (J2EE server) is running, confirm the process ID of the `cjstartsv` process and execute the `kill` command. Execute the `kill` command in the following format. Note that the process terminates when the `kill` command is executed. Hitachi, therefore, recommends that you execute the kill command before restarting.

  ```
  kill -6 cjstartsv-process-ID
  ```

- Acquiring a core dump and thread dump concurrently in a running Java process

  Execute the `javacore` command to acquire a core dump and thread dump concurrently in the running Java process. The execution format of the `javacore` command is described below. Note that for the options that can be specified, see *javacore (Acquiring the core file and thread dump/in UNIX)* in the *uCosminexus Application Server Command Reference Guide*.

  ```
  javacore process-ID
  ```

  The following message is output if you execute the command in the above-mentioned format.

  ```
  send SIGQUIT to 8662: ? (y/n)
  ```

  If you enter `y`, `javacore` *process-ID*`.`*output-date-and-time*`.core` (core dump) and `javacore` *process-ID*`.`*output-date-and-time*`.txt` (thread dump) is output to the current directory from where the Java program is being executed. If you enter `n`, the command is terminated without acquiring the core dump and thread dump.

  When acquiring a core dump and thread dump, the following message is output by the Java program being executed. Note that the information in Italics is not actually displayed.

  ```
  Now generating core file (javacore8662.030806215140.core)...
  done
  ```

```
(output of core dump and thread dump terminated)

Writing Java core to javacore8662.030806215140.txt... OK
```

4.9.3 Acquiring the CORBA Naming Service Memory Dump

How to acquire the CORBA naming service memory dump for each OS is described below.

(1) In Windows

If the CORBA naming service is running (if the CORBA naming service process exists), collect the memory dump from the task manager#.

If the CORBA naming service is down, collect the memory dump from the Windows debug tool#.

#

For details, see the Microsoft website.

To acquire the memory dump when the CORBA naming service is down, you need to specify settings in advance. For details on how to specify the settings, see *3.3.15 Settings for Collecting a User Dump*.

(2) In UNIX

When the CORBA naming service is running (when the CORBA naming service process exists), confirm the process ID of the CORBA naming service and execute the kill command. Execute the `kill` command in the following format. Note that the process terminates when the `kill` command is executed. Hitachi, therefore, recommends that you execute the `kill` command before restarting.

```
ps -ef | grep java
kill -6 CORBA-naming-service-process-ID
```

4.9.4 Acquiring the Management Server Memory Dump

This section describes how to acquire the Management Server memory dump for each OS.

(1) In Windows

If Management Server is running (if the `cjstartsv.exe` process exists), collect the memory dump from the task manager#.

If Management Server is down, collect the memory dump from the Windows debug tool#.

#

For details, see the Microsoft website.

To acquire the memory dump when Management Server is down, you need to specify settings in advance. For details on how to specify the settings, see *3.3.15 Settings for Collecting a User Dump*.

(2) In UNIX

If Management Server (the `cjstartsv` process) is down, acquire the core dump output in *Application-Server-installation-directory*`/manager/containers/m/ejb/`*server-name-of-Management-Server*.

When Management Server is restarted, the names of the core dump files are renamed to `core.`*output-date-time*# (in AIX and HP-UX) or `core.`*process-ID*`.`*output-date-and-time*# (in Linux). The core dumps are not saved by overwriting when Management Server is restarted, so you can save the core dumps generated when errors occur.

#

The output date and time is output in the *YYMMDDhhmmss* format.

YY: Western calendar year (Last 2 digits), *MM*: Month (2 digits), *DD*: Day (2 digits)

hh: Hour (2 digits in 24 hour notation), *mm*: Minute (2 digits), *ss*: Seconds (2 digits)

After acquiring the core dump, if you want to acquire only the stack trace information from the core dump, execute the `javatrace` command. The stack trace information is the information required for investigating the cause of abnormal termination of JavaVM. For details about how to acquire the stack trace information, see *4.18 JavaVM stack trace information*.

You can acquire a core dump in the following cases. The following describes how to acquire a core dump for each case:

- **Acquiring a core dump when Management Server is running**

  To acquire a core dump while Management Server is running, check the process ID of the `cjstartsv` process and execute the `kill` command. Execute the `kill` command in the following format. Note that the process terminates when the `kill` command is executed. Therefore, we recommend that you execute the `kill` command before Management Server is restarted.

  ```
  kill -6 Management-Server-(cjstartsv)-process-ID
  ```

- **Acquiring a core dump and thread dump concurrently in a running Java process**

  Execute the `javacore` command to acquire a core dump and thread dump concurrently in the running Java process. The execution format of the `javacore` command is described below. For details about the options that can be specified, see *javacore (Acquiring the core file and thread dump/in UNIX)* in the *uCosminexus Application Server Command Reference Guide*.

  ```
  javacore -p process-ID
  ```

  The following message is output if you execute the command in the above format.

  ```
  send SIGQUIT to 8662: ? (y/n)
  ```

  If you enter `y`, `javacore`*process-ID*`.`*output-date-and-time*`.core` (core dump) and `javacore`*process-ID*`.`*output-date-and-time*`.txt` (thread dump) are output to the current directory from where the Java program is being executed. If you enter `n`, the command is terminated without acquiring the core dump and thread dump.

  When the core dump and thread dump are acquired, the following message is output to the running Java program. Note that the information in italics is not actually displayed.

  ```
  Now generating core file (javacore8662.030806215140.core)...
    done

  (End of core dump and thread dump output)

  Writing Java core to javacore8662.030806215140.txt... OK
  ```

4.9.5 Acquiring the Administration Agent Memory Dump

This section describes how to acquire the Administration Agent memory dump for each OS.

(1) In Windows

If Administration Agent is running (if the `adminagent` process exists), collect the memory dump from the task manager#.

If Administration Agent is down, collect the memory dump from the Windows debug tool#.

#

For details, see the Microsoft website.

To acquire the memory dump when Administration Agent is down, you need to specify settings in advance. For details on how to specify the settings, see *3.3.15 Settings for Collecting a User Dump*.

(2) In UNIX

If Administration Agent (the `adminagent` process) is down, acquire the core dump output in *Application-Server-installation-directory*`/manager/bin`.

After acquiring a core dump, if you want to acquire only the stack trace information from the core dump, execute the `javatrace` command. The stack trace information is the information required for investigating the cause of abnormal termination of JavaVM. For details about how to acquire the stack trace information, see *4.18 JavaVM stack trace information*.

You can acquire a core dump in the following cases. The following describes how to acquire a core dump for each case:

- **Acquiring a core dump when Administration Agent is running**
  To acquire a core dump while Administration Agent is running, check the process ID of the `adminagent` process and execute the `kill` command. Execute the `kill` command in the following format. Note that the process terminates when the `kill` command is executed. Therefore, we recommend that you execute the `kill` command before Administration Agent is restarted.

  ```
  kill -6 Administration-Agent-(adminagent)-process-ID
  ```

- **Acquiring a core dump and thread dump concurrently in a running Java process**
  Execute the `javacore` command to acquire a core dump and thread dump concurrently in the running Java process. The execution format of the `javacore` command is described below. For details about the options that can be specified, see *javacore (Acquiring the core file and thread dump/in UNIX)* in the *uCosminexus Application Server Command Reference Guide*.

  ```
  javacore -p process-ID
  ```

  The following message is output if you execute the command in the above format.

  ```
  send SIGQUIT to 8662: ? (y/n)
  ```

  If you enter `y`, `javacore`*process-ID*`.`*output-date-and-time*`.core` (core dump) and `javacore`*process-ID*`.`*output-date-and-time*`.txt` (thread dump) are output to the current directory from where the Java program is being executed. If you enter `n`, the command is terminated without acquiring the core dump and thread dump.

  When the core dump and thread dump are acquired, the following message is output to the running Java program. Note that the information in italics is not actually displayed.

```
Now generating core file (javacore8662.030806215140.core)...
  done

(End of core dump and thread dump output)

Writing Java core to javacore8662.030806215140.txt... OK
```

4.9.6 Notes on obtaining the memory dump

(1) In Windows

- In the following cases, user dumps are not output.
  - Forceful termination by specifying the `-XX:+HitachiOutOfMemoryAbort` option
  - Forceful termination due to an insufficient C heap during JavaVM processing

4.10 JavaVM log (JavaVM log file)

The JavaVM log is a log that you can acquire with the extension option added by Hitachi in the standard JavaVM. You can acquire more troubleshooting information as compared to the information acquired from the standard JavaVM. The JavaVM log is output to the log file, when any of the following options are specified. Note that this log file is called the *JavaVM log file*.

- `-XX:+HitachiOutOfMemoryStackTrace`

  Note that the JavaVM log file is output even if you specify `-XX:+HitachiOutOfMemorySize` and `-XX:+HitachiOutOfMemoryCause` that are specified when you specify `-XX:+HitachiOutOfMemoryStackTrace` option.
- `-XX:+HitachiVerboseGC`
- `-XX:+HitachiOutOfMemoryHandling`
- `-XX:+HitachiJavaClassLibTrace`
- `-XX:+JITCompilerContinuation`

Other than this, you need to specify in options even for the output contents and output method. For details about settings to acquire the JavaVM material, see the subsection *3.3.17 Settings for Acquiring the JavaVM Material*.

The JavaVM log file is output with the specified file name in the output destination specified with the `-XX:HitachiJavaLog:`*path-and-file-name* option. If the file specification is omitted, the Java VM log file is output under the name of `javalogxx.log` in the directory specified in the `ejb.server.log.directory` parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file. *xx* is a serial number of 2 digits starting from 01.

Reference note

When you perform the settings to output the JavaVM log file, create a file while starting the JavaVM. Therefore, until the JavaVM is terminated, the JavaVM log file without any information remains, if the JavaVM log is not output.

4.11 JavaVM Output Message Logs (Standard Output or Error Report File)

This section describes how to acquire the message log output by JavaVM.

When the JavaVM crashes, JavaVM outputs the debug information to the standard output and error report file.

The error report file is output in the following cases:

- When a signal occurs in the JNI
- When C heap is insufficient in the JavaVM
- When an unexpected signal occurs in the JavaVM
- When an Internal Error (internal logical error) occurs in the JavaVM

However, this file is not created if a signal or memory is insufficient when creating error report files.

4.11.1 In Windows

The output destination of the Error report file and output file name is as follows.

working-directory`\ejb\`*server-name*`\hs_err_pid`*server-process-ID*`.log`

Reference note

When C heap is insufficient, the message and dump are output in the following order. Acquire the necessary information.

1. A message log indicating insufficient C heap is output to the error report file and standard output.
2. If the memory is insufficient during the execution of 1., a simple message is output to the standard output.

4.11.2 In UNIX

The output destination of the Error report file and output file name is as follows.

working-directory`/ejb/`*server-name*`/hs_err_pid` *server-process-process-ID*`.log`

Reference note

When C heap is insufficient, a message is output and a core dump is generated in the following order. Acquire the necessary information.

1. A message log indicating insufficient C heap is output to the error report file and standard output.
2. If the memory is insufficient during the execution of 1., a simple message is output to the standard output.
3. If the memory is still insufficient when a simple message is output, output of the message and error log file is interrupted and a core dump is generated.

4.12 OS Status Information and OS Logs

This section describes the OS log information required as troubleshooting information.

4.12.1 Acquiring the OS Status Information

How to acquire the OS status information required as troubleshooting information, is described below for each OS:

(1) In Windows

You can acquire the OS status information by using the cjgetsysinfo command. If you specify the `-f` option, you can output the OS status information to the OS status output file.

Execute the command in the following format.

```
cjgetsysinfo -f OS-status-output-file-path
```

The information acquired with this command and the following commands of OS is same.

```
netstat -e
netstat -s
netstat -an
set
```

Note that the required OS status information to be acquired as troubleshooting information when the cjgetsysinfo command is not to be executed is described below. Create a directory for acquiring each information in advance and generate the file in that directory. Create the directory in any path.

Table 4–28: Required OS status information as a troubleshooting information

Information type	Default file name

Network information	Protocol statistical information and current TCP/IP network connection information. Acquire using the following commands in a sequence.
	`netstat -e > netstat_e.txt`
	`netstat -s > netstat_s.txt`
	`netstat -an > netstat_an.txt`

Environment variable	Currently set environment variables. Acquire using the following command. `set >set.txt`

(2) In UNIX

You can acquire the OS status information by using the cjgetsysinfo command. If you specify the `-f` option, you can output the OS status information to the OS status output file.

Execute the command in the following format.

```
cjgetsysinfo -f OS-status-output-file-path
```

The information acquired with this command and the commands of OS shown in the following table is the same.

Table 4–29: OS commands executed by executing the cjgetsysinfo command

In AIX	In Linux
• df -k • ps -elf • ps -A -m -o THREAD • vmstat -t 1 1 • vmstat -s • lsps -s • netstat -i • netstat -m • netstat -an • iostat • svmon -P • svmon -G • sar -A 1 • instfix -i • lslpp -hac • uname -a • env • set • ipcs -a	• df • ps -eflm • vmstat • netstat -s • netstat -an • iostat# • top -b -n 1 • sysctl -a • sar -A 1 1# • rpm -qa • rpm -qai • uname -a • env • set • ipcs • ipcs -t • ipcs -p • ipcs -c • ipcs -u • ipcs -l

#

To execute the sar command and the iostat command, you must install the sysstat package included in the Linux.

Note that the methods (commands) to acquire the required OS status information to be acquired as troubleshooting information when the cjgetsysinfo command is not to be executed, are as follows:

In AIX

```
df -k > df_k`date +"%y%m%d%H%M%S"`.txt
ps -efl > ps_efl`date +"%y%m%d%H%M%S"`.txt
ps -A -m -o THREAD > ps_AmoTHREAD`date +"%y%m%d%H%M%S"`.txt
vmstat -t 1 5 > vmstat`date +"%y%m%d%H%M%S"`.txt
vmstat -s > vmstat_s`date +"%y%m%d%H%M%S"`.txt
lsps -s > lsps_s`date +"%y%m%d%H%M%S"`.txt
netstat -i > netstat_i`date +"%y%m%d%H%M%S"`.txt
netstat -m > netstat_m`date +"%y%m%d%H%M%S"`.txt
netstat -an > netstat_an`date +"%y%m%d%H%M%S"`.txt
iostat 1 5 > iostat`date +"%y%m%d%H%M%S"`.txt
svmon -P > svmon_P`date +"%y%m%d%H%M%S"`.txt#1
svmon -G -i 1 5 > svmon_G`date +"%y%m%d%H%M%S"`.txt#1
sar -A 1 5 > sar_A`date +"%y%m%d%H%M%S"`.txt#1
/usr/samples/kernel/vmtune > vmtune`date +"%y%m%d%H%M%S"`.txt
instfix -i > instfix_i`date +"%y%m%d%H%M%S"`.txt
lslpp -hac > lslpp_hac`date +"%y%m%d%H%M%S"`.txt
uname -a > uname_a`date +"%y%m%d%H%M%S"`.txt
env > env`date +"%y%m%d%H%M%S"`.txt
set > set`date +"%y%m%d%H%M%S"`.txt
ipcs -a > ipcs_a`date +"%y%m%d%H%M%S"`.txt
```

In Linux

```
df > df`date +"%y%m%d%H%M%S"`.txt
ps -eflm > ps`date +"%y%m%d%H%M%S"`.txt
vmstat 1 5 > vmstat`date +"%y%m%d%H%M%S"`.txt
netstat -s > netstat_s`date +"%y%m%d%H%M%S"`.txt
netstat -an > netstat_an`date +"%y%m%d%H%M%S"`.txt
iostat 1 5 > iostat`date +"%y%m%d%H%M%S"`.txt#2
top n 5 > top`date +"%y%m%d%H%M%S"`.txt
sar -A 1 5 > sar`date +"%y%m%d%H%M%S"`.txt#2
sysctl -a > sysctl`date +"%y%m%d%H%M%S"`.txt
rpm -qa > rpm_qa`date +"%y%m%d%H%M%S"`.txt
rpm -qai > rpm_qai`date +"%y%m%d%H%M%S"`.txt
uname -a > uname_a`date +"%y%m%d%H%M%S"`.txt
env > env`date +"%y%m%d%H%M%S"`.txt
set > set`date +"%y%m%d%H%M%S"`.txt
ipcs > ipcs`date +"%y%m%d%H%M%S"`.txt
ipcs -t > ipcs_t`date +"%y%m%d%H%M%S"`.txt
ipcs -p > ipcs_p`date +"%y%m%d%H%M%S"`.txt
ipcs -c > ipcs_c`date +"%y%m%d%H%M%S"`.txt
ipcs -u > ipcs_u`date +"%y%m%d%H%M%S"`.txt
ipcs -l > ipcs_l`date +"%y%m%d%H%M%S"`.txt
```

#1

The root permissions are required for executing the command.

#2

To execute the `sar` command and the `iostat` command, you must install the `sysstat` package included in the Linux.

4.12.2 Acquiring OS Logs

How to acquire the OS logs required as troubleshooting information is described below for each OS:

(1) In Windows

The following table describes the OS logs required as troubleshooting information.

Table 4–30: OS log information required as troubleshooting information

Information type	Default file name
Event log	Open the event viewer and save the application and system log.

(2) In UNIX

The location to save the OS logs (syslog) required as a troubleshooting information is as follows:

In AIX

All under `/var/adm/ras`

In Linux

All under `/var/log`

4.13 OS Statistical Information

How to acquire the statistical information of OS is described below for each OS.

4.13.1 In Windows

Save the performance log after the occurrence of errors. For details on the `performance` operation, see the manuals provided with the OS.

Tip

The OS statistical information can be acquired only when collection of the performance log is started by the OS-dependent performance function in advance.

Extract the following system monitor logs when a J2EE server is running at an interval of 60 seconds. For details about the settings methods, see the manuals provided with the OS.

Table 4–31: System monitor settings

Performance object	Instance	Item name	Description
processor	--	%Processor Time	CPU utilization (total value excluding the threads in non-idle state).
		%Privileged Time	CPU utilization (in kernel mode).
		%User Time	CPU utilization (in user mode).
memory	--	Cache Bytes	Number of bytes used currently by the file system cache.
		Cache Faults/sec	Frequency of fetching from different memory location or from the disk per second.
		Page Faults/sec	Number of page faults per second.
		Transition Faults/sec	Number of faults per second.
process	_Total	Handle Count	Total number of handles currently opened.
		Page Faults/sec	Occurrence rate of page faults.
		Private Bytes	Used memory (bytes).
		Virtual Bytes	Used virtual memory (bytes).
		Working Set Bytes	Used actual memory (bytes).
	cjstartsv	%Processor Time	CPU utilization (total value excluding the threads in non-idle state).
		%Privileged Time	CPU utilization (in kernel mode).
		%User Time	CPU utilization (in user mode).
		Page Faults/sec	Occurrence rate of page faults.
		Thread Count	Thread count.
		Private Bytes	Used memory (bytes).

Performance object	Instance	Item name	Description
		Virtual Bytes	Used virtual memory (bytes).
		Working Set Bytes	Used actual memory (bytes).

Legend:
--: Not applicable

4.13.2 In UNIX

Execute the commands shown below while starting the application server and acquire the OS statistical information. Though Hitachi recommends you to acquire the OS statistical information at an interval of 60 seconds, you can decide the interval according to your disk capacity. Note that if you make the acquisition interval longer, the performance deterioration due to acquiring of statistical information of an OS can be reduced, but the accuracy of the OS statistical information might deteriorate.

In AIX

```
ps -efl
ps -A -m -o THREAD
vmstat -t
vmstat -s
lsps -s
svmon -P <Process ID of cjstartsv, cjstartweb>#1
svmon -G#1
sar -A 1#1
```

In Linux

```
ps -eflm
top n 1
vmstat
sar -A 1#2
```

#1
The root permissions are required for executing the command.

#2
To execute the `sar` command, you must install the `sysstat` package included in the Linux.

4.14 Application Server definition information

Acquire the Application Server definition. Use this information to confirm the set definitions when an error occurs.

Definition information related to Cosminexus Component Container

Acquire the set of files that have been saved under the following directory.

In Windows

- *Cosminexus-installation-directory*\CC\server\usrconf\ejb*server-name*

In UNIX

- /opt/Cosminexus/CC/server/usrconf/ejb/*server-name*

We recommend you to delete the information such as password that cannot be made public, from the files saved for troubleshooting.

4.15 Contents of J2EE server or batch server working directory

When a problem occurs in a system, maintenance personnel may examine the working directory for investigating the cause. A *working directory* is specified in the `ejb.public.directory` parameter in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file. Note that the default directory path of the working directory is *Cosminexus-installation-directory*`\CC\server\public` (in Windows) or `/opt/Cosminexus/CC/server/public` (in UNIX).

Confirm the settings, and when trouble occurs, save the files in this directory.

4.16 Application Server Resource Setting Information

Acquire the resource settings information to confirm the resource settings such as connection pool settings. The location to save the settings information is as follows:

- In Windows
  - *working-directory*\ejb*server-name*\import
  - *working-directory*\ejb*server-name*\rars
- In UNIX
  - *working-directory*/ejb/*server-name*/import
  - *working-directory*/ejb/*server-name*/rars

4.17 Web Server Logs

Acquire Web server (Cosminexus HTTP Server or Microsoft IIS) logs that are used in the system.

- **In the case of** Cosminexus HTTP Server
  The location to save the log is as follows:
  - In Windows
    Cosminexus-installation-directory`\httpsd\logs`
  - In UNIX
    `/opt/hitachi/httpsd/logs` (default)
- **Microsoft IIS**
  The location to save the log is as follows:
  `C:\inetpub\logs` (Specify the system drive in place of C:)

4.18 JavaVM stack trace information

In UNIX, when JavaVM terminates abnormally and a core dump is output, the information (stack trace information) required for investigating the cause for such abnormal termination can be acquired by the javatrace command. With the `javatrace` command, acquire the stack trace information from the core dump output. The `javatrace` command is installed in `/opt/Cosminexus/jdk/bin`.

The execution format of the javatrace command is as follows:

```
javatrace core dump file name, Execution file name where core dump is creat
ed
```

When JavaVM terminates abnormally and you execute this command when a core dump is created with a file name called core, as a result, a file called javatrace.log is output under the current directory. Send this file to maintenance personnel.

Execution example

The example below describes a message that is output when JavaVM terminates abnormally and a core dump is created.

```
 ...
# You can get further information from javatrace.log file generated
# by using javatrace command.
# usage: javatrace core-file-name loadmodule-name [out-file-name] [-l(libr
ary-name)...]
# Please use javatrace command as follows and submit a bug report
# to Hitachi with javatrace.log file:
# [/opt/Cosminexus/jdk/bin/javatrace core /opt/Cosminexus/CC/server/bin/cj
startsv]
#
```

Execute the character string of the javatrace command displayed in the message. In the case of this example, execute `/opt/Cosminexus/jdk/bin/javatrace core /opt/Cosminexus/CC/server/bin/cjstartsv`. As a result of execution, a file called `javatrace.log` is output under the current directory.

Note that the file name of the core dump actually output depends on the OS and it may be core.process ID in some cases. In such cases, specify the file name of the core dump actually output in the javatrace argument.

4.19 Event log of the Explicit Memory Management functionality

An event log output by an Explicit Memory Management functionality is output to the file specified in the `-XX:HitachiExplicitMemoryJavaLog` option in the JavaVM startup option. Moreover, the output contents differ depending on the log output level specified in the `-XX:HitachiExplicitMemoryLogLevel` option.

For details about setting the above-mentioned options, see *3.3.17(3) Settings for acquiring the event log of Explicit Memory Management functionality*.

For details about the JavaVM startup option, see *14. Options for Invoking JavaVM* in the *uCosminexus Application Server Definition Reference Guide*.

4.20 Information on the execution of the Component Container Administrator setup command (In UNIX)

When Component Container Administrator is specified, the information on the execution of the Component Container Administrator setup (`cjenvsetup` command) is output to a message log and a text file. If the previous execution information exists, a maximum of five files are stored including the latest execution information stored until the fourth generation for each type.

Also, in the case of an overwrite installation, if Component Container Administrator was set up before the overwrite installation, the Component Container Administrator setup (`cjenvsetup` command) is automatically executed during installation.

The following table describes the output destination of the execution information.

Table 4–32: Output destination of the information on the execution of the Component Container Administrator setup (cjenvsetup command)

Classification	Contents	Log output destination and log file name
Message log	Operation log of the `cjenvsetup` command	`/opt/Cosminexus/CC/logs/cjenvsetupmessage[`*n*`]#.log`
Text file	File (directory) information before change	`/opt/Cosminexus/CC/logs/before_cjenvsetup_files[`*n*`]#.txt`
Text file	File (directory) information after change	`/opt/Cosminexus/CC/logs/after_cjenvsetup_files[`*n*`]#.txt`

#:

In [*n*], specify the generation number (from 1 to 4). The generation number is not added to the latest execution information. A maximum of five log files are stored.

For details on the output contents, see *4.1.4 Notes on setting Component Container administrator (For UNIX)* in the *uCosminexus Application Server System Setup and Operation Guide*.

5 Problem Analysis

This chapter describes the log and trace information used for troubleshooting.

5.1 Organization of this chapter

This section describes the output contents of the data for troubleshooting, from among the description related to the troubleshooting.

Determine the cause for a trouble on the basis of contents output to the acquired data. Note that among the data acquired in *2.3 Acquiring the Data*, for the OS statistical information, see the manuals provided with the OS being used. Moreover, the memory dump is not explained here as it is checked by maintenance personnel.

Important note

- In Windows, if you open a large log file (3 megabytes or more) with a text editor, the computer is burdened and this might affect the system. When you want to reference a log file, take proper precautions.
- Note the following points if you are using Windows.
  When there is a Unicode supplementary character in the contents output to a log or a PRF trace that supplementary character cannot be output correctly. Other than this, there will not be any problem in other output contents or the application operations. The Unicode supplementary character is sometimes included in the requests from the clients, such as Internet Explorer.

The following table describes the organization of this chapter.

Table 5–1: Organization of this chapter (Data Output Contents used for troubleshooting)

Category	Title	Reference
Explanation	Application Server Log	*5.2*
	Contents of the EJB Client Application Log	*5.3*
	Contents of the Trace based performance analysis	*5.4*
	Contents of the JavaVM Thread Dump	*5.5*
	Contents of the JavaVM GC Log	*5.6*
	Contents of the JavaVM log (JavaVM log file)	*5.7*
	Contents of the Message Log Output by JavaVM (Standard Output and Error Report File)	*5.8*
	OS Status Information and Contents of the OS Log	*5.9*
	Contents of the JavaVM Stack Trace Information	*5.10*
	Contents of event log of Explicit Memory Management functionality	*5.11*

For an overview of troubleshooting, output destinations and output methods of data, and settings related to the data acquisition and output, see the respective sections:

- Overview of troubleshooting and methods for automatic output of data
  2. Troubleshooting
- Settings related to data acquisition and output
  3. Preparing for Troubleshooting
- Default output destination of data and methods to output data separately
  4. Output Destinations and Output Methods of Data Required for Troubleshooting

5.2 Application Server Log

This section describes how to investigate Application Server log.

By investigating message logs and user logs of Application Server, you can investigate the cause of error occurrence. Moreover, even in the case of process errors, you can confirm the progress status of the process and error indications.

Furthermore, among Application Server logs, for the precautions when referencing a system log of the EJB client applications, see *2.6.1 Precautions Related to the System Log of an EJB Client Application*.

The Application Server log includes the following three types of logs:

- Trace common library format (single process)
- Trace common library format (multi-process)
- Specific format

The trace common library format is the log that is output by using the trace common library. For details on the trace common library format, see *8.2.2 Mechanism of the user log output* in the *uCosminexus Application Server Expansion Guide*.

The specific format is the log that is output in a format other than the trace common library format.

This section classifies and describes the Application Server log for each log type. The following Application Server log is described here:

- Hitachi Trace Common Library format log
- Event log
- syslog
- Log output to audit log
- Log output by Cosminexus JMS Provider

Note that the time-based switching of output destinations and file naming in Shift mode might be enabled in the Hitachi Trace Common Library format log. For details, see *3.2.1 Specifiable contents*.

The following table describes the logs that are output, the corresponding log types, and whether time-specified switching and shift mode are supported, for each acquired log.

Table 5–2: J2EE server log type

Category	Contents	Type	Support for time-specified switching and Shift mode
Message log	Operation log	Hitachi Trace Common Library format (single process)	Y
	Log operation log	Proprietary format	--
	Operation log of resource adapter that is deployed and used as J2EE resource adapter	Hitachi Trace Common Library format (single process)	Y

Category	Contents	Type	Support for time-specified switching and Shift mode
	Operation log of resource adapter used by including in J2EE application	Hitachi Trace Common Library format (single process)	Y
	Web servlet log	Hitachi Trace Common Library format (single process)	Y
User log	User output log	Hitachi Trace Common Library format (single process)	Y
	User error log	Hitachi Trace Common Library format (single process)	Y
	JavaVM maintenance information and GC log	Proprietary format	--
	Event log of the Explicit Memory Management functionality	Proprietary format	--
Exception log	Exception information when an error occurs	Hitachi Trace Common Library format (single process)	Y
Maintenance log	Maintenance information	Hitachi Trace Common Library format (single process)	Y
	Console message	Hitachi Trace Common Library format (single process)	Y
	EJB container maintenance information	Hitachi Trace Common Library format (single process)	Y
	Web container maintenance information	Hitachi Trace Common Library format (single process)	Y
	Start process standard output information	Proprietary format	--
	Start process standard error information	Proprietary format	--
	Termination process information	Hitachi Trace Common Library format (single process)	--
Event log	Log showing J2EE server start, stop or abnormal termination	Hitachi Trace Common Library format (single process)	--
syslog	Log showing J2EE server start, stop or abnormal termination	Hitachi Trace Common Library format (single process)	--
Access log	Processing results of HTTP and WebSocket communication	Access log format	Y

Legend:

Y: Supported

--: Not supported or not applicable

Table 5–3: Server management command log

Category	Contents	Type	Support for time-specified switching and Shift mode
Message log	Operation log	Hitachi Trace Common Library format (multi processes)	--
	Log operation log	Proprietary format	--
Exception log	Exception information when an error occurs	Hitachi Trace Common Library format (multi processes)	--
Maintenance log	Maintenance information	Hitachi Trace Common Library format (multi processes)	--
	Console message	Hitachi Trace Common Library format (multi processes)	--
	Server management command maintenance information	Hitachi Trace Common Library format (multi processes)	--

Legend:

--: Not supported or not applicable

Table 5–4: Resource adapter version-up command (cjrarupdate) log

Category	Contents	Type	Support for time-specified switching and Shift mode
Message log	Operation log	Hitachi Trace Common Library format (multi processes)	--
Exception log	Exception information when an error occurs	Hitachi Trace Common Library format (multi processes)	--
Maintenance log	Maintenance information	Hitachi Trace Common Library format (multi processes)	--

Legend:

--: Not supported or not applicable

Table 5–5: Migration command (cjenvupdate) log

Category	Contents	Type	Support for time-specified switching and Shift mode
Message log	Operation log of the `cjenvupdate` command	Hitachi Trace Common Library format (multi processes)	--
Exception log	Exception information of the `cjenvupdate` command	Hitachi Trace Common Library format (multi processes)	--
Maintenance log	Maintenance information of the `cjenvupdate` command	Hitachi Trace Common Library format (multi processes)	--

Legend:

--: Not supported or not applicable

Table 5–6: Resource depletion monitoring log

Monitored Resources	Type	Support for time-specified switching and Shift mode
Memory	Hitachi Trace Common Library format (single process)	Y
File descriptors	Hitachi Trace Common Library format (single process)	Y
Threads	Hitachi Trace Common Library format (single process)	Y
Thread dump	Hitachi Trace Common Library format (single process)	Y
HTTP requests pending queue	Hitachi Trace Common Library format (single process)	Y
HTTP session numbers	Hitachi Trace Common Library format (single process)	Y
Connection pool	Hitachi Trace Common Library format (single process)	Y

Legend:

Y: Supported

Table 5–7: Administration agent, Management agent, and Management Server log

Category	Contents	Type	Support for time-specified switching and Shift mode
Integrated log	Integrated message log	Hitachi Trace Common Library format (multi processes)	Y
	Integrated trace log	Hitachi Trace Common Library format (multi processes)	Y
	Command maintenance log	Hitachi Trace Common Library format (multi processes)	--
Administration agent	Standard error output of Administration agent	Hitachi Trace Common Library format (single process)	--
	Standard output of Administration agent	Hitachi Trace Common Library format (single process)	--
	Standard command line error output of Administration agent	Proprietary format	--
	Administration agent log	Hitachi Trace Common Library format (single process)	--
	Administration agent start, stop command log	Hitachi Trace Common Library format (single process)	--
	Administration agent maintenance log	Hitachi Trace Common Library format (single process)	--
	Console log	Hitachi Trace Common Library format (single process)	Y
	Administration agent service log	Hitachi Trace Common Library format (single process)	--

Category	Contents	Type	Support for time-specified switching and Shift mode
	Administration agent service standard output	Proprietary format	--
	Administration agent service standard error output	Proprietary format	--
Management agent	Management agent log and trace J2EE server system JP1 event and J2EE server user JP1 event log Management event issue log	Hitachi Trace Common Library format (single process)	--
Management Server	Management Server service log	Hitachi Trace Common Library format (single process)	--
	Management Server service standard error output	Proprietary format	--
	Management Server service standard output	Proprietary format	--
	Management Server service start stop command	Hitachi Trace Common Library format (single process)	--
	Management Server log System JP1 event log of Management Server#	Hitachi Trace Common Library format (single process)	--
	Execution log of the `mngenvsetup` command	Hitachi Trace Common Library format (single process)	--
	Management Server maintenance log	Hitachi Trace Common Library format (single process)	--

Legend:

Y: Supported

--: Not supported or not applicable

#

Management Server of Application Server.

Table 5–8: Internal setup tool of the virtual server manager and Server Communication Agent logs

Category	Contents	Type	Support for time-specified switching and Shift mode
Internal setup tool of the virtual server manager	Logs for the internal setup tool of the virtual server manager	Trace common library format (single process)	Y
	Maintenance logs for the internal setup tool of the virtual server manager	Trace common library format (single process)	--
Server Communication Agent	Server Communication Agent logs	Trace common library format (single process)	Y
	Service logs of the Server Communication Agent	Trace common library format (single process)	--

Category	Contents	Type	Support for time-specified switching and Shift mode
	Start and stop command logs of the Server Communication Agent	Trace common library format (single process)	--
	Standard error output of the Server Communication Agent	Trace common library format (single process)	--
	Standard output of the Server Communication Agent	Trace common library format (single process)	--
	Console logs	Trace common library format (single process)	Y
	Maintenance logs of the Server Communication Agent	Trace common library format (single process)	--
	Maintenance logs for the Server Communication Agent services	Trace common library format (single process)	--
	Maintenance logs for the start and stop commands of the Server Communication Agent	Trace common library format (single process)	--
	JavaVM log file of the Server Communication Agent	Trace common library format (single process)	--

Legend:

Y: Supported

--: Not supported or not applicable

Table 5–9: Cosminexus Performance Tracer log

Contents	Type
PRF daemon and PRF command log	Proprietary format
Module trace	Proprietary format
Structured exception occurrence log	Proprietary format
Maintenance information	Proprietary format

Table 5–10: Cosminexus Component Transaction Monitor log

Contents	Type
CTM daemon or CTM command log	Proprietary format
Maintenance information	Proprietary format

Table 5–11: Log output to audit log

Category	Contents	Type	Support for time-specified switching and Shift mode
Message log	Message log of audit log	Hitachi Trace Common Library format (multi processes)	--

Category	Contents	Type	Support for time-specified switching and Shift mode
Exception log	Exception information of audit log	Hitachi Trace Common Library format (multi processes)	--

Legend:

--: Not supported or not applicable

Table 5–12: Log output by Cosminexus JMS Provider

Category	Contents	Type	Support for time-specified switching and Shift mode
Message log	CJMSP Broker message log	Hitachi Trace Common Library format (single process)	--
	Management command (`cjmsicmd` command) message log	Hitachi Trace Common Library format (multi processes)	--
	CJMSP resource adapter message log	Hitachi Trace Common Library format (single process)	Y
Exception log	CJMSP Broker exception log	Hitachi Trace Common Library format (single process)	--
	Management command (`cjmsicmd` command) exception log	Hitachi Trace Common Library format (multi processes)	--
	CJMSP resource adapter exception log	Hitachi Trace Common Library format (single process)	Y

Legend:

Y: Supported

--: Not supported or not applicable

5.2.1 Output Format and Output Items of the Hitachi Trace Common Library Format Log

The output format and output items of the Hitachi Trace Common Library format log are described below.

Application Server log is output in the Hitachi Trace Common Library format.

(1) Output format

The output format of Hitachi Trace Common Library format log is as follows:

```
Number Date Time AP name pid tid message ID Type Message text CRLF
```

(2) Output items

The output items of the Hitachi Trace Common Library format log are as follows:

Table 5–13: Output items of the Hitachi Trace Common Library format log

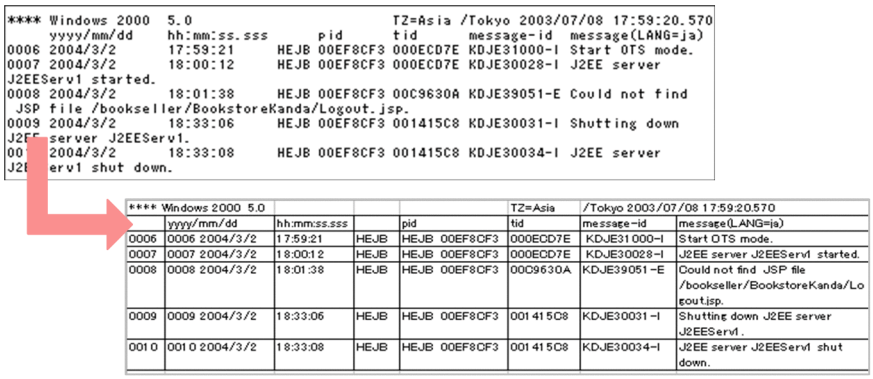
Item name	Description
No.	A 4 digit number showing the trace record serial number is output.
Date	Date of acquiring the trace is output in the yyyy/mm/dd format.
Time	Time of acquiring the trace is output in the hh:mm:ss.sss format.
AP name	String showing the program is output.
Pid	Process ID is output.
Tid	Thread ID is output.
Message ID	Message ID is output in the XXXXnnnnn-Y format.
Type	Event type that triggers the trace output is output.
Message text	Message text is output (up to 4,095 bytes). Any part that exceeds 4,095 bytes will be truncated. Additional information might also be output.
CRLF	Terminal code of the record (0x0D, 0x0A) is output.

Editing and displaying log files

When the sort or filter function is used by displaying the Hitachi Trace Common Library format log in Microsoft Excel, you can effectively investigate the cause of the error that occurred.

The example of displaying the Hitachi Trace Common Library format log by using Microsoft Excel is as follows:

Figure 5–1: Display example of log of Hitachi Trace Common Library format using Microsoft Excel



**** Windows 2000 5.0					TZ=Asia	/Tokyo 2003/07/08 17:59:20.570	
	yyyy/mm/dd	hh:mm:ss.sss		pid	tid	message-id	message(LANG=ja)
0006	0006 2004/3/2	17:59:21	HEJB	HEJB 00EF8CF3	000ECD7E	KDJE31000-I	Start OTS mode.
0007	0007 2004/3/2	18:00:12	HEJB	HEJB 00EF8CF3	000ECD7E	KDJE30028-I	J2EE server J2EEServ1 started.
0008	0008 2004/3/2	18:01:38	HEJB	HEJB 00EF8CF3	00C9630A	KDJE39051-E	Could not find JSP file /bookseller/BookstoreKanda/Logout.jsp.
0009	0009 2004/3/2	18:33:06	HEJB	HEJB 00EF8CF3	001415C8	KDJE30031-I	Shutting down J2EE server J2EEServ1.
0010	0010 2004/3/2	18:33:08	HEJB	HEJB 00EF8CF3	001415C8	KDJE30034-I	J2EE server J2EEServ1 shut down.

Reference note

- As a delimiting data format, select fixed length field data that is right or left justified by using spaces.
- Delete the unnecessary arrows in the Data Preview box.
- Specify `Character string` in `Display format` of each column.

5.2.2 Precautions to Be Taken When Referencing the Hitachi Trace Common Library Format Log

This section describes the precautions to be taken when using the Hitachi Trace Common Library format log.

(1) Common precautions for multi and single processes

The common precautions for multi and single processes are as follows:

- Do not edit the log file that has been output.
- Do not lock files by using a function such as the text editor.
- When setting the access rights for the output log file manually, assign the appropriate access rights.
- Do not change the update time of the log file that has been output.
- During trace output, do not delete log files or change file names. Delete the log files or change file names after stopping all the trace output processes.
- The management file *xxxxxx*`.conf` is output with the log file. Do not edit or delete this file while the trace output process is running.
- Stop the trace output process before changing the log file size, number of log files, or other settings. When doing so, back up or delete all log files and management files (*xxxxxx*`.conf`) under the log output directory (including subdirectories).
- If you change the owner or owner group of the log files, also change the owner or owner group of the management file (*xxxxxx*`.conf`) to the same owner or owner group.

(2) Precautions when referencing multi processes Hitachi Trace Common Library

The precautions to be taken when referencing multi processes compliant Hitachi Trace Common Library format log file are as follows:

- The line feed code at the end of the message is CRLF, regardless of the OS being used.
- In some cases you may need to delete the log files to enable changes in the log file size, number of files and mode options. Before deleting the log files, stop all the processes that output trace.
- Even if trace is output, there are cases when the update time of the log file is not updated. Therefore, you cannot judge whether the trace is output based on the updated time of the file.

(3) Changes in the multi-process trace common library from the earlier versions

The log files of the multi-process trace common library have been changed from the earlier versions. The following table describes the changes.

Table 5–14: Differences with the earlier versions of the multi-process trace common library

Item	Application Server V9	Earlier than Application Server V9
File size	Variable (specified size might be exceeded)	Fixed (specified size)
File switching time	The files are switched at the following times: • When the file size exceeds the specified size • For time specified rotation • When the specified time is reached while the process is runing	When the file size + the size you want to output exceeds the specified size
Size when the files are switched	0 (All lines will be deleted)	Fixed (overwrite and store)

Item	Application Server V9	Earlier than Application Server V9
File switching operation (Wraparound method)	The trace data is overwritten and output from the beginning of the file (the information before wraparound remains as is)	The trace data is overwritten and output from the beginning of the file (the information before wraparound remains as it is)
File switching operation (Shift method)	All the lines are deleted and the trace data is output from the beginning of the file	--
Log file name numbering rules (Wraparound method)	xxxxxx1.log, xxxxxx2.log, xxxxxx3.log, ...	xxxxxx1.log, xxxxxx2.log, xxxxxx3.log, ...
Log file name numbering rules (Shift method)	xxxxxx.log, xxxxxx1.log, xxxxxx2.log, ...	--
End identifier	No (end of file)	Yes#
Management file	Created	Created
Determining the output file used when the process restarts	Time stamp	Management file, or time stamp

Legend:

--: Not applicable.

#

The end identifier used in the multi-process trace common library of the earlier versions is as follows:

```
EOF CRLF CRLF CRLF CRLF----------< End of Data >----------CRLF CRLF
```

EOF is the character (0x1A) that indicates the end of the trace data. CRLF indicates the linefeed (0x0D, 0x0A).

Table 5–15: List of change files for the multi-process trace common library format

Classification of log	Log type	Log file and default file output destination
Common log	Audit log	*Product-installation-directory*/auditlog/audit?.log
		Product-installation-directory/auditlog/rasexception?.log
		Product-installation-directory/auditlog/rasmessage?.log
Functionality-specific public log	Performance, error analysis trace	*Product-work-directory*/ejb/*J2EE-server-name*/logs/RM/maintenance/mtd_*RM-display-name*_??.log
		Product-work-directory/ejb/*J2EE-server-name*/logs/RM/maintenance/shq_*RM-display-name*_??.log
		Product-work-directory/ejb/*J2EE-server-name*/logs/RM/maintenance/lin_*RM-display-name*_??.log
		Product-installation-directory/CC/server/public/ejb/*J2EE-server-name*csmxsec_trace?.log
	Message log	*Product-work-directory*/ejb/*J2EE-server-name*/logs/CJW/cjwmessage?.log

Classification of log	Log type	Log file and default file output destination
		Product-work-directory/ejb/*J2EE-server-name*/logs/CJR/cjrmessage?.log
		Product-work-directory/ejb/*J2EE-server-name*/logs/WS/ <log_file_prefix>-j2ee-*J2EE-server-name*-<log_file_num>.log
		Product-work-directory/ejb/*J2EE-server-name*/logs/WS/ c4webcl-default-?.log
Manager log	Integrated message log	*Product-installation-directory*/manager/log/message/mngmessage?.log
	Integrated trace log	*Product-installation-directory*/manager/log/trace/mngtrace?.log
	Console log	*Product-installation-directory*/manager/log/processConsole?.log
	Internal setup tool log of the virtual functionality	*Product-installation-directory*/manager/setup/log/rasetup?.log
	Server communication agent log	*Product-installation-directory*/sinagent/log/sinaviagent?.log
	Server communication agent console log	*Product-installation-directory*/sinagent/log /processConsole?.log

5.2.3 Output format and output items of access log of NIO HTTP Server

This subsection describes the output format and the output items of access log of the NIO HTTP Server.

The processing results of requests of the NIO HTTP server are output to the access log. The following tables describe the output format and output items for the HTTP communication and the WebSocket communication handled by the NIO HTTP server.

(1) Access log for HTTP communication

The following table describes the output contents.

Table 5–16: Output contents for HTTP communication

Format argument	Output contents	Output example
%a	IP address of the Web client	10.20.30.40
%A	IP address of the J2EE server	10.20.30.100
%b	Number of bytes sent excluding the HTTP header. A hyphen (-) is output when the number of bytes is 0.	2048
%B	Number of bytes sent excluding the HTTP header. 0 is output when the number of bytes is 0.	1024
%h	Host name or IP address of the Web client. The IP address is output when the host name cannot be obtained.	10.20.30.40

Format argument	Output contents	Output example
%H	Request protocol	HTTP/1.1
%l	Remote log name. A hyphen (-) is always output[#1].	--
%m	Request method	GET
%p	Port number that received the request from the Web client	80
%q	Query string. The query string begins with a question mark (?). If the query string does not exist, an empty string is output.	?id=100&page=15
%r	Request line	GET /index.html HTTP/1.0
%s	Final status code	200
%S[#2]	The value of the cookie name JSESSIONID is output. If the value does not exist, a hyphen (-) is output.	00455AFE4DA4E7B7789F247B8FE5D605
%t	The time when the processing of the request from the Web client was started is displayed with second precision. [*dd*/*MMM*/*YYYY*:*HH*:*mm*:*ss Z*]	[18/Jan/2005:13:06:10 +0900]
%T	Time required for processing the request from the Web client (unit: seconds)	2
%d	The time when the processing of the request from the Web client was started is displayed with millisecond precision. [*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 request from the Web client (unit: milliseconds)	38
%u	Basic authentication user name or form authentication user name. If there is no authentication user name, a hyphen (-) is output.	user
%U	Request file path	/index.html
%v	Local host name of the J2EE server	server
%{foo}i[#3]	Contents of the request header foo. If the foo header does not exist, a hyphen (-) is output.	In the case of %{Host}i: www.example.com:8888
%{foo}c	Of the cookie information sent by the Web client, the contents of the cookie whose name is foo. If there is no cookie whose name is foo, a hyphen (-) is output.	In the case of %{MYSESSIONID}c: 00455AFE4DA4E7B7789F247B8FE5D605
%{foo}o[#3]	Contents of the response header foo.	In the case of %{Server}o:

Format argument	Output contents	Output example
	If the `foo` header does not exist, a hyphen (`-`) is output.	`CosminexusComponentContainer`
`%rootap`	Root application information	`10.100.10.100/1234/0x0000000000000001`
`%clport`	Port number that sent the request from the Web client	`888`

Notes

- If you specify a character string that begins with `%` (such as `%G`) other than those listed in the preceding table, the message KDJE39401-W is output and the default format is used. Also, if you specify 0 characters (for example, `%{}i`) in the header content or cookie name specified in `%{foo}i`, `%{foo}c`, or `%{foo}o`, the message KDJE39401-W is output and the default format is used.
- The maximum string length that can be used for the format type is 1,024 characters. If the string exceeds 1,024 characters, the message KDJE39400-W is output and the default value is used.
- The characters that can be used for the format type are those from ASCII code 32 (decimal number) to less than 127 (decimal number).
- If no string is specified, the message KDJE39009-W is output and the default value is used.
- If a character outside the range is specified, the message KDJE39401-W is output and the default value is used.

#1

The remote log name is the user name on the Web client side that is obtained by the Identification Protocol stipulated by RFC 1413.

#2

The value displayed by `%S` is the value of the cookie name JSESSIONID, which is used as the standard HTTP session ID. If the name of the cookie name JSESSIONID has been changed in Servlet 3.0 or later, use `%{foo}c`.

#3

The same header name might be sent multiple times in one HTTP request or HTTP response. In this case, the contents of all headers are output separated by commas (`,`).

The following shows the notation described with the format arguments.

```
%h %{X-Forwarded-For}i %l %u %d %rootap "%r" %s %b %D %S
```

The following shows the output format.

```
host-name-or-IP-address-of-the-Web-clientΔX-Forwarded-For-headerΔremote-log-
nameΔauthentication-user-nameΔtime-when-the-processing-of-the-request-from-t
he-Web-client-was-startedΔroot-application-informationΔ"request-line"Δfinal-
status-codeΔnumber-of-bytes-sent-excluding-the-HTTP-headerΔtime-required-for
-processing-the-request-from-the-Web-clientΔHTTP-session-ID
```

Note

A new line is inserted after the HTTP session ID.

Legend

Δ: Single byte space

The following shows an example of output.

```
10.20.30.40 50.60.70.80 - user [18/Jan/2005:13:06:10.152 +0900] 10.100.10.10
0/1234/0x0000000000000001 "GET /index.html HTTP/1.0" 200 1024 38 00455AFE4DA
4E7B7789F247B8FE5D605
```

(2) Access log for WebSocket communication

The following table describes the output contents.

Table 5–17: Output contents for WebSocket communication

Format argument	Output contents	Output example
%TS	Time of sending or receiving the WebSocket frame	2001/01/01 01:01:01.111 +0900
%IO	Direction of sending or receiving the WebSocket frame. IN or OUT is output. IN: Indicates that the server instance received a WebSocket frame. OUT: Indicates that the server instance sent a WebSocket frame.	IN
%OPCODE	Type of the WebSocket frame. Text, Binary, Ping, Pong, or Close is output.	Text
%URI	Request URL	/websocket_server/test001
%FIN	Identifier that indicates the end of the WebSocket frame. CONT or FINAL is output. CONT: Continuation of the WebSocket frame. FINAL: End of the WebSocket frame.	CONT
%PAYLOADDATALEN	Payload data length	100
%ROOTAP	Root application information	10.100.10.100/1234/0x0000000000000001
%CLIENTAP	Client application information	10.100.10.100/1234/0x0000000000000001
%CLOSEREASON	Reason why the WebSocket connection was disconnected	NORMAL_CLOSURE:closereason specified by WebSocketClient001
%CLIENTADDR	IP address and port number of the Web client	10.20.30.40:55555
%SERVERADDR	IP address and port number of the J2EE server	10.20.30.100:44444
%SESSIONID	WebSocket session ID	11111111-2222-3333-4444-555555555555

Format argument	Output contents	Output example
%MASK	Indicates the mask set for the WebSocket frame information. MASK: The information is masked. NOMASK: The information is not masked.	MASK
%MASKKEY	Key used to mask the information in the WebSocket frame. If the information is not masked, a hyphen (-) is output.	EEEEEEEE
%ISEXTENDED	Indicates the frame for which the endpoint is set to send or receive messages. BASE: A basic frame is used. EXTENDED: An extended frame is used.	BASE
%RSV	Indicates reserved bits set by the client endpoint during the negotiation of an extension.	RSV-000
%FRAMEMAINTYPE	Specify whether the WebSocket frame is a data frame or control frame. Data: A data frame is used. Control: A control frame is used.	Data
%PAYLOADDATA	Payload data	aaaaaa
%PAYLOADDATA(n)	Payload data when displaying only a certain number of characters indicating the beginning and end of the message. For binary, this is always a hyphen (-).	aaaaaaaaaa....aaaaaaaaaa

Notes

- The maximum string length that can be used for the format type is 1,024 characters. If the string exceeds 1,024 characters, the message KDJE39400-W is output and the default value is used.
- If no string is specified, the message KDJE39009-W is output and the default value is used.
- You can specify only the format arguments defined in the table connected by single byte spaces. When doing so, note the following points:

  - The number of single byte spaces is not preserved.

  - Leading and trailing spaces are not preserved.

  - There are no restrictions on the order of the format arguments.

  - There is a restriction on the number of occurrences (the use of the same format argument multiple times). If the same format argument is defined multiple times, only the first definition will be valid, and the second and subsequent definitions will not be valid.

The following shows the notation described with the format arguments.

```
%TS %IO %OPCODE %ROOTAP %URI %FIN %PAYLOADDATALEN %CLIENTAP %CLOSEREASON
```

The following shows the output format.

```
time-of-sending-or-receiving-the-WebSocket-frameΔdirection-of-sending-or-rec
eiving-the-WebSocket-frameΔtype-of-the-WebSocket-frameΔroot-application-info
rmationΔrequest-URLΔidentifier-that-indicates-the-end-of-the-WebSocket-frame
Δpayload-data-lengthΔclient-application-informationΔreason-why-the-WebSocket
-connection-was-disconnected
```

Note

A new line is inserted after the payload data length.

Legend

Δ: Single byte space

The following shows an example of output.

```
2001/01/01 01:01:01.111 +0900 IN Text10.100.10.100/1234/0x0000000000000001
/websocket_server/test001 CONT 100 10.100.10.100/1234/0x0000000000000003 -
```

5.2.4 Output Format and Output Items of the Event Log (In Windows)

The output format and output items of the event log are described below.

An Event log is output when starting, stopping, and abnormally terminating the J2EE server. The output format is as follows:

```
ID character string pid: Message text
```

The output items are as follows:

Table 5–18: Output items of event log

Item name	Description
ID character string	HEJB is output as the character string showing the application.
Pid	Process ID is output.
Message text	Message text is output.

5.2.5 Output Format and Output Items of syslog (In UNIX)

The output format and output items of syslog are described below.

syslog is output when starting, stopping, and abnormally terminating the J2EE server. The output format is as follows:

```
Date Time Host name ID character string pid: Message text
```

The output items are as follows:

Table 5–19: Output items of syslog

Item name	Description
Date	Date when the message is output.
Time	Time when the message is output.
Host name	Character string that shows host name is output.
Pid	Process ID is output.
ID character string	HEJB is output as the character string showing the application.
Message text	Message text is output.

5.3 EJB Client Application Log

The EJB client application log types are described below. For details on log types, log output format and output items for each log, and precautions while referencing logs, see the section *5.2 Application Server Log*.

The log types of an EJB client application are as follows:

Table 5–20: Types of log of EJB client application

Types	Contents	Type
Message log	Operation log	Hitachi Trace Common Library format (multi processes)
	`cjclstartap` command operation log	Hitachi Trace Common Library format (multi processes)
	`cjcldellog` command operation log	Hitachi Trace Common Library format (multi processes)
User log	User output log	Hitachi Trace Common Library format (multi processes)
	User error log	Hitachi Trace Common Library format (multi processes)
Java log	JavaVM maintenance information, GC log	Proprietary format
Exception log	Exception information when an error occurs	Hitachi Trace Common Library format (multi processes)

Note that the EJB client application logs only support the wraparound mode of switching the output destinations for file sizes.

5.4 Trace based performance analysis

By investigating the trace based performance analysis, you can analyze the processes that are moving further or the processes that are in a bottleneck for each request. Moreover, when you investigate a session trace, the life cycle of that session can be confirmed.

With these traces as a base, you can locate the places where problem has occurred and take appropriate measures for the places with a bottleneck. Check as per the requirement.

For details on the output contents of the trace based performance analysis, see *7. Performance Analysis by Using Trace Based Performance Analysis*.

5.5 JavaVM Thread Dump

When you investigate the JavaVM thread dump, it is easy to investigate the cause of the error at the Java program level such as system dead locks.

The type of output information differs depending on the options specified when starting the J2EE server. For details on settings for acquiring the JavaVM material, see the subsection *3.3.17 Settings for Acquiring the JavaVM Material*.

5.5.1 Structure of thread dump information

The structure of JavaVM thread dump information is as follows:

Table 5–21: Structure of the thread dump information

Output information	Contents
Header	Date, JavaVM version information, start command line is output.
System settings	Following information is output: • Java home path indicating the installation location of the JDK execution environment • Java DLL path indicating the installation directory of the library configuring JDK • System class path • Java command options
Operation environment	Following information is output: • Host name • OS version • CPU information • Resource information (in UNIX)
Memory information (in Windows)	Current memory usage size and various unused size information is output.
Java heap information	Memory usage status of each generation of Java heap is output.
JavaVM internal memory map information	Memory area information securing the JavaVM its self is output.
JavaVM internal memory size information	Memory size information securing the JavaVM its self is output.
Application environment	Following information is output: • Signal handler • Environment variable
Library information	Loaded library information is output.
Thread information <Thread 1> ... <Thread n>	Thread information for each thread is output.
Java monitor dump[#]	List of Java monitor objects is displayed.
JNI global reference information[#2]	Output the number of global references to JNI maintained by JavaVM.
Explicit heap detail information	Output the following information for each class of Java process when you use the Explicit Memory Management functionality: • Use status of entire Explicit heap • Use status for each Explicit memory block

Output information	Contents
	When you execute the `eheapprof` command while using the Explicit Memory Management functionality, the statistical information of object within the Explicit memory block and the release rate information for the Explicit memory block is output.
Class-wise statistical information	Output the following information for each class of Java process specified by the `jheapprof` command. • Total size and reference of instance as the member of an instance • Total size of instances possessed by static member • Total size of object classes and instances causing an increase in the Tenured area
Footer	The time when thread dump terminated is displayed.

#

In UNIX, the `notify` pending list might not be displayed.

For details on the JavaVM thread dump information, see *-XX:[+|-]HitachiThreadDump (Option to output the extended thread dump information)* in the *uCosminexus Application Server Definition Reference Guide*. For details on the class-wise statistical information, see *9.3 Class-wise statistical functionality*. For details on the Explicit heap details information, see *5.5.3 Output contents of Explicit heap details information*.

5.5.2 Mapping between thread dump and trace based performance analysis file

Moreover, when slowdown or hang-up occurs in a J2EE application, you can investigate the locations where the errors occurred by correlating the thread dump with trace based performance analysis.

To correlate the thread dump with trace based performance analysis, use the thread ID output to the trace based performance analysis file with the nativeID (OS level thread ID) of the thread information output to the thread dump. To specify the corresponding trace based performance analysis file from the thread dump:

1. Collect the thread dump and trace based performance analysis file.

   For details on the collection methods of thread dump, see *4.7 JavaVM thread dump*. For details on the collection methods of the trace based performance analysis file, see *7.3.1 How to collect a trace based performance analysis file*.

2. Select the thread dump and trace based performance analysis file to be used.

   Based on the time when the thread dump and trace based performance analysis file are output, select the thread dump and trace based performance analysis file to be used for investigation. For details on the output time, see the following information:

   Thread dump

   Date and time output at the end of the file name and file.

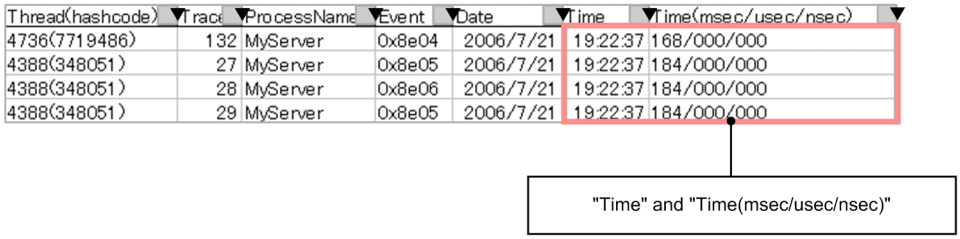
   The following is an example showing the date and time output at the end of a file:

```
  ...
  ...
Full thread dump completed. Fri Jul 21 19:22:47 2006
```

   Trace based performance analysis file

   Time and Time(msec/usec/nsec)

   The following figure shows the Time and Time (msec/usec/nsec) of the trace based performance analysis file.



3. Convert nid (hexadecimal) and jid (hexadecimal) of thread dump to decimal.

   - In Windows and AIX

     Convert the thread dump nid (hexadecimal) to decimal value.

```
 ...
 ...
"VBJ ThreadPool Worker" daemon prio=5 jid=0x00054f93 tid=0x04cef380 nid
=0x1124 in Object.wait() [0x0632f000..0x0632fd18]
 stack=[0x06330000..0x062f5000..0x062f1000..0x062f0000]
 [user cpu time=0ms, kernel cpu time=15ms] [blocked count=1, waited cou
nt=29]
at java.lang.Object.wait(Native Method)
 ...
 ...
```

     1124(Hexadecimal)=4388(Decimal)

   - In Linux

     Convert the thread dump nid (hexadecimal) to decimal value.

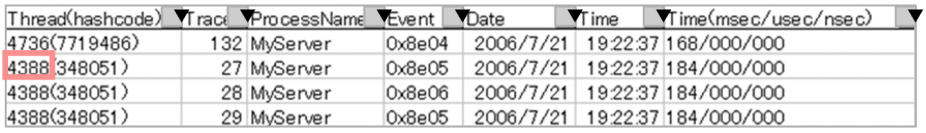
```
 ...
 ...
"main" prio=1 jid=0x00006d75 tid=0x00201d70 nid=0x1e51 waiting on condi
tion [0x00000000..0xbfe80488] stack=[0xbfe87000..0xbfc8c000..0xbfc88000
..0xbfc87000] [user cpu time=1320ms, kernel cpu time=4280ms] [blocked c
ount=5, waited count=4]
 ...
 ...
```

     6d75(Hexadecimal) = 28021 (Decimal)

4. Search the row in which the value (Decimal) of Thread(hashcode) in the trace based performance analysis file matches with the value converted to decimal by procedure 3., and specify the trace information.

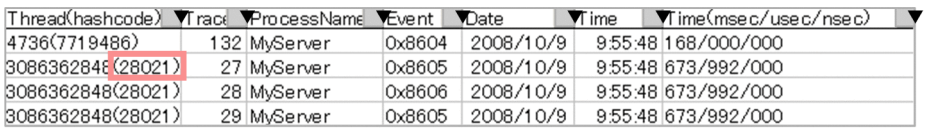
   - In Windows and AIX

     Search the line where value of Thread (Thread ID) is matching with the value converted to decimals.



   - In Linux

     Search the line where value of hashcode (Hashcode) is matching with the value converted to decimals.



5.5.3 Output contents of Explicit heap details information

The Explicit heap information and the Explicit memory block information is output to the Explicit heap details information. When there is more than one Explicit memory block, they are output to the Explicit memory block information. You can also output the object statistical information within the Explicit memory blocks and the release rate information for the Explicit memory block to the Explicit memory block information.

The output format, output items, and output examples of Explicit heap details information are as follows:

Output format

The output format differs depending on the execution of `eheapprof` command.

- When the `eheapprof` command is executed

```
Explicit Heap Status
--------------------
 max <EH_MAX>, total <EH_TOTAL>, used <EH_USED>, garbage <EH_GARB> (<EH
_PER1> used/max, <EH_PER2> \
used/total, <EH_PER3> garbage/used), <EM_NUMS> spaces exist

 Explicit Memories(<EM_MGR_PTR>)
...
 "<EM_NAME>" eid=<EID>(<EM_PTR>)/<EM_TYPE>, total <EM_TOTAL>, used <EM_
USED>, garbage <EM_GARB> \
(<EM_PER1> used/total, <EM_PER2> garbage/used, <FL_BLOCKS> blocks) <EM_
STAT>
 deployed objects
 ________________Size__Instances__FreeRatio__Class_________________
   <ISIZE> <INUM> <FRATIO> <CNAME>
    ...
  <AISIZE> <AINUM> total
...
```

Note: For details on the signs used in the output format, see *5.11.2(3) Signs used in description of output format of event log*.

- When the `eheapprof` command is not executed

```
Explicit Heap Status
--------------------
 max <EH_MAX>, total <EH_TOTAL>, used <EH_USED>, garbage <EH_GARB> (<EH
_PER1> used/max, <EH_PER2> \
used/total, <EH_PER3> garbage/used), <EM_NUMS> spaces exist

 Explicit Memories(<EM_MGR_PTR>)
...
 "<EM_NAME>" eid=<EID>(<EM_PTR>)/<EM_TYPE>, total <EM_TOTAL>, used <EM_
USED>, garbage <EM_GARB> \
(<EM_PER1> used/total, <EM_PER2> garbage/used, <FL_BLOCKS> blocks) <EM_
STAT>
...
```

Note: For details on the signs used in the output format, see *5.11.2(3) Signs used in description of output format of event log*.

Output items

The following table describes about each item in output format.

Table 5–22: Output items (Explicit heap details information)

Category	Output items	Output contents	Meaning
Explicit heap information	<EH_MAX>	<const>K	Maximum size of Explicit heap is output. The unit is kilo bytes.
	<EH_TOTAL>	<const>K	Secured Explicit heap size is output. The unit is kilo bytes.
	<EH_USED>	<const>K	Used Explicit heap size is output. The unit is kilo bytes.
	<EH_GARB>	<const>K	The internal status of Explicit heap is output.
	<EH_PER1>	<decimal>%	Explicit heap utilization rate (<EH_USED>/<EH_MAX>) is output in % sign.
	<EH_PER2>	<decimal>%	Explicit heap utilization rate (<EH_USED>/<EH_TOTAL>) is output in % sign.
	<EH_PER3>	<decimal>%	The internal status of Explicit heap is output.
	<EM_NUMS>	<const>	The valid Explicit memory blocks are output.
	<EM_MGR_PTR>	<ptr>	Memory address that has internal information for Explicit heap control is output. The memory address can be used for investigating errors.
Explicit memory block information	<EM_NAME>	<letters>	Name of Explicit memory block is output. The output contents are uncertain when multi byte characters are included in name of Explicit memory block (Usually garbled and output). At times "NULL" is output when name of Explicit memory block is output at about the same time of initialization of Explicit memory block or when JavaVM generates the Explicit memory block internally.
	<EID>	<const>	ID of Explicit memory block is output.
	<EM_PTR>	<ptr>	Memory address with Explicit memory block internal structure is output. The memory address can be used for investigating errors.
	<EM_TYPE>	R|B|A	Explicit type is output. R indicates Explicit memory block used internally in Application Server. B indicates Explicit memory block used by application. A indicates the Explicit memory block that is specified using the automatic allocation configuration file.
	<EM_TOTAL>	<const>K	Secured memory size of Explicit memory block is output. The unit is kilo bytes.
	<EM_USED>	<const>K	Used size of Explicit memory block is output. The unit is kilo bytes.
	<EM_GARB>	<const>K	Internal status of Explicit memory block is output. The unit is kilo bytes.
	<EM_PER1>	<decimal>%	Explicit memory block utilization ratio (<EM_USED>/<EM_TOTAL>) is output in % sign.
	<EM_PER2>	<decimal>%	Internal status of Explicit memory block is output.
	<FL_BLOCKS>	<const>	Normally 0 is output
	<EM_STAT>	Enable|Disable	Sub-status of Explicit memory block is output.
Object statistical information[#1]	<ISIZE>	<const>	Size in Explicit memory block of the object that sets the instance of a certain class is output.

Category	Output items	Output contents	Meaning
	<INUM>	<const>	Fixed quantity within the Explicit memory block of the object that sets the instance of certain class is displayed.
	<CNAME>	<letters>	Entire class name of class indicating <ISIZE> and <INUM> is output.
	<AISIZE>	<const>	Total size of all objects in Explicit memory block is output.
	<AINUM>	<const>	Number of all blocks in Explicit memory block is output.
Object release rate information#2	<FRATIO>	<decimal>%	The ratio of the objects released (object release rate) in the automatic release processing of the Explicit memory block is output with the % (percentage) sign. Object release rate = *(Number-of-class-objects-before-the-automatic-release-processing - Number-of-class-objects-after-the-automatic-release-processing)/Number-of-class-objects-before-the-automatic-release-processing* ×100 Note that when the object release rate information is output, "--" is output for the Explicit memory block that was not subject to the automatic release processing.

Note:

For details on the signs used in the output contents, see *5.11.2(3) Signs used in description of output format of event log*.

#1

The object statistical information is output while executing the eheapprof command. You can output actually created size, "[I" indicating the int type array greater than the quantity, in the object statistical information. In such cases, "[I" indicates objects that are not used in an Explicit memory block. Create the int type array in internal process of JavaVM for objects that are not used in the Explicit memory block.

#2

The object release rate information is output when you execute the eheapprof command specifying the -freeratio option.

Example of output

The output format differs depending on execution of the eheapprof command.

- When eheapprof command is executed

```
Explicit Heap Status
--------------------
 max 31415926K, total 162816K, used 150528K, garbage 10004K (0.0% used/
max, 91.1% used/total, 6.6% garbage/used), 3 spaces exist

 Explicit Memories(0x12345678)

 "EJBMgrData" eid=1(0x02f25610)/R, total 54272K, used 50176K, garbage 0
K (91.2% used/total, 0.0% garbage/used, 0 blocks)
 deployed objects
 ________________Size__Instances__FreeRatio__Class_________________
              35234568       10648           - java.util.HashMap
               5678900       10668           - [Ljava.util.HashMap$Entr
y;
               4456788        7436           - java.util.HashMap$Entry
               4321000         200           - java.util.WeakHashMap
               1234568         190           - [Ljava.util.WeakHashMap$
Entry;
                454400           4           - java.util.WeakHashMap$En
try
              51380224       29146 total

 "VJBStored" eid=3(0x02f25910)/B, total 54272K, used 50176K, garbage 10
```

```
004K (90.7% used/total, 19.9% garbage/used, 5 blocks)
 deployed objects
 _________________Size__Instances__FreeRatio__Class_________________
              35234568        10648          49 java.util.HashMap
               5678900        10668          43 [Ljava.util.HashMap$Ent
ry;
               4456788         7436          50 java.util.HashMap$Entry
               4321000          200          32 java.util.WeakHashMap
               1234568          190          45 [Ljava.util.WeakHashMap
$Entry;
                454400            4          22 java.util.WeakHashMap$E
ntry
              51380224        29146 total

 "ExplicitMemory-2" eid=2(0x02f25700)/B, total 54272K, used 50176K, gar
bage 0K (91.1% used/total, 0.0% garbage/used, 0 blocks)
 deployed objects
 _________________Size__Instances__FreeRatio__Class_________________
              35234568        10648           - java.util.HashMap
               5678900        10668           - [Ljava.util.HashMap$Ent
ry;
               4456788         7436           - java.util.HashMap$Entry
               4321000          200           - java.util.WeakHashMap
               1234568          190           - [Ljava.util.WeakHashMap
$Entry;
                454400            4           - java.util.WeakHashMap$E
ntry
              51380224        29146 total
```

- When the `eheapprof` command is not executed

```
Explicit Heap Status
--------------------
 max 31415926K, total 213971K, used 205369K, garbage 1234K (1.1% used/m
ax, 96.2% used/total, 0.0% garbage/used), 3 spaces exist

 Explicit Memories(0x12345678)

 "EJBMgrData" eid=1(0x02f25610)/R, total 154272K, used 150176K, garbag
e 1234K (97.0% used/total, 1.2% garbage/used, 0 blocks) Enable

 "VJBStored" eid=3(0x02f25910)/B, total 54272K, used 50176K, garbage 0
K (90.9% used/total, 0.0% garbage/used, 2 blocks) Enable

 "ExplicitMemory-2" eid=2(0x02f25700)/R, total 5427K, used 5017K, garba
ge 0K (92.1% used/total, 0.0% garbage/used, 0 blocks) Enable
```

5.6 JavaVM GC Log

The GC log is output to the JavaVM log file.

For details, see *5.7 JavaVM log (JavaVM log file)*.

5.7 JavaVM log (JavaVM log file)

The JavaVM log file is output by using extended options added by Hitachi in the standard JavaVM. To acquire this log, you need to specify the necessary options when starting the target J2EE server.

5.7.1 Options to output the JavaVM log file

The options to output the JavaVM log file are as follows:

- `-XX:+HitachiOutOfMemoryStackTrace`

  It is an option to output the stack trace when OutOfMemoryError occurs. Note that the JavaVM log file is output even if you specify `-XX:+HitachiOutOfMemorySize` and `-XX:+HitachiOutOfMemoryCause` that are specified when you specify the `-XX:+HitachiOutOfMemoryStackTrace` option.

- `-XX:+HitachiVerboseGC`

  It is an option to output the extended verbosegc information when GC is performed. For acquiring the extended verbosegc information, see *5.7.2 Acquiring the extended verbosegc information*.

- `-XX:+HitachiJavaClassLibTrace`

  This option is to output the API call trace of `System.gc()`, `System.exit()`, `System.runFinalizersOnExit()`, `Runtime.exit()`, `Runtime.halt()`, or `Runtime.runFinalizersOnExit()` APIs when either of them is executed.

  Note that when you specify the `-XX:HitachiJavaClassLibTraceLineSize` option, the number of characters in the output trace are within the specified number of characters (number of bytes). When the number of characters in one line exceeds the specified value, the first half of the character string after at is deleted and the specified number of characters is output.

- `-XX:+JITCompilerContinuation`

  This option enables the JIT compiler continuation functionality. If the JIT compilation fails due to a logical inconsistency in a method configuring the application, the JIT compiler continuation functionality log is output to the JavaVM log file.

For details on the output contents when specifying each option, see the following points in the *uCosminexus Application Server Definition Reference Guide*:

- *-XX:[+|-]HitachiOutOfMemoryStackTrace (Option for stack trace output)*
- *-XX:[+|-]HitachiVerboseGC (Option for extended verbosegc information output)*
- *-XX:[+|-]HitachiJavaClassLibTrace (Option to output the stack trace of class library)*
- *-XX:[+|-]JITCompilerContinuation (Option for the JIT compiler continuation functionality)*

5.7.2 Acquiring the extended verbosegc information

In the J2EE server `usrconf.cfg` file, if you specify the options shown in the table below, you can acquire extended verbosegc information. From the extended verbosegc information, you can acquire the information for estimating the Java heap area size, Metaspace area size, required for that server. Note that the stack trace at OutOfMemoryError occurrence time is also output in the Java log.

Table 5–23: Options to be specified for acquiring the extended verbosegc information

Optional	Meaning
`-XX:+HitachiVerboseGC`	Specify whether to output the extended verbosegc information. The information is output for each type of GC's internal areas: Eden, Survivor, Tenured, and Metaspace areas. Nothing is output by default. If `-XX:+HitachiVerboseGC` is specified, the extended verbosegc information is output and if `-XX:-HitachiVerboseGC` is specified, the extended verbosegc information is not output.
`-XX:+HitachiVerboseGCPrintDate`	Specify whether to display the date of log output at the beginning of each line of the log in which the extended verbosegc information is output.
`-XX:+HitachiVerboseGCCpuTime`	During the period between the start of GC until the end of GC, specify whether only the time spent in user mode and kernel mode by the GC execution thread should be displayed or the execution time should be displayed.
`-XX:HitachiVerboseGCIntervalTime=`*Time-interval*	Specify a numeric value (Unit: Seconds) as an output time interval for `-XX:+HitachiVerboseGC`. The default value for the time interval is 0 (Output every time GC occurs). Note that when you specify the time interval, the GC frequency during that time interval is also displayed.
`-XX:+HitachiVerboseGCPrintCause`	Specify whether to display the cause for the occurrence of GC in the log in which the extended verbosegc information is output.
`-XX:+HitachiCommaVerboseGC`	Specify whether the log in which the extended verbosegc information is output should be in CSV format. If the log is output in the CSV format, all the brackets (( ) [ ] < >) and delimiters of the extended verbosegc information are omitted, and the numeric values or character strings delimited by comma (,) are output.
`-XX:+HitachiVerboseGCPrintTenuringDistribution`	Specify whether to output Tenuring Distribution information of Survivor area. Nothing is output by default. For output format or output information, see *9.11 Tenuring distribution information output functionality of the Survivor area*.
`-XX:+HitachiVerboseGCPrintJVMInternalMemory`	Specify whether the heap information managed in JavaVM is output to the JavaVM log file.
`-XX:+HitachiVerboseGCPrintThreadCount`	Specify whether the Java thread count is output to the JavaVM log file to monitor the Java thread count.
`-XX:+HitachiVerboseGCPrintDeleteOnExit`	Specify whether the cumulative heap size allocated by JavaVM by invoking `java.io.File.deleteOnExit()` and the frequency of method invocation are output to the JavaVM log file.
`-XX:+PrintCodeCacheInfo`	Specify whether to output the usage of the code cache area, and also whether to output a message informing the user that the usage has reached the threshold value.

The format and the example of the log file output are as follows:

Output format

```
[id]<date>(Skip Full:full_count, Copy:copy_count)[gc_kind gc_info, gc_tim
e secs][DefNew::Eden: eden_info][DefNew::Survivor: survivor_info][Tenured
: tenured_info][Metaspace: metaspace_info][class space: class_space_info
]  [cause:cause_info][User: user_cpu secs][Sys: system_cpu secs] ][IM:jv
m_alloc_size, mmap_total_size, malloc_total_size][TC: thread_count][DOE: d
oe_alloc_size, called_count] [CCI: cc_used_sizeK, cc_max_sizeK, cc_infoK]
```

Description

- `id`: JavaVM log file identifier

  The following table describes the JavaVM log file identifiers. Use these identifiers for investigation by filtering log as per log contents (functions).

Table 5–24: JavaVM log file identifiers

Identifiers	Log contents
CCI	Code cache area information
CLT	Stack trace of class library
JCC	Information on the failure of JIT compilation
JMS	Information on the JIT complier thread that prevents JIT compilation
OMH	Information on the frequency of OutOfMemory occurrences
OOM	Exception information when OutOfMemoryError occurs and stack trace
PTD	Tenuring distribution information of Survivor area
VGC	Extended verbosegc information

- date: Date and time
- full_count: Frequency of skipping Full GC (Only when you specify `-XX:HitachiVerboseGCIntervalTime`)
- copy_count: Frequency of skipping copy GC (Only when you specify `-XX:HitachiVerboseGCIntervalTime`)
- gc_kind: GC type (Full GC or GC)
- gc_info: GC information (Area length prior to GC-> Area length after GC (Area size))
  (Example) 264K->0K(512K)
- gc_time: GC elapsed time (Unit: Seconds)
- eden_info: Eden information
- survivor_info: Survivor information
- tenured_info: Tenured information
- metaspace_info: Metaspace area information
- class_space_info: CompressedClassSpace information
- cause_info: Cause of GC
- user_cpu secs: CPU time spent in user mode by the GC thread (Unit: Seconds)
- system_cpu secs: CPU time spent in kernel mode by the GC thread. (Unit: Seconds)
- jvm_alloc_size: The size of the area currently in use, among the areas managed in JavaVM (size of the area currently in use among the total size of `mmap_total_size` and `malloc_total_size`) (only when `-XX:+HitachiVerboseGCPrintJVMInternalMemory` is specified)
- mmap_total_size: The total size of C heap allocated by mmap (in Windows VirtualAlloc) , among the areas managed in JavaV (only when `-XX:+HitachiVerboseGCPrintJVMInternalMemory` is specified)
- malloc_total_size: The total size of C heap allocated by malloc, among the areas managed in JavaVM (only when `-XX:+HitachiVerboseGCPrintJVMInternalMemory` is specified)
- thread_count: Java thread count (only when `-XX:+HitachiVerboseGCPrintThreadCount` is specified)
- doe_alloc_size: The accumulated heap size allocated by invoking `java.io.File.deleteOnExit()` (only when `-XX:+HitachiVerboseGCPrintDeleteOnExit` is specified)

- called_count: The number of times `java.io.File.deleteOnExit()` is invoked (only when `-XX:+HitachiVerboseGCPrintDeleteOnExit` is specified)
- cc_used_size: Size of the code cache area used during GC (unit: kilobyte) (only when `-XX:+PrintCodeCacheInfo` is specified)
- cc_max_size: Maximum size of the code cache area (unit: kilobyte) (only when `-XX:+PrintCodeCacheInfo` is specified)
- cc_info: Maintenance information (only when `-XX:+PrintCodeCacheInfo` is specified)

Example of output

Examples of output when the `-XX:+HitachiCommaVerboseGC` option is specified are as follows:

```
VGC,Fri Jan 23 21:37:50 2004,11,41,0,GC,16886,16886,65088,0.0559806,
4094,0,4096,447,447,448,12345,16439,60544,1116,1116,4096,0,0.0312500, 0.01
56250,729,928,0,509,2167,2054,2301,49152,2304
VGC,Fri Jan 23 21:37:55 2004,6,24,0,Full GC,65082,65082,65088,0.4294532,
4094,4094,4096,447,447,448,60541,60541,60544,1116,1116,4096,0,0.0156250, 0
.0312500,729,928,0,509,16,170,2301,49152,2304
```

5.7.3 Contents of the code cache area-related log

JavaVM speeds up the processing by executing the JIT compilation for the Java methods with a high invocation count and loop count. The JIT compile code generated by the JIT compilation is allocated to the code cache area.

Normally there is no problem if the code cache area size is the default value. However, depending on the scale of the execution environment and Java applications, the code cache area size might be depleted with the default value.

If the code cache area is depleted, JavaVM cannot execute the JIT compilation and, proper performance might not be obtained from the execution of Java applications. In such cases, enable `-XX:[+|-]PrintCodeCacheInfo` (option to output the code cache area information) or `-XX:[+|-]PrintCodeCacheFullMessage` (option to output the code cache area depletion message), and then monitor the usage of the code cache area and the messages that are output.

For details on these options, see *-XX:[+|-]PrintCodeCacheInfo (Option for the output of the code cache area information)* and *-XX:[+|-]PrintCodeCacheFullMessage (Option for the output of the code cache area depletion message)* in the *uCosminexus Application Server Definition Reference Guide*.

(1) Output contents of the message informing the user that the usage of the code cache area has reached the threshold value

The output format of the message informing the user that the usage of the code cache area has reached the threshold value is as follows:

```
[cc_id]<cc_date>CodeCache usage has exceeded the threshold.[cc_used_sizeK, c
c_max_sizeK, cc_infoK]
```

The following table describes the output items:

Output item	Explanation
`cc_id`	Outputs the CCI (JavaVM log file identifier).
`cc_date`	Outputs the data and time of JIT compilation.

Output item	Explanation
cc_used_size	Outputs the size of the code cache area used after the JIT compilation (unit: kilobyte).
cc_max_size	Outputs the maximum size of the code cache area (unit: kilobyte).
cc_info	Outputs the maintenance information.

(2) Output contents of the message informing the user that the code cache area has depleted

The output format of the message informing the user that the code cache area has depleted is as follows:

```
[cc_id]<cc_date>CodeCache is full. Compiler has been disabled.[cc_used_sizeK
, cc_max_sizeK, cc_infoK]
```

The following table describes the output items:

Output item	Explanation
cc_id	Outputs the CCI (JavaVM log file identifier).
cc_date	Outputs the date on which the Java method became subject to the JIT compilation.
cc_used_size	Outputs the used size of the code cache area when the Java method became subject to the JIT compilation (unit: kilobyte).
cc_max_size	Outputs the maximum size of the code cache area (unit: kilobyte).
cc_info	Outputs the maintenance information.

5.8 Message log output by JavaVM (Standard output and error report file)

When the JavaVM crashes, JavaVM outputs the debug information to the standard output and error report file.

The following are cases when the debug information is output to the error report file:

- When a signal occurs in JNI
- When C heap is insufficient in the JavaVM
- When an unexpected signal occurs in the JavaVM
- When an Internal Error (internal logical error) occurs in the JavaVM

The contents of message log output in the following cases are as follows.

Table 5–25: Message log output by JavaVM

Message type	Output destination
When a signal occurs in JNI or message in JavaVM#	Standard output Error report files
Insufficient C heap message#	Standard output Error report files
Internal Error occurrence message#	Standard output Error report files
Thread creation failure message#	Standard output

#
The JavaVM proprietary output destination or output contents exist.

Note that the thread creation failure message is output only in the standard output.

5.8.1 When a Signal Occurs

When a signal occurs, the items shown below are output to the log. The JavaVM extended contents are included in the output contents.

- Abnormal termination location and signal type#
- Current thread information
- Save destination address of signal information#
- Signal information
- siginfo information# (in UNIX)
- Register information
- Information saved from the beginning of the stack
- Command code information
- Stack trace
- Thread information
- VM status

- Memory information#
- Usage status of Java heap#
- Card table map address display
- Polling page address display
- Large pages allocation failure information
- CodeCache information
- Event information
- Library
- Command and VM parameters#
- Environment variable
- Registered signal handlers
- Machine information#
- System name, CPU, actual memory and VM information
- Time information#
- Command line of the javatrace startup command# (in UNIX)

#

These are the output contents extended by Hitachi.

Each output contents are described below.

(1) Abnormal termination location and signal type

Any one of the following contents are output in compliance with the status at the time of abnormal termination. These contents are extended by JavaVM.

(a) When a signal is detected

The following message is output:

```
#
# A fatal error has been detected by the Java Runtime Environment:
```

The following contents are output:

Contents output when a signal is detected

```
#
# A fatal error has been detected by the Java Runtime Environment:
#
# Occurred-signal-name (signal-number) at pc=PC-address, pid=Process-ID, tid
=Thread-ID
#
# JRE version :(jre version information)
# Java VM: Java HotSpot(TM) VM-type (Sun-version-information-Hitachi-version
-information-build-date mixed modeOS-name-CPU-type)
# Problematic frame:
```

```
# <type-code> <library-name-where-signal-occurred+Offset
#
```

Note:

When you are able to extract a function name where the signal occurred, that function name and offset may be displayed in continuation to *library-name-where-signal-occured+offset*.

(b) When an internal logical error occurs

The following contents are output:

Contents output when an internal logical error occurs

```
#
# Internal Error (file-name:number-of-lines or internal-error-code), pid=pro
cess-ID, tid=thread-ID
# Internal logical Error: Internal logical error message
#
# JRE version: JRE-version-information
# Java VM: Java HotSpot(TM) VM-type (Sun-version-information-Hitachi-version
-information-build-date mixed modeOS-name-CPU-type compressed-OOP)
#
# core-file-information
```

Note:

Either a combination of the file name and number of lines or the Internal Error code is output to the Internal Error. In `Internal logical Error`, any one of `fatal error`, `guarantee(logical type)failed`, or `Error` is output, according to the type of the internal logical error.

(2) Current thread information

The following three types of information are output to the messages according to the thread type:

```
Current thread (address): thread-name "thread-name" [_state, id=thread-ID, s
tack(start-address,end-address)]

Or
Current thread (address): thread-name [_id=thread-ID, stack(start-address,en
d-address)]

Or
Current thread is native thread
```

(3) Save destination address of the signal information

The following contents are output. These contents are extended by JavaVM.

```
siginfo address: address, context address: address
```

(4) Signal information

The following contents are output.

In Windows

EXCEPTION_ACCESS_VIOLATION(Read violation)

```
siginfo: ExceptionCode=signal-number, reading address address
```

EXCEPTION_ACCESS_VIOLATION(Write violation)

```
siginfo: ExceptionCode=signal-number, writing address address
```

EXCEPTION_ACCESS_VIOLATION(Others)

```
siginfo: ExceptionCode=signal-number, ExceptionInformation=additional-info
rmation
```

Other than EXCEPTION_ACCESS_VIOLATION

```
siginfo: ExceptionCode=signal-number, ExceptionInformation=additional-info
rmation-1 additional-information-2 ...
```

In UNIX

```
siginfo:si_signo=_occurred-signal-number (occurred-signal-name), si_errno
:_number, si_code:_number(signal-reason-type), si_addr:_address
```

(5) siginfo information (in UNIX)

The following contents are output. These contents are extended by JavaVM.

```
siginfo structure dump (location: siginfo-address)
siginfo-address siginfo-address siginfo-address siginfo-address
 ...
siginfo-address siginfo-address siginfo-address siginfo-address
```

Note:

The *siginfo-address* is output in hexadecimal.

(6) Register information

The following contents are output. However, these contents are not output in the case of internal logical errors.

```
Registers: register-information
...
```

Note:

In UNIX, different BSP register value and debugger (gdb) value are output. This is because, in debugger, the contents of the backing store area that indicates BSP is output and the position that indicates BSP is modified.

(7) Information saved from the top of stack

The following contents are output. However, these contents are not output in the case of internal logical errors.

```
Top of Stack: (sp=Address-of-stack-pointer)
Address: Saved-contents
...
```

Note:

saved-contents are output in hexadecimal.

(8) Stack trace

The following contents are output. However, these contents are not output when the Current thread is other than JavaThread.

```
Java frames: (J=compiled Java code, j=interpreted, Vv=VM code)
 Stack-trace
...
```

(9) Thread information

The following contents are output:

```
Java Threads: ( => current thread )
 address JavaThread "thread-name" [state, id=thread-ID, stack(start-address,
end-address)]
 :
=>address JavaThread "thread-name" [state, id=thread-ID, stack(start-address
,end-address)]
Other Threads:
 address thread-name [stack(start-address,end-address)] [id=thread-ID]
 :
```

(10) VM status

The following contents are output.

```
VM state: current-status

VM Mutex/Monitor currently owned by a thread: <mutexs/moniter>
```

Note:

The lock information may be output in continuation with this information.

(11) Memory information

The following contents are output. These contents are extended by JavaVM.

```
Memory:
secure-memory-function:addressStart-address - End-address (size: size)
 ...

 Heap Size: secured-memory-size
 Alloc Size: memory-size-in-use
 Free Size: unused-memory-size
```

memory-securing-function is either `mmap()` or `malloc()`. The address is displayed in hexadecimal.

The unit of each type of memory size is bytes.

(12) Java heap usage status

The following contents are output. These contents are extended by JavaVM.

```
Heap#
Java-heap-information
```

#

The header section differs between the extended thread dump and the error report file.

For the extended thread dump	`Heap Status` `----------`
For the error report file	`Heap`

(13) Card table map address display

The following contents are output.

```
Card table byte_map: [address,address] byte_map_base: address
```

(14) Polling page address display

The following contents are output.

```
Polling page: address
```

(15) Large pages allocation failure information

If memory allocation by using the `mmap()` function fails, the number of failures is output.

```
Large page allocation failures have occurred number-of-times times
```

(16) CodeCache information

The following contents are output.

```
CodeCache: size=total-size used=used-size max_used=maximum-size free=free-sp
ace-size
 bounds [bottom, commit-addr, reserve-addr]
 total_blobs=total-number-of-CodeBlobs nmethods=total-number-of-methods adap
ters=total-number-of-adapters
 compilation: enabled or disabled
```

(17) Event information

All contents of the event buffer are output.

```
event-type-name (number-of-events events):
event-record
  :
```

Event information is managed in a ring buffer, and the maximum number of events retained per event type is 10. If the number of events is 0, `No events` is output to *event-record*.

The following are examples of output for each *event-type* that can be output.

- Compilation events

  JIT compilation information

```
Compilation events (10 events):
Event: 0.923 Thread 0x00002aaab2f01800  389    b        java.io.FileOutput
Stream::write (12 bytes)
Event: 0.923 Thread 0x00002aaab2f01800 nmethod 389 0x00002aaaac3ea490 cod
e [0x00002aaaac3ea5e0, 0x00002aaaac3ea668]
    :
```

- GC Heap History

  Information before GC and information after GC

```
GC Heap History (4 events):
Event: 23.719 GC heap before
{Heap before GC invocations=0 (full 0):
 def new generation   max 154880K, total 9664K, used 345K (0.2% used/max,
3.6% used/total)
    :
```

- Deoptimization events

  Deoptimization information

```
Deoptimization events (10 events):
Event: 0.818 Thread 0x00002aaaaba7e000 Uncommon trap 24 fr.pc 0x00002aaaac
3d1eec
Event: 0.818 Thread 0x00002aaaaba7e000 Uncommon trap 54 fr.pc 0x00002aaaac
3d0dd8
    :
```

- Internal exceptions

  Internal exception information

```
Internal exceptions (2 events):
Event: 0.025 Thread 0x00002aaaaba7e000 Threw 0x00000000db606140 at /hotspo
t/src/share/vm/prims/jni.cpp:4008
Event: 0.061 Thread 0x00002aaaaba7e000 Threw 0x00000000db649980 at /hotspo
t/src/share/vm/prims/jvm.cpp:1167
    :
```

- Events

  Class loader information and other information

```
Events (10 events):
Event: 0.080 loading class 0x00002aaab302e990
Event: 0.080 loading class 0x00002aaab302e990 done
Event: 4.286 Executing VM operation: EnableBiasedLocking
```

```
Event: 4.286 Executing VM operation: EnableBiasedLocking done
    :
```

(18) Libraries

The list of loaded libraries is output in continuation to the following contents.

```
Dynamic libraries:
libraries
...
```

(19) Command and VM parameters

The following contents are output. These contents are extended by JavaVM.

```
Command : command-line

Java Home Dir : JDK-execution-environment-install-directory
Java DLL Dir : JDK-library-install-directory
Sys Classpath : system-class-path
User Args :
command-option-1
command-option-2
...
```

(20) Environment variables

The following contents are output.

```
Environment Variables:
environment-variable=value
...
```

(21) Registered signal handlers

The following contents are output.

```
Signal Handlers:
signal-type:
 [signal-handler-address], sa_mask[0]= mask-signal, sa_flags=special-flag
...
Changed Signal Handlers -
signal-type: [signal-handler-address], sa_mask[0]=signal-mask, sa_flags=spec
ial-flag
...
```

The meaning of output contents are as follows:

- signal-type: It is the signal name defined in `/usr/include/sys/signal.h`.
- signal-handler-address: It is the signal handler address output in hexadecimal. It may also be displayed in the *library-name*+*offset* format.

- signal-mask: It is the value where the sa_mask field value of the structure extracted by sigaction() is output in hexadecimal.
- special-flag: It is the value where the sa_flags field value of the structure extracted by sigaction() is output in hexadecimal.

(22) Machine information

The following contents are output. These contents are extended by JavaVM.

```
Host: host-name:IP-address
```

Note:

Multiple IP addresses may be displayed in IP-address.

(23) System name, CPU, actual memory, and VM information

The following contents are output.

In Windows

```
OS:OS-version

CPU: number-of-CPUs-that-can-be-used, CPU-type

Memory:actual-memory-information

vm_info:VM-information
```

In UNIX

```
OS:OS version

[uname:uname output]
[libc:version-number-of-libc(at-times-version number-is-not-output)]
[rlimit:limit-value]
[load average:load-average]
[/proc/meminfo:/proc/meminfo contents]


CPU:number-of-CPUs-that-can-be-used,CPU-type

Memory:actual-memory-information

vm_info:VM-information
```

Time information

The following contents are output:

```
time: execution-date

elapsed time: execution-time seconds (formatted-execution-time-output)
```

Note:

An example of the execution date is as follows:

Example: Wed Aug 25 14:55:04 2004

It is difficult to understand the execution time given only in unit of seconds, so the execution time is output in (*formatted-execution-time-output*) in the format of days, hours, minutes, and seconds.

(Example) `elapsed time: 900 seconds (0d 0h 15m 0s)`

(24) Command line of javatrace start command (in UNIX)

The following contents are output. These contents are extended by JavaVM.

```
# You can get further information from javatrace.log file generated
# by using javatrace command.
# usage: javatrace core-file-name loadmodule-name [out-file-name] [-l(librar
y-name)...]
# Please use javatrace command as follows and submit a bug report
# to Hitachi with javatrace.log file:
#[installation-directory/bin/javatrace core-file load-module]
```

5.8.2 When C Heap Is Insufficient

When C heap is insufficient, the message output, and dump output or core dump generation is performed in the following order:

1. A message log indicating insufficient C heap is output to the error report file and standard output.
2. If the memory is insufficient during the execution of 1., a simple message is output to the standard output.
3. In UNIX, if the memory is still insufficient during the output of a simple message, output of the message and error log file is interrupted and a core dump is generated.

Each output format is described below.

(1) Output contents of a message log indicating insufficient C heap

The output format of a message log indicating insufficient C heap is described below. This format is common for the error report file and standard output.

- In Windows

```
Exception in thread <ThreadName> java.lang.OutOfMemoryError:requested <n>
bytes [ for <message>].

Memory Status
-------------
       Memory in use: utilization-rate%
       Physical memory: free-memory-size/total-memory-sizefree
       Virtual memory: free-memory-size/Total-memory-sizefree
       Paging file: free-volume/total-volumefree

Heap Status
-----------
```

```
Java-heap-information
-----------
Stack Trace
-----------
stack-trace
JVM Internal Memory Status
--------------------------
area-information-managed-by-the-unique-memory-management-function
Insufficient memory for malloc. JVM generates core file.
-----------
```

- In AIX or Linux

```
Exception in thread <ThreadName> java.lang.OutOfMemoryError: requested <n
> bytes [for <message>].

Memory Status
-------------
 maximum size of data segment
 soft(current) limit: software-limit-value-acquired-by-getrlimit(RLIMIT_DA
TA) kbytes (Hexadecimal)
 hard limit: hardware-limit-value-acquired-by-getrlimit(RLIMIT_DATA) kbyte
s (Hexadecimal)
 current end of the heap: value-acquired-by-sbrk(0)
 JVM allocation size by malloc: memory-size-allocated-to-JavaVM kbytes (He
xadecimal)
 malloc information
     total space in arena            :mallinfo.arena value
     number of ordinary blocks       :mallinfo.ordblks value
     number of small blocks          :mallinfo.smblks value
     number of holding blocks        :mallinfo.hblks value
     space in holding block headers  :mallinfo.hblkhd value
     space in small blocks in use    :mallinfo.usmblks value
     space in free small blocks      :mallinfo.fsmblks value
     space in ordinary blocks in use :mallinfo.uordblks value
     space in free ordinary blocks   :mallinfo.fordblks value
     cost of enabling keep option    :mallinfo.keepcost value

Heap Status
-----------
Java-heap-information
-----------
Stack Trace
-----------
stack-trace
-----------

JVM Internal Memory Status
--------------------------
area-information-managed-by-the-unique-memory-management-function
```

When such a message is output, take appropriate measures such as reducing C heap.

The output contents are as follows:

Table 5–26: Output items of message log when C heap is insufficient

Output items	Description
ThreadName	The thread name extracted by the `Thread#getName()` method is output.
n	The size of memory securing requests is output.
message	The internal message required by maintenance personnel for investigation is output. In some cases it may not be output.
Java heap information	The usage status of Java heap is output.
Stack trace	The stack trace is output when the thread for which the memory is insufficient is the thread that is executing a Java code. The stack trace is not output when the memory is insufficient in the thread that is executing internal process such as compilation process in JavaVM.

(2) Output contents of message showing memory insufficiency

If memory becomes insufficient while a message log indicating insufficient C heap is being output, the process cannot be continued. In this case, a simple message in the following format is output to the standard output.

```
java.lang.OutOfMemoryError:requested <n> bytes for <message>
```

The output contents are as follows:

Table 5–27: Output contents of a simple message when the memory is insufficient

Output items	Description
n	The size of memory securing requests is output.
message	The internal message required by maintenance personnel for investigation is output.

(3) Output contents of the message indicating core dump generation

If the memory is still insufficient when a simple message is output, output of the message and error log file is interrupted and a core dump is generated. When a core dump is generated, the message in the following format is output in the standard output.

```
Can't create logs because of memory shortage.
Insufficient memory for malloc. JVM generates core file
```

5.8.3 When an Internal Error Occurs

When an internal error that is a logical error within JavaVM occurs, the following information is output.

- Abnormal termination location and signal type
- Current thread information
- Save destination address of signal information#
- Thread information
- VM status

- Memory information#
- Heap information#
- Card table map address
- Polling page address
- Large pages allocation failure information
- CodeCache information
- Event information
- Library
- Command VM parameter#
- Environment variable
- Registered signal handlers
- Machine information#
- System name, CPU, actual memory and VM information
- Time information
- Command line of the javatrace startup command# (in UNIX)

#
These are the output contents extended by Hitachi.

For the output format of each information, see *5.8.1 When a Signal Occurs*.

5.8.4 When Thread Creation Fails

When memory insufficiency (OutOfMemoryError) occurs and a new thread cannot be created, the number of threads at that time is output to the standard output. The threads that could not be created are also included in this number of threads.

An example of output when thread creation failed due to memory insufficiency is as follows:

```
java.lang.OutOfMemoryError:unable to create new native thread.1200 threads e
xist.
...
```

An example of output when thread creation failed at JavaVM startup time is as follows:

```
Error occurred during initialization of VM
Could not create thread for VM:VM Thread.5 threads exist.
```

5.9 OS status information and OS log

Confirm the existence of an abnormal trend from the acquired OS status and log information. For more details on this, see the manuals provided with the OS.

5.10 JavaVM stack trace information

Of the information output in the stack trace, the contents extended by JavaVM are described here.

When an error occurs in a server and applications, you can investigate the cause for error occurrence by confirming the stack trace contents until the error occurred.

The stack trace is output at either of the following timings:

- When an exception occurs in the J2EE servers or J2EE applications
- When an exception occurs in batch servers or batch applications
- When JavaVM thread dump is output

In JavaVM, you can output the information of local variables in Java methods in the stack trace by specifying start options when starting the server. The information of local variables defined in Java methods is effective for analyzing the cause of errors when exceptions occur.

Note that, the *Local variables* referenced here are the objects (this) that are passed to the methods as arguments and are invoked by instance methods. In the *local variable information*, these local variable names, type names, and local variable values are output. Note that the type name is the basic type name, class name (including interface name), or array type name.

The following table describes the options to output the local variable information to stack trace. For details on settings for acquiring the JavaVM material, see the subsection *3.3.17 Settings for Acquiring the JavaVM Material*.

Table 5–28: Options to output the local variable information to stack trace

Startup options	Timing to output stack trace	Option that can be specified simultaneously
`-XX:+HitachiLocalsInThrowable`	When an exception occurred in servers or applications#	• `-XX:+HitachiLocalsSimpleFormat` • `-XX:+HitachiTrueTypeInLocals` • `-XX:HitachiCallToString=`*applicable-range*
`-XX:+HitachiLocalsInStackTrace`	When JavaVM thread dump is output	• `-XX:+HitachiLocalsSimpleFormat` • `-XX:+HitachiTrueTypeInLocals`

#

However, when the exception occurred is `java.lang.StackOverflowError` or `java.lang.OutOfMemoryError`, the local variables are not output to stack trace.

The contents output is described with an example as base when you specify each of these options. For details on the items output when each option is specified, see the following points in the *uCosminexus Application Server Definition Reference Guide*:

- *-XX:[+|-]HitachiLocalsInThrowable (Option for collecting the local variable information when an exception occurs)*
- *-XX:[+|-]HitachiLocalsInStackTrace (Option to output the local variable when the thread dump is output)*

5.10.1 When the -XX:+HitachiLocalsInThrowable Option Is Specified

For each stack frame information of the stack trace information output by the `java.lang.Throwable.printStackTrace` method, the local variable information within the method corresponding to that stack frame is inserted and output.

(1) Example of output in standard format and simple output format

This is the example of output when you specify the `-XX:+HitachiLocalsInThrowable` option only as a function to output local variables.

The example of Java program and output of the corresponding local variable information within the stack trace are as follows:

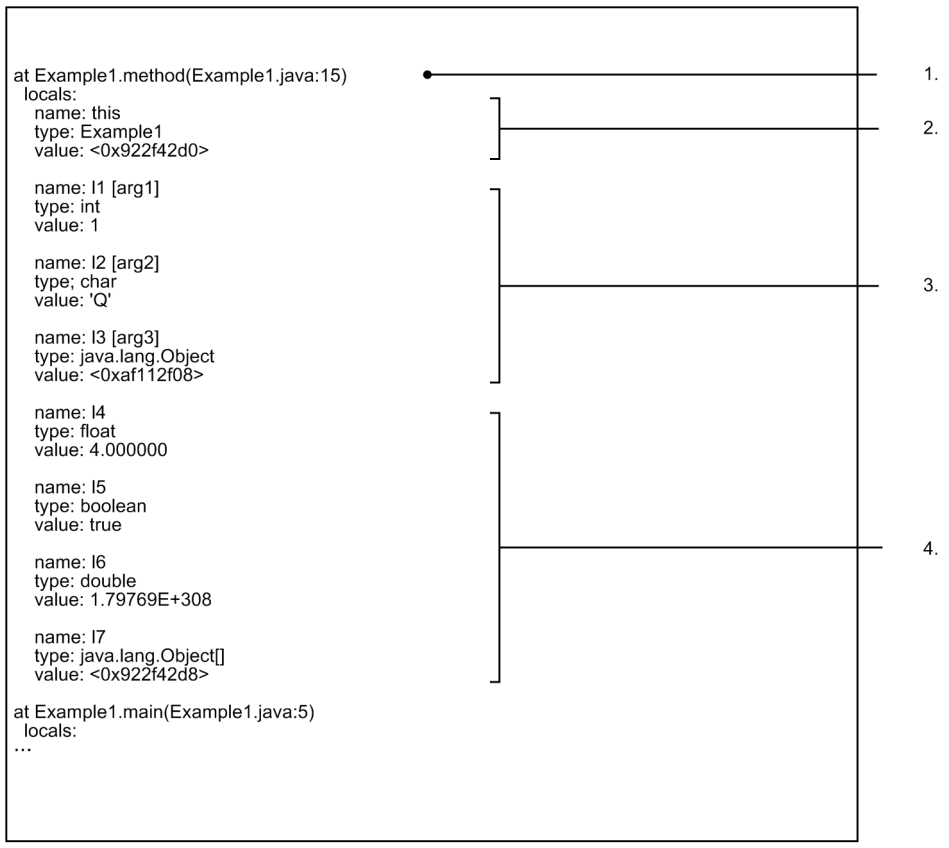
Java program example 1

```
class Example1 {
    public static void main(String[] args) {
        Example1 e1 = new Example1();
        Object obj = new Object();
        e1.method(1, 'Q', obj); // Execute-e1.method (5th line).
    }

    void method(int l1, char l2, Object l3) {
        float l4 = 4.0f;
        boolean l5 = true;
        double l6 = Double.MAX_VALUE;
        Object[] l7 = new Object[10];

        try {
        <Exception occurred!> // methodProcess-when-there-is-an-exception-in
-process-of-method(15th line).
        } catch (Exception e) {
          e.printStackTrace(); // output-stack-trace-information (17th line
).
        }
    }
}
```

The example of output is described below.

This example is the stack trace information output by `e.printStackTrace` method on the 17th line when an exception occurs in e1.method method executed on the 5th line of example 1 of the Java program.

Figure 5–2: Output example of local variable information for example 1 of Java program (When the class file is created by specifying -g option or -g:vars option)



The output contents are as follows:

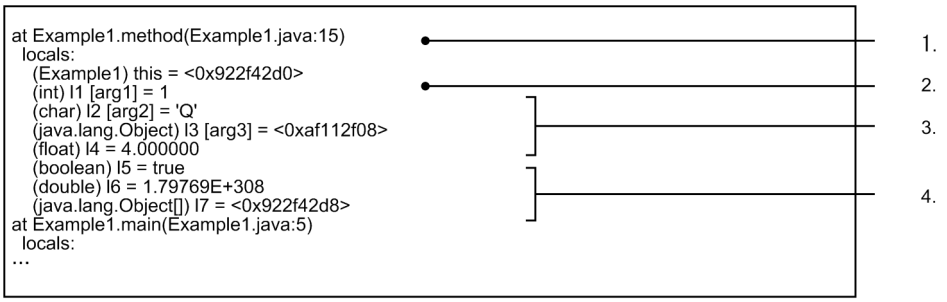
1. The information of the method that executes the stack trace output process is output. This example shows the output of the stack trace information when an exception occurs on the 15th line of the Java program 1.
2. As the local variable information, the information of objects invoked by the instance method is output. In this example, the class name and address of the object of example 1 class created on the 3rd line of example 1 of the Java program is output.
3. As local variable information, the information of the values of local variables specified as arguments of the method are output. The [arg*] after variable name is the information indicating the number of the method argument. The values specified when executing the e1.method method on the 5th line of example 1 of Java program are output.

   Note that for local variables 11 and 12, the actual value is output since they are basic type (int type and char type) variables. The local variable l3 is a `java.lang.Object` class type variable, as a result, the address is output.
4. As local variable information, from the local variables in the method, information of the values of local variables that are not specified as method argument is output.

   Note that since the local variables l4 to l6 are of basic type (float type, boolean type, and double type), their actual value is output. Since the local variable l7 is the `java.lang.Object` class type variable, the address is output.

The example of output in the simple output format when you specify `-XX:+HitachiLocalsSimpleFormat` option, is described below. Further, the explanation of the output contents is the same as that for standard format.

Figure 5–3: Output example when -XX:+HitachiLocalsSimpleFormat option is specified

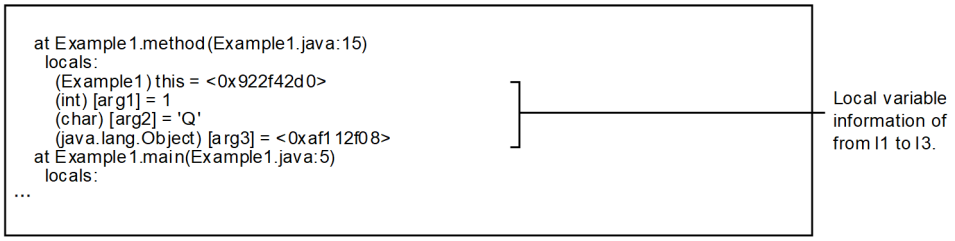


Moreover, if the `-g` option or `-g:vars` option is not specified when executing `javac` command, and since there is no local variable information, the output contents are restricted in the following manner. This is same as when executing the native method.

- The local variables that can be output are only the arguments that are passed to the method and objects (this) invoked by the instance method.
- For the arguments passed to the method, the variable name is not output. Only the argument number is output.
- In the case of the native method, the value when the native method is invoked is output as the value of the local variable. It is not the output result that reflects the result of the executing native method.

The example of output in the case of Java program example 1, when the local variable information does not exist, is described below. The example of output in the simple output format is as follows:

Figure 5–4: Output example when local variable information does not exist (Simple output format)



There are following differences as compared to Figure 5-3.

- The variable names of the arguments (l1 to l3) passed to the method is not output.
- As the value is output when the method is invoked, the information (l4 to l7) for the local variables set in the method is not output.

(2) Example of output when class or array type variables are output as a character string

When the local variables to be output are of class or array type, there are times when the information required for troubleshooting cannot be acquired in the value expression of only the address. At this time, if you specify the `-XX:HitachiCallToString` option, you can acquire the value of class or array type variable as a character string. In options, you can specify `minimal` or `full` as the applicable scope.

When you specify `-XX:HitachiCallToString=minimal` option, among the classes in the `java.lang` package, String, StringBuffer, Boolean, Byte, Character, Short, Integer, Long, Float, or Double are the targets of a parameter. When specifying the `-XX:HitachiCallToString=full` option, all the classes are targets.

An example of the Java program and example of output when the `-XX:HitachiCallToString` option is specified is described below. Further, the simple output format is used below.

Java program example 2

```
class Example2 {
    public static void main(String[] args) {
        Example2 e2 = new Example2();
        e2.method();// Executes-e2.method-method (4th line).
    }

    void method() {
        String l1 = "local 1";
        StringBuffer l2 = new StringBuffer(l1);
        l2.append(" + local 2");
        Boolean l3 = new Boolean(false);
        Character l4 = new Character('X');
        Long l5 = new Long(Long.MIN_VALUE);
        Object l6 = new Thread();
        Object[] l7 = new Thread[10];

        try {
          <Exception occurred!> // Process-when-exception-occurred-in-proces
s-of-method (18th line).
        } catch (Exception e) {
          e.printStackTrace(); // Output stack trace information (20th line
).
        }
    }

    public String toString() {
      return "I am an Example2 instance.";
    }
}
```

An example of output when specifying the `-XX:HitachiCallToString=minimal` option is described below:

This example is the stack trace information output by the 20th line `e.printStackTrace` method when an exception occurs in the `e2.method` method executed on the 4th line of example 1 of the Java program.

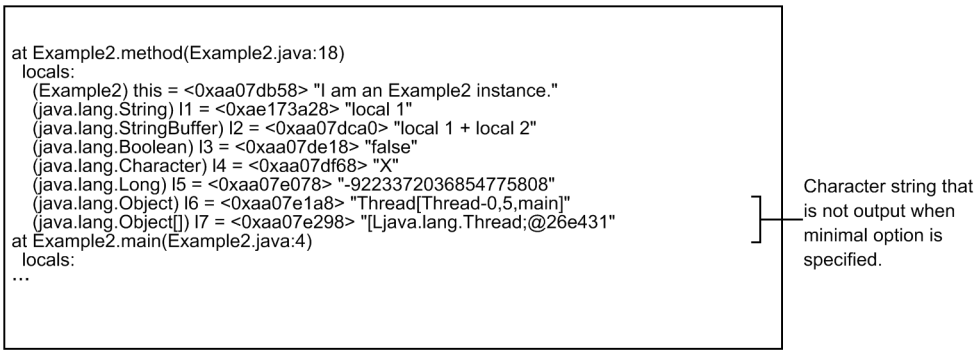
Figure 5–5: Output example when -XX:HitachiCallToString=minimal option is specified (Simple output format)

The output contents are as follows:

1. Among the local variables of class type, this information is that of the local variables of class type for which a character string is to be output. In continuation to the address, the value converted into a character string is output.
2. Among the local variables of class type, this information is that of local variables of class type for which a character string is not to be output. Only the address is output.

Shown below is the example of output when you specify the `-XX:HitachiCallToString=full` option.

Figure 5–6: Output example when -XX:HitachiCallToString=full option is specified (Simple output format)



There are following differences as compared to Figure 5-5.

- When you specify minimal, the character string is output even for the local variables (l6 and l7) of the arrays of `java.lang.Object` and `java.lang.Object` class for which a character string is not to be output.

However, the character string is not output and only the address is output, such as in the following cases:

- If the value of the local variable is null
- If an error occurs again when a character string is being output, you cannot acquire the value successfully.

Important note

The object output in the local variable information may be operated in parallel by other threads. For this reason, the character string output by specifying this option may be different than the information corresponding to that object when an exception actually occurs.

(3) Example of output when the actual type name of class or array type variables is to be output

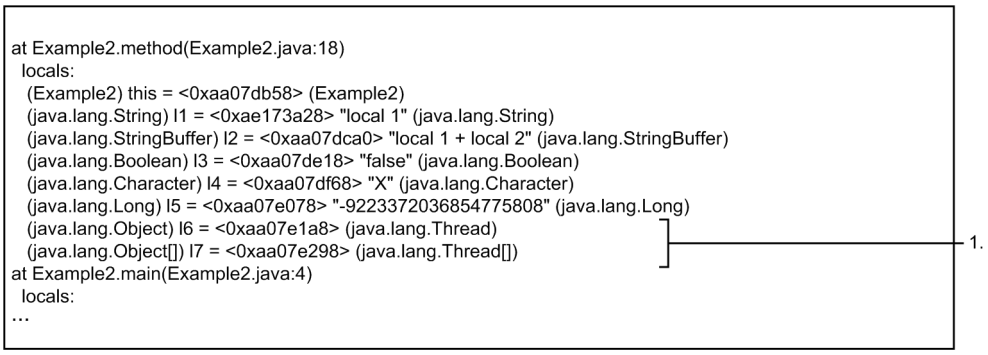
When local variables that are output are of class or array type, there are cases when that variable type name and the type name of actually substituted value differ. For example, depending on the inheritance relationship of classes or the implementation relationship of interfaces, values of different type names are substituted in variables.

In this case, by specifying the `-XX:+HitachiTrueTypeInLocals` option, you can add the type name of a value actually substituted to the class or array type variables and acquire the character string.

The acquired character string is output by adding one single byte space at the end of the value expression and enclosing with parentheses (()). At this time, there is no limit on the output character string length.

An example of output when you specify `-XX:+HitachiTrueTypeInLocals` option is described below. Simple output format is shown below. The program to be executed is example 2 of the Java program. Moreover, this example is an example where you specify the `-XX:HitachiCallToString=minimal` option.

Figure 5–7: Output example when -XX:+HitachiTrueTypeInLocals option is specified (Simple output format)



The output contents are as follows:

1. The variable type name is an array of `java.lang.Object` class or `java.lang.Object` class but the type name of the actually substituted value is output as the array of `java.lang.Thread` class and `java.lang.Thread` class.

Note that you can specify the `-XX:+HitachiTrueTypeInLocals` option along with the `-XX:HitachiCallToString=full` option.

5.10.2 When the -XX:+HitachiLocalsInStackTrace Option Is Specified

For each stack frame information of stack trace information in a thread dump, the local variable information in the method corresponding to that stack frame is inserted and output.

The output format and output contents are the same as the contents output to the standard output when you specify `-XX:+HitachiLocalsInThrowable`.

However, specifying this option with the `-XX:+HitachiLocalsInStackTrace` option is invalid.

- `-XX:HitachiCallToString` option

The example of Java program and output of the corresponding local variable information within the stack trace are as follows:

Java program example 3

```
class Example3 {
   public static void main(String[] args) {
        Example3 e3 = new Example3();
        e3.method();
    }

    synchronized void method() {
        int l1 = 1;
        float l2 = 2.0f;
```

```
        String l3 = "local 3";
        Character l4 = new Character('X');
        Object l5 = new Thread();
        Object[] l6 = new Thread[10];

        The-thread-dump-is-output-here!
    }
}
```

The example of output is described below. This is an example of the following cases.

- The class file is created by specifying the `-g` option or `-g:vars` option
- `-XX:+HitachiLocalSimpleFormat` option is specified
- `-XX:+HitachiTrueTypeInLocals` option is specified

Figure 5–8: Output example of the local variable information for example 3 of a Java program

```
...
"main" prio=1 tid=0xb6e88d20 nid=0xb7492080 runnable [bfffb000..bfffb474]
        at Example3.method(Example3.java:15)
        - locked <0xab040550> (a Example3)
          locals:
            (Example3) this = <0xab040550> (Example3)
            (int) l1 = 1
            (float) l2 = 2.000000
            (java.lang.String) l3 = <0xaf112cc0> (java.lang.String)
            (java.lang.Character) l4 = <0xab040698> (java.lang.Character)
            (java.lang.Object) l5 = <0xab0407c8> (java.lang.Thread)
            (java.lang.Object[]) l6 = <0xab0408b8> (java.lang.Thread[])
        at Example3.main(Example3.java:4)
          locals:
            (java.lang.String[]) args [arg1] = <0xab040540> (java.lang.String[])
            (Example3) e3 = <0xab040550> (Example3)
...
```

5.11 Event log of Explicit Memory Management functionality

This subsection describes the contents of event log of the Explicit Memory Management functionality.

The event log of Explicit Memory Management functionality is output at the timing when events related to the Explicit Memory Management functionality, such as occurrence of GC, initialization, release, extension of Explicit memory block, moving object to Explicit memory block occur. The output contents differ as per the settings of log output level specified in the Java VM start option.

The event log of the Explicit Memory Management functionality can be used for the following purpose:

- Tuning of Explicit HEAP
- Debug of application implementing the Explicit Memory Management functionality
- Statistics collection

For details on the changes in the size of Explicit heap, number of Explicit memory blocks, and changes in each area size of Java heap, see the event log. Furthermore, event log is output as a text file.

5.11.1 Output trigger of event log of the Explicit Memory Management functionality

This subsection describes the output trigger for event log of the Explicit Memory Management functionality. The events triggered differ depending on the set log output level.

The log output level comprises of three types; `normal`, `verbose` and `debug`. When you specify `verbose`, besides the contents output to `normal`, the contents corresponding to `verbose` are output. When you specify `debug`, besides the contents output to `verbose`, the contents corresponding to `debug` are output.

The following table describes the correspondence between log output levels and the events that trigger the output. Prefixes output for the corresponding events are also displayed in the following table. See the reference sections for contents output to each event.

Table 5–29: Correspondence between log output levels and events that trigger the output

Log output level	Events that trigger the output	Prefix#	Reference
`normal`	GC occurrence (Output Explicit heap usage status)	`ENS`	*5.11.3(1)*
	Explicit memory block explicit release process	`ENS`	*5.11.3(2)*
	Java heap overflow in Explicit memory block explicit release process	`ENS`	*5.11.3(3)*
	Automatic release processing of the explicit memory block	`ENS`	*5.11.3(4)*
	Java heap overflow in the automatic release processing for the explicit memory block	`ENS`	*5.11.3(5)*
	Error in opening the automatic allocation configuration file for explicit memory management	`ENA`	*5.11.3(6)*

Log output level	Events that trigger the output	Prefix#	Reference
	Error in parsing the automatic allocation configuration file for explicit memory management	ENA	*5.11.3(7)*
	Automatic allocation error in explicit memory management	ENA	*5.11.3(8)*
	Error in opening the configuration file of the functionality for specifying the classes to be excluded from the Explicit Memory Management functionality	ENO	*5.11.3(9)*
	Error in parsing the configuration file of the functionality for specifying the classes to be excluded from the Explicit Memory Management functionality	ENO	*5.11.3(10)*
verbose	Initialization of Explicit memory block	EVO	*5.11.4(1)*
	Initialization failure of Explicit memory block	EVO	*5.11.4(2)*
	Disabling sub status of Explicit memory block	EVO	*5.11.4(3)*
	Object generation to Explicit memory block	EVS	*5.11.4(4)*
	Migration to Explicit memory block (Output detail information)	EVS	*5.11.4(5)*
	Explicit memory block explicit release process (Output detail information)	EVS	*5.11.4(6)*
	Release reservation of Explicit memory block by finalize	EVO	*5.11.4(7)*
	Automatic allocation to explicit memory management	EVA	*5.11.4(8)*
debug	Migration of object to Java heap by explicit release of Explicit memory block	EDO	*5.11.5(1)*
	Initialization of Explicit memory block (Output detail information)	EDO	*5.11.5(2)*
	Details of automatic release processing for the explicit memory block (Output detail information)	EDO	*5.11.5(3)*

#

Prefixes are the alphabetic characters (three characters) enclosed in "[" and "]" that are output at the beginning of the log line.

For details on each event, see the *uCosminexus Application Server Expansion Guide*.

5.11.2 Confirmation method of event log of Explicit Memory Management functionality

This subsection describes the reference method of event log output by the Explicit Memory Management functionality.

(1) Confirmation method of output trigger

Events that trigger the log output are output in [cause:<CAUSE>] format. You can confirm the event triggered by confirming the item.

The following table describes the meaning of character string output to <CAUSE>.

Table 5–30: Meaning of character string output to <CAUSE>

<CAUSE>	Meaning
GC	Copy GC occurrence
Full GC	Full GC occurrence
Reclaim	Explicit memory block explicit release process
Reclaiming	Java heap overflow in Explicit memory block explicit release process
New	Object generation in Explicit memory block
Migrate	Automatic release processing for the explicit memory block
Migrating	Java heap overflow in the automatic release processing for the explicit memory block

However, <CAUSE> is not output in a part of the log.

(2) Checking methods using log prefix

You can use log prefixes such as ENS, EVO to perform checking by filtering the log.

The meaning of prefixes of the Explicit Memory Management functionality is as follows:

- "E" of first character indicates that a prefix is for the log of Explicit Memory Management functionality.
- Second character indicates log output level. N is output for normal level, V is output for verbose level and D is output for debug level.
- The third character indicates whether the memory size of Explicit heap or Java heap will change. The memory size changes if the third character is S. The memory size does not change when third character is O. When O, the names of events occurred are output to log. Also, A indicates the log of the automatic allocation configuration file for the explicit memory management.

For example, when you want to check changes of each heap size, perform filtering by S and when you want to check the occurred event, perform filtering by O.

(3) Signs used in description of output format of event log

The following table describes the signs used in the description of output format.

Table 5–31: Signs used in description of output format

Signs	Usage examples	Meaning
*	X*	Repeat the left side items for minimum zero times. Usage example indicates that X is repeated for minimum zero times.
?	X?	Repeat the left side items for minimum one time. Usage example indicates that X is repeated for minimum once.
{n,m}	X{1,5}	Repeat the left side items for minimum n times and maximum m times. Usage example indicates that X is repeated minimum once and maximum five times.
{	{ABC}*	The range included in { and } is reference unit at left side of *, ?, {n,m}. Usage example indicates that ABC is repeated for minimum zero times.
}		
\	None	This sign indicates the places with linefeed for visibility in the manual. There is no linefeed in the actual output contents.
|	X|Y	This sign indicates either the left side or the right side. Usage example indicates either X or Y.
.	None	Indicates any character.
...	X|Y...	Repeat minimum once from the line that has same indent as this line and that is nearest to forward match of this line, to the line previous to this line. Usage example indicates that either of X or Y is repeated minimum once.

In the description of output format, numeric values are expressed by combining signs as in Table 5-32 and signs in Table 5-33.

The following table describes the signs used to indicate numeric values.

Table 5–32: Signs used to indicate numeric values

Sign	Definition	Meaning
<digit>	0|1|2|3|4|5|6|7|8|9	Indicates 0 to 9.
<hex>	<digit>|a|b|c|d|e|f	Indicates hexadecimals.
<const>	<const><digit>|<digit>	Indicates positive whole number.
<decimal>	<const>.<digit>	Indicates positive integer (Up to 1st Decimal position).
<ptr64>	0x<hex><hex><hex><hex><hex><hex><hex><hex><hex><hex><hex><hex><hex><hex><hex><hex>	Indicates 64 bit pointer value.
<ptr32>	0x<hex><hex><hex><hex><hex><hex><hex><hex>	Indicates 32 bit pointer value.
<ptr>	<ptr64>|<ptr32>	Indicates pointer value.
<letters>	.?	Indicates any character or character string.

Note The signs indicated in Table 5-32 would be the reference unit at left side or right side of *, ? or |. For example, when <digit>|<hex> is mentioned, indicates numeric value from zero to nine or hexadecimal.

5.11.3 Contents output when output level is normal

This subsection describes the contents output for each event when normal is specified in log output levels.

The normal is the log output level specified when there are normal operations.

(1) GC occurrence (Output Explicit heap usage status)

Output Explicit heap usage status when GC occurred.

Output this log when there is no object for migration from Java heap to Explicit heap. <EH_USED_BF> and <EH_USED_AF> would have same value when there is no object to migrate to Explicit heap.

(a) Output trigger

Output trigger is the termination of GC.

(b) Output format

```
[ENS]<ctime>[EH: <EH_USED_BF>-><EH_USED_AF>(<EH_TOTAL>/<EH_MAX>)][E/F/D: <AC
_NUM>/<FL_NUM>/<DA_NUM>][cause:<CAUSE>][CF: <CF_CNT>]
```

(c) Output items

The following table describes each item indicated in *(b) output format*.

Table 5–33: Output items (GC occurrence (Output Explicit heap usage status))

Output items	Output contents	Meaning
<ctime>	<letters>	Indicates occurrence date and time of GC. Output in the format same as extended verbosegc information. Output in millisecond units when the -XX:+HitachiOutputMilliTime option is set.
<EH_USED_BF>	<const>K	Output used size of Explicit heap before execution of GC. The unit is kilo bytes.
<EH_USED_AF>	<const>K	Output used size of Explicit heap after execution of GC. The unit is kilo bytes.
<EH_TOTAL>	<const>K	Output secured memory size of Explicit heap after execution of GC. The unit is kilo bytes.
<EH_MAX>	<const>K	Output Explicit heap maximum size. The unit is kilo bytes.
<AC_NUM>	<const>	Output number of Explicit memory blocks whose sub status is Enable after GC execution.
<FL_NUM>	<const>	Normally 0 is output
<DA_NUM>	<const>	Output number of Explicit memory blocks whose sub status is Disable after GC execution.
<CAUSE>	GC|Full GC	Output type of GC that is triggered. GC indicates copy GC and Full GC indicates Full GC.
<CF_CNT>	<const>	Output the failure count of initialization of Explicit memory blocks from the occurrence of previous GC up to occurrence of current GC.

(d) Example of output

Output example

```
[ENS]<Thu Oct 21 14:55:50 2007>[EH: 150528K->162816K(162816K/1048576K)][E/F/
D: 200/0/0][cause:GC][CF: 0]
```

You can check the following contents in this output example:

- Output trigger is copy GC occurred on October 21, 2007 (Friday) 14:55:50.
- Migration to Explicit heap occurred in GC and used size of Explicit heap changed from 150,528K to 162,816K.
- The secured size of Explicit heap after occurrence of GC is 162,816K. Maximum size of Explicit heap is 1,048,576K.
- After the occurrence of GC, Explicit memory blocks whose sub status is Enable are 200 blocks.

(2) Explicit memory block explicit release process

Output usage status of the Explicit heap or Java heap after termination of the explicit release process of Explicit memory block.

(a) Output trigger

The Explicit release process of Explicit memory block.

The Explicit release of Explicit memory block occurs immediately after GC The following log is output after log described in *(1) GC occurrence (Output Explicit heap usage status)*.

(b) Output format

```
[ENS]<ctime>[EH: <EH_USED_BF>-><EH_USED_AF>(<EH_TOTAL>/<EH_MAX>), <ELAPSED>
secs][E/F/D: <AC_NUM>/<FL_NUM>/<DA_NUM>]\
[DefNew::Eden: <ED_USED_BF>-><ED_USED_AF>(<ED_TOTAL>)][DefNew::Survivor: <SV
_USED_BF>-><SV_USED_AF>(<SV_TOTAL>)]\
[Tenured: <TN_USED_BF>-><TN_USED_AF>(<TN_TOTAL>)][User: <USERCPU> secs][Sys
: <SYSCPU> secs][cause:<CAUSE>]
```

(c) Output items

The following table describes each item indicated in *(b) output format*.

Table 5–34: Output items (Explicit memory block explicit release process)

Output items	Output contents	Meaning
<ctime>	<letters>	Indicates the occurrence date and time of the Explicit memory block explicit release process. Output in the format same as extended verbosegc information. Output in milli-seconds unit when -XX:+HitachiOutputMilliTime option is set.
<EH_USED_BF>	<const>K	Output used size of the Explicit heap before the Explicit memory block explicit release process. The unit is kilo bytes.
<EH_USED_AF>	<const>K	Output used size of the Explicit heap after the Explicit memory block explicit release process. The unit is kilo bytes.

Output items	Output contents	Meaning
<EH_TOTAL>	<const>K	Output secured size of the Explicit heap after the Explicit memory block explicit release process. The unit is kilo bytes.
<EH_MAX>	<const>K	Output Explicit heap maximum size. The unit is kilo bytes.
<ELAPSED>	<time>	Output time required for the Explicit memory block explicit release process. The unit is seconds.
<AC_NUM>	<const>	Output number of Explicit memory blocks whose sub status is `Enable` after execution of the Explicit memory block explicit release process.
<FL_NUM>	<const>	Normally 0 is output
<DA_NUM>	<const>	Output number of Explicit memory blocks whose sub status is `Disable` after execution of the Explicit memory block explicit release process.
<ED_USED_BF>	<const>K	Output used size of the Eden area before executing the Explicit memory block explicit release process. The unit is kilo bytes.
<ED_USED_AF>	<const>K	Output used size of the Eden area after executing the Explicit memory block explicit release process. The unit is kilo bytes.
<ED_TOTAL>	<const>K	Output secured size of the Eden area after executing the Explicit memory block explicit release process. The unit is kilo bytes.
<SV_USED_BF>	<const>K	Output used size of the Survivor area before executing the Explicit memory block explicit release process. The unit is kilo bytes.
<SV_USED_AF>	<const>K	Output used size of the Survivor area after executing the Explicit memory block explicit release process. The unit is kilo bytes.
<SV_TOTAL>	<const>K	Output secured size of the Survivor area after executing the Explicit memory block explicit release process. The unit is kilo bytes.
<TN_USED_BF>	<const>K	Output used size of the Tenured area before executing the Explicit memory block explicit release process. The unit is kilo bytes.
<TN_USED_AF>	<const>K	Output used size of the Tenured area after executing the Explicit memory block explicit release process. The unit is kilo bytes.
<TN_TOTAL>	<const>K	Output secured size of the Tenured area after executing the Explicit memory block explicit release process. The unit is kilo bytes.
<USERCPU>	<time>	Output the user CPU time taken by the explicit release processing of the Explicit memory block. The unit is seconds.
<SYSCPU>	<time>	Output the system CPU time taken by the explicit release processing of the Explicit memory block. The unit is seconds.
<CAUSE>	`Reclaim`	Output as `Reclaim`. `Reclaim` is the log output in the explicit release processing of the Explicit memory block.

(d) Example of output

Output example:

```
[ENS]<Tue Jul 24 01:23:51 2007>[EH: 150528K->149528K(162816K/1048576K), 0.11
29602 secs][E/F/D: 523/0/0]\
[DefNew::Eden: 0K->0K(243600K)][DefNew::Survivor: 0K->0K(17400K)][Tenured: 1
03400K->103400K(556800K)]\
[User: 0.0900000 secs][Sys: 0.0200000 secs][cause:Reclaim]
```

You can check the following contents in this output example:

- Output trigger is the Explicit memory block explicit release process occurred on July 24, 2007 (Tuesday) 1:23:51.
- Explicit heap used size reduced from 150,528K to 149,528K by the explicit release process of the Explicit memory block.
- Secured size of Explicit heap after Explicit memory block explicit release process is 162,816K. Maximum size of Explicit heap is 1,048,576K.
- The time for the explicit release process of Explicit memory block is 0.1129602 seconds.
- There are 523 Explicit memory blocks whose sub status is `Enable` after Explicit memory block explicit release process.
- There are no changes in each area of Java heap by the Explicit memory block explicit release process. In other words, there is no object moved to Java heap.
- The explicit release processing of the Explicit memory block took 0.0900000 seconds of the user CPU time, and 0.0200000 seconds of the system CPU time.

(3) Java heap overflow in Explicit memory block explicit release process

The object moves to Java heap in the explicit release process of Explicit memory block and the status is output when Java heap overflows. Output the usage status of Explicit heap and Java heap when Java heap overflows.

(a) Output trigger

The object moves from Explicit heap to Java heap in the explicit release process of the Explicit memory block and output is triggered when Java heap overflows.

(b) Output format

```
[ENS]<ctime>[EH: <EH_USED_BF>-><EH_USED_AF>(<EM_TOTAL>/<EH_MAX>), <ELAPSED>
secs][E/F/D: <AC_NUM>/<FL_NUM>/<DA_NUM>]\
[DefNew::Eden: <ED_USED_BF>-><ED_USED_AF>(<ED_TOTAL>)][DefNew::Survivor: <SV
_USED_BF>-><SV_USED_AF>(<SV_TOTAL>)]\
[Tenured: <TN_USED_BF>-><TN_USED_AF>(<TN_TOTAL>)][User: <USERCPU> secs][Sys
: <SYSCPU> secs][cause:<CAUSE>]
```

(c) Output items

The following table describes each item indicated in *(b) output trigger*.

Table 5–35: Output items (Java heap overflow in the explicit release process of the Explicit memory block)

Output items	Output contents	Meaning
`<ctime>`	`<letters>`	Indicates the occurrence date and time of the Explicit memory block explicit release process. Output in the format same as extended `verbosegc` information. Output in milli-seconds unit when `-XX:+HitachiOutputMilliTime` option is set.
`<EH_USED_BF>`	`<const>K`	Output used size of the Explicit heap before the Explicit memory block explicit release process. The unit is kilo bytes.
`<EH_USED_AF>`	`<const>K`	Output used size of the Explicit heap after Java heap overflow. When Java heap overflows, the explicit release process of Explicit memory block is not executed and hence the value is always same as `<EH_USED_BF>`. The unit is kilo bytes.

Output items	Output contents	Meaning
<EH_TOTAL>	<const>K	Output secured size of the Explicit heap after Java heap overflow. The unit is kilo bytes.
<EH_MAX>	<const>K	Output Explicit heap maximum size. The unit is kilo bytes.
<ELAPSED>	<time>	Output time from start of the Explicit memory block explicit release process to Java heap overflow. The unit is seconds.
<AC_NUM>	<const>	Output number of the Explicit memory blocks whose sub status is Enable after Java heap overflow.
<FL_NUM>	<const>	Normally 0 is output
<DA_NUM>	<const>	Output number of the Explicit memory blocks whose sub status is Disable after Java heap overflow.
<ED_USED_BF>	<const>K	Output used size of the Eden area before executing the Explicit memory block explicit release process. The unit is kilo bytes.
<ED_USED_AF>	<const>K	Output used size of the Eden area after Java heap overflow. The unit is kilo bytes.
<ED_TOTAL>	<const>K	Output secured size of the Eden area after Java heap overflow. The unit is kilo bytes.
<SV_USED_BF>	<const>K	Output used size of the Survivor area before executing the Explicit memory block explicit release process. The unit is kilo bytes.
<SV_USED_AF>	<const>K	Output used size of the Survivor area after Java heap overflow. The unit is kilo bytes.
<SV_TOTAL>	<const>K	Output secured size of the Survivor area after Java heap overflow. The unit is kilo bytes.
<TN_USED_BF>	<const>K	Output used size of the Tenured area before executing the Explicit memory block explicit release process. The unit is kilo bytes.
<TN_USED_AF>	<const>K	Output used size of the Tenured area after Java heap overflow. The unit is kilo bytes.
<TN_TOTAL>	<const>K	Output secured size of the Tenured area after Java heap overflow. The unit is kilo bytes.
<USERCPU>	<time>	Output the user CPU time taken from the start of the explicit release processing of the Explicit memory block until Java heap overflows. The unit is seconds.
<SYSCPU>	<time>	Output the system CPU time taken from the start of the explicit release processing of the Explicit memory block until Java heap overflows. The unit is seconds.
<CAUSE>	Reclaiming	Output as "Reclaiming". Reclaiming is the log output by Java heap overflow in the release process of Explicit memory block.

(d) Example of output

Output example:

```
[ENS]<Tue Jul 24 01:23:51 2007>[EH: 706728K->706728K(706728K/1048576K), 0.11
29602 secs][E/F/D: 523/0/0]\
[DefNew::Eden: 0K->243600K(243600K)][DefNew::Survivor: 0K->17400K(17400K)][T
enured: 278000K->556800K(556800K)]\
[User: 0.0900000 secs][Sys: 0.0200000 secs][cause:Reclaiming]
[ENS]<Tue Jul 24 01:23:51 2007>[EH: 706728K->706728K(706728K/1048576K)][E/F/
```

```
D: 523/0/0][cause:Full GC][CF: 0]
[ENS]<Tue Jul 24 01:23:53 2007>[EH: 706728K->148528K(148528K/1048576K), 0.01
23405 secs][E/F/D: 521/0/0]\
[DefNew::Eden: 0K->0K(243600K)][DefNew::Survivor: 0K->0K(17400K)][Tenured: 5
51800K->552800K(556800K)]\
[User: 0.0090000 secs][Sys: 0.0020000 secs][cause:Reclaim]
```

You can confirm the following contents in this output example:

- Output trigger is the Java heap overflow in the Explicit memory block explicit release process occurred on July 24, 2007(Tuesday) 1:23:51.
- The time from starting the explicit release process of Explicit memory block to Java heap overflow is 0.1129602 seconds.
- There are 523 Explicit memory blocks whose sub status is Enable after Java heap overflow.
- 5,398,00K moved to Java heap in the explicit release process of Explicit memory block causing Java heap overflow.
- The process from the start of the explicit release processing of the Explicit memory block until the overflowing of the Java heap took 0.0900000 seconds of the user CPU time, and 0.0200000 seconds of the system CPU time.

In the output contents of output example, third line `ENS` onwards is the log output by the explicit release process of Explicit memory block. Always output the log of Explicit memory block release process after log output by Java heap overflow. In this example, the following contents are output:

- Restarted the explicit release process of Explicit memory block on July 24, 2007 1:23:53 and output this log.
- The used size of Explicit heap reduced from 706,728K to 148,528K in the explicit release process of Explicit memory block that was restarted.
- The secured size of Explicit heap after executing the explicit release process of restarted Explicit memory block is 148,528K. Maximum size of Explicit heap is 1,048,576K.
- The time for the explicit release process of restarted Explicit memory block is 0.0123405 seconds.
- There are 521 Explicit memory blocks whose sub status is `Enable` after the restarted Explicit memory block explicit release process.
- The usage size of Tenured area of Java heap increased from 551,800K to 552,800K as a result of the explicit release process of restarted Explicit memory block.
- The explicit release processing of the restarted Explicit memory block took 0.0090000 seconds of the user CPU time and 0.0020000 seconds of the system CPU time.

(4) Automatic release processing of the explicit memory block

The usage status of the explicit memory and explicit memory block is output from the automatic release - automatic reserve or automatic release - explicit reserve of the explicit memory block up to the automatic release processing of the explicit memory block.

(a) Output trigger

The log is output when the automatic release - automatic reserve or automatic release - explicit reserve of the explicit memory block and automatic release processing of the Explicit memory block occur.

(b) Output format

```
[ENS]<ctime>[EH: <EH_USED_BF>-><EH_USED_AF>(<EH_TOTAL>/<EH_MAX>), <ELAPSED>
secs][E/F/D: <AC_NUM>/<FL_NUM>/<DA_NUM>]\
```

```
[DefNew::Eden: <ED_USED_BF>-><ED_USED_AF>(<ED_TOTAL>)]
[DefNew::Survivor: <SV_USED_BF>-><SV_USED_AF>(<SV_TOTAL>)]\
[Tenured: <TN_USED_BF>-><TN_USED_AF>(<TN_TOTAL>)][target:<EH_MIG_TRG>/<EH_MI
G_DED>/<EH_MIG_LIV>]\
[User: <USERCPU> secs][Sys: <SYSCPU> secs][cause:<CAUSE>]
```

(c) Output items

The following table describes the items specified in *(b) Output format*.

Table 5–36: Output items (Automatic release processing of the explicit memory block)

Output items	Output contents	Meaning
<ctime>	<letters>	Indicates the occurrence date and time of automatic release - automatic reserve of the explicit memory block. Output with the format same as the extended verboseGC functionality. Output in milliseconds, when the HitachiOutputMilliTime option is set up.
<EH_USED_BF>	<const>K	Outputs the used size of the explicit memory before the automatic release processing of the explicit memory block. The unit is kilobytes.
<EH_USED_AF>	<const>K	Outputs the used size of the explicit memory after the automatic release processing of the explicit memory block. The unit is kilobytes.
<EH_TOTAL>	<const>K	Outputs the secured size of the explicit memory after the automatic release processing of the Explicit memory block. The unit is kilobytes.
<EH_MAX>	<const>K	Outputs the maximum explicit memory size. The unit is kilobytes.
<ELAPSED>	<time>	Outputs the time from the start of the automatic release-automatic reserve process of the explicit memory block up to the end of the automatic release processing. The unit is seconds.
<AC_NUM>	<const>	Outputs the number of valid explicit memory blocks that have the Enable sub state, after the automatic release processing of the explicit memory block.
<FL_NUM>	<const>	Always outputs 0.
<DA_NUM>	<const>	Outputs the number of valid explicit memory blocks that have the Disable sub state, after the automatic release processing of the explicit memory block.
<ED_USED_BF>	<const>K	Outputs the used size of the Eden area before the automatic release processing of the explicit memory block. The unit is kilobytes.
<ED_USED_AF>	<const>K	Outputs the used size of the Eden area after the automatic release processing of the explicit memory block. The unit is kilobytes.
<ED_TOTAL>	<const>K	Outputs the secured size of the Eden area after the automatic release processing of the explicit memory block. The unit is kilobytes.
<SV_USED_BF>	<const>K	Outputs the used size of the Survivor area before the automatic release processing of the explicit memory block. The unit is kilobytes.
<SV_USED_AF>	<const>K	Outputs the used size of the Survivor area after the automatic release processing of the explicit memory block. The unit is kilobytes.
<SV_TOTAL>	<const>K	Outputs the secured size of the Survivor area after the automatic release processing of the explicit memory block. The unit is kilobytes.
<TN_USED_BF>	<const>K	Outputs the used size of the Tenured area before the automatic release processing of the Explicit memory block. The unit is kilobytes.
<TN_USED_AF>	<const>K	Outputs the used size of the Tenured area after the automatic release processing of the explicit memory block. The unit is kilobytes.
<TN_TOTAL>	<const>K	Outputs the secured size of the Tenured area after the automatic release processing of the explicit memory block. The unit is kilobytes.

Output items	Output contents	Meaning
<EH_MIG_TRG>	<const>K	Outputs the used size of the explicit memory for which the automatic release processing of the explicit memory block was executed. The unit is kilobytes.
<EH_MIG_DED>	<const>K	Outputs the used size of the explicit memory that decreased due to the execution of the automatic release processing of the explicit memory block. The unit is kilobytes.
<EH_MIG_LIV>	<const>K	Outputs the used size of the explicit memory that did not decrease in spite of the execution of the automatic release processing of the explicit memory block. The unit is kilobytes.
<USERCPU>	<time>	Outputs the user CPU time taken from the start of the automatic release - automatic reserve processing of the Explicit memory block until the end of the automatic release processing. The unit is seconds.
<SYSCPU>	<time>	Outputs the system CPU time from the start of the automatic release - automatic reserve processing of the Explicit memory block until the end of the automatic release processing. The unit is seconds.
<CAUSE>	Migrate	Outputs Migrate. Indicates that the log was output due to the automatic release processing of the explicit memory block.

(d) Example of output

The following is an output example:

```
[ENS]<Tue Jul 14 02:31:22 2009>[EH: 256512K->256128K(256256K/1048576K), 0.11
24626 secs][E/F/D: 423/0/0]\
[DefNew::Eden: 0K->0K(243600K)][DefNew::Survivor: 0K->0K(17400K)][Tenured: 1
03400K->103400K(556800K)][target:584K/384K/200K]\
[User: 0.0900000 secs][Sys: 0.0200000 secs][cause:Migrate]
```

You can check the following details in this output example:

- The output trigger is the automatic release processing of the explicit memory block in the GC that occurred on July 14, 2009 (Tuesday) at 2:31:22.
- The used size of explicit memory changed from 256,512K to 256,128K due to the automatic release processing.
- The secured size of explicit memory after the automatic release processing is 256,256K and the maximum size is 1,048,576K.
- The time taken for the automatic release processing was 0.1124626 seconds.
- There are 423 explicit memory blocks whose sub-state is Enable after the automatic release processing.
- Among the used size of explicit memory, 584K were automatically released using the automatic release processing. The size that decreased was 384K and the size that did not decrease was 200K.
- There are no changes in the Java heap areas due to the automatic release processing for the explicit memory block.
- The automatic release processing took 0.0900000 seconds of the user CPU time, and 0.0200000 seconds of the system CPU time.

(5) Java heap overflow in the automatic release processing of the explicit memory block

During the automatic release processing of the explicit memory block, objects were moved to the Java heap. In this case, the usage status of the explicit memory and the Java heap is output when the Java heap overflows.

The overflow of Java heap indicates that while moving the objects to the Java heap, there was no free space in the Java heap. For details, *see 7. Suppression of Full GC by Using the Explicit Memory Management Functionality* in the *uCosminexus Application Server Expansion Guide*.

(a) Output trigger

If there is a shortage of free space in the explicit memory area during the automatic release processing of the explicit memory block, the objects are moved to the Java heap. A log is output, when the Java heap overflows.

(b) Output format

```
[ENS]<ctime>[EH: <EH_USED_BF>-><EH_USED_AF>(<EH_TOTAL>/<EH_MAX>), <ELAPSED>
secs][E/F/D: <AC_NUM>/<FL_NUM>/<DA_NUM>]\
[DefNew::Eden: <ED_USED_BF>-><ED_USED_AF>(<ED_TOTAL>)]
[DefNew::Survivor: <SV_USED_BF>-><SV_USED_AF>(<SV_TOTAL>)]\
[Tenured: <TN_USED_BF>-><TN_USED_AF>(<TN_TOTAL>)][target:<EH_MIG_TRG>/<EH_MI
G_DED>/<EH_MIG_LIV>]\
[User: <USERCPU> secs][Sys: <SYSCPU> secs][cause:<CAUSE>]
```

(c) Output items

The following table describes the items specified in *(b) Output format*.

Table 5–37: Output items (Java heap overflow in the automatic release processing for the explicit memory block)

Output items	Output contents	Meaning
<ctime>	<letters>	Indicates the occurrence date and time of the automatic release processing of the explicit memory block. Output in the format same as would output for the extended `verboseGC` functionality. Output in milliseconds when the `HitachiOutputMilliTime` option is set up.
<EH_USED_BF>	<const>K	Outputs the used size of the explicit memory before the automatic release processing of the explicit memory block. The unit is kilobytes.
<EH_USED_AF>	<const>K	Outputs the used size of the explicit memory after the Java heap overflows. The unit is kilobytes.
<EH_TOTAL>	<const>K	Outputs the secured size of explicit memory after the Java heap overflows. The unit is kilobytes.
<EH_MAX>	<const>K	Outputs the maximum explicit memory size. The unit is kilobytes.
<ELAPSED>	<time>	Outputs the time from the start of the automatic release processing of the explicit memory block to the overflow of the Java heap. The unit is seconds.
<AC_NUM>	<const>	Outputs the number of explicit memory blocks that have the `Enable` sub state after the automatic release processing of the explicit memory block.
<FL_NUM>	<const>	Always outputs 0.
<DA_NUM>	<const>	Outputs the number of explicit memory blocks that have the `Disable` sub state after the Java heap overflows.
<ED_USED_BF>	<const>K	Outputs the used size of the Eden area before the automatic release processing of the explicit memory block. The unit is kilobytes.

Output items	Output contents	Meaning
<ED_USED_AF>	<const>K	Outputs the used size of the Eden area after the Java heap overflows. The unit is kilobytes.
<ED_TOTAL>	<const>K	Outputs the secured size of the Eden area after the Java heap overflows. The unit is kilobytes.
<SV_USED_BF>	<const>K	Outputs the used size of the Survivor area before the automatic release processing of the explicit memory block. The unit is kilobytes.
<SV_USED_AF>	<const>K	Outputs the used size of the Survivor area after the Java heap overflows. The unit is kilobytes.
<SV_TOTAL>	<const>K	Outputs the secured size of the Survivor area after the Java heap overflows. The unit is kilobytes.
<TN_USED_BF>	<const>K	Outputs the used size of the Tenured area before the automatic release processing of the explicit memory block. The unit is kilobytes.
<TN_USED_AF>	<const>K	Outputs the used size of the Tenured area after the Java heap overflows. The unit is kilobytes.
<TN_TOTAL>	<const>K	Outputs the secured size of the Tenured area after the Java heap overflows. The unit is kilobytes.
<EH_MIG_TRG>	<const>K	Outputs the used size of the explicit memory for which the automatic release processing of the explicit memory block was executed. The unit is kilobytes.
<EH_MIG_DED>	<const>K	Outputs the used size of explicit memory that has decreased due to the execution of the automatic release processing of the explicit memory block, until the Java heap overflowed. The unit is kilobytes. Always outputs OK.
<EH_MIG_LIV>	<const>K	Outputs the used size of the explicit memory that did not decrease due to the execution of the automatic release processing of the explicit memory block, until the Java heap overflowed. The unit is kilobytes. Does not include the size of the objects that caused the Java heap overflow.
<USERCPU>	<time>	Outputs the user CPU time taken from the start of the automatic release processing of the Explicit memory block until the overflowing of the Java heap. The unit is seconds.
<SYSCPU>	<time>	Outputs the system CPU time taken from the start of the automatic release processing of the Explicit memory block until the overflowing of the Java heap. The unit is seconds.
<CAUSE>	Migrating	Outputs Migrating. Indicates that the log was output due to the Java heap overflow in the automatic release processing of the explicit memory block.

(d) Example of output

An output example is as follows:

```
[ENS]<Tue Jul 14 02:31:22 2009>[EH: 706728K->706728K(706728K/706728K), 0.112
9602 secs][E/F/D: 522/0/1]\
[DefNew::Eden: 0K->243600K(243600K)][DefNew::Survivor: 0K->17400K(17400K)][T
enured: 278000K->556800K(556800K)]\
[target:372000K/0K/339800K] [User: 0.0900000 secs][Sys: 0.0200000 secs][caus
e:Migrating]
```

You can check the following details in this output example:

- The output trigger is the Java heap overflow in the automatic release processing for the explicit memory block in the GC that occurred on July 14, 2009 (Tuesday) at 2:31:22.
- The used size of explicit memory changed from 706,728K due to the automatic release processing.
- The secured size of explicit memory after the automatic release processing is 706,728K and the maximum size is 706,728K.
- The time taken for the automatic release processing was 0.1129602 seconds.
- There are 522 explicit memory blocks that have `Enable` sub state after the automatic release processing. There is one explicit memory block that has `Disable` sub state.
- Among the used size of explicit memory, 372,000K were automatically released using the automatic release processing. The size that did not decrease until Java heap overflow was 339,800K.
- Due to Java heap overflow during the automatic release processing, the used size of Java heap reached the upper limit for each area.
- The process from the start of the automatic release processing of the Explicit memory block until the overflowing of the Java heap took 0.0900000 seconds of the user CPU time, and 0.0200000 seconds of the system CPU time.

(6) Error in opening the automatic allocation configuration file for Explicit Memory Management

An error message is output, when an attempt to open or read the automatic allocation configuration file for Explicit Memory Management fails.

(a) Output trigger

The error message is output when an attempt to open or read the automatic allocation configuration file for Explicit Memory Management fails. For example, when the file does not exist, the user does not have the permission to read the file, or an unexpected IO error occurs while reading the file.

(b) Output format

```
[ENA]<ctime> failed to open file. [file=<FILENAME>]
```

(c) Output items

The following table describes the items specified in *(b) Output format*.

Table 5–38: Output items (Error in opening the automatic allocation configuration file for Explicit Memory Management)

Output items	Output contents	Meaning
`<ctime>`	`<letters>`	Indicates the occurrence time, when an attempt to open the automatic allocation configuration file using Explicit Memory Management has failed. Output with the format same as would output for the extended `verboseGC` functionality. Output in milliseconds, when the `HitachiOutputMilliTime` option is set up.
`<FILENAME>`	`<letters>`	Outputs the name of the automatic allocation configuration file that has failed to open (does not include the directory name).

(d) Example of output

An output example is as follows:

```
[ENA]<Tue Jul 24 01:23:51 2007> failed to open file. [file=usrexmem.cfg]
```

You can check the following details in this output example:

- An attempt to open the automatic allocation configuration file for Explicit Memory Management failed on July 24, 2007 (Tuesday) at 1:23:51.

(7) Error in parsing the automatic allocation configuration file for Explicit Memory Management

An error message is output, when a line that failed in the parsing of the automatic allocation configuration file for Explicit Memory Management exists.

(a) Output trigger

The error message is output when a line that failed in the parsing of the automatic allocation configuration file for Explicit Memory Management exists. If there are coding format errors in multiple lines of the file, the log is output several times.

(b) Output format

```
[ENA]<ctime> parsed error line. [file=<FILENAME> line=<LINENO>]
```

(c) Output items

The following table describes the items specified in *(b) Output format*.

Table 5–39: Output items (Error in parsing the automatic allocation configuration file for Explicit Memory Management)

Output items	Output contents	Meaning
<ctime>	<letters>	Indicates the time when an attempt to parse the automatic allocation configuration file using the Explicit Memory Management functionality has failed. Output with the format same as would output for the extended verboseGC functionality. Output in milliseconds, when the HitachiOutputMilliTime option is set up.
<FILENAME>	<letters>	Outputs the name of the automatic allocation configuration file for which file parsing failed (does not include the directory name).
<LINENO>	<const>	Outputs the line number where the parsing failed.

(d) Example of output

An output example is as follows:

```
[ENA]<Tue Jul 24 01:23:51 2007> parsed error line. [file=usrexmem.cfg line=2
5]
```

You can check the following details in this example of output:

- The parsing of the automatic allocation configuration file for the Explicit Memory Management functionality failed in the 25th line on July 24, 2007 (Tuesday) at 1:23:51.

(8) Automatic allocation error in Explicit Memory Management

An error message is output, when a class specified by the Explicit Memory Management functionality fails to allocate automatically to Explicit Memory Management.

(a) Output trigger

The error message is output, when a class specified by the Explicit Memory Management functionality fails to allocate automatically to Explicit Memory Management.

(b) Output format

```
[ENA]<ctime> creation <CLASS_LIST> class's object in explicit memory is fail
ed. [target=<CLASS_METHOD> \
detail=<MESSAGE>]
```

(c) Output items

The following table describes the items specified in *(b) Output format*.

Table 5–40: Output items (Automatic allocation error in Explicit Memory Management)

Output items	Output contents	Meaning
<ctime>	<letters>	Indicates the occurrence time when an attempt of the automatic allocation using Explicit Memory Management has failed. Output with the format same as would output for the extended verboseGC function. Output in milliseconds, when the HitachiOutputMilliTime option is set.
<CLASS_LIST>	<letters>	Outputs the list of fully qualified class names of the objects for which an attempt of the automatic allocation using Explicit Memory Management has failed. The list might be blank.
<CLASS_METHOD>	<letters>	Outputs the fully qualified names of the classes for which an attempt of the automatic allocation using Explicit Memory Management has failed. The method names indicating more detailed failure locations might also be output.
<MESSAGE>	<letters>	Outputs a detailed message indicating the cause of the failure during the automatic allocation using Explicit Memory Management.

(d) Example of output

An output example is as follows:

```
[ENA]<Tue Jul 24 01:23:51 2007> creation java.util.HashMap, java.util.Linked
List \
class's object in explicit memory is failed. [target=com.sample.MainClass.ma
in \
detail=Invalid class file format. (max_stack = 65536, max = 65535, min = 0)]
```

You can check the following details in this output example:

- With the Explicit Memory Management functionality, an attempt to automatically allocate the class `jp.co.sample.Main` on the explicit memory management using has failed on July 24, 2007 (Tuesday) at 1:23:51.

(9) Error in opening the configuration file of the functionality for specifying the classes to be excluded from the Explicit Memory Management functionality

When an attempt to open and read the configuration file of the functionality for specifying the classes to be excluded from the Explicit Memory Management functionality fails, an error message is output.

(a) Output trigger

The error message is output when an attempt to open and read the configuration file of the functionality for specifying the classes to be excluded from the Explicit Memory Management functionality fails. For example, this includes cases such as when the file does not exist, the user does not have the permission to read the file, and an unexpected IO error occurs while the file is being read.

(b) Output format

```
[ENO]<ctime> failed to open file. [<TYPE>] [file=<FILENAME>]
```

(c) Output items

The following table describes the items specified in *(b) Output format*.

Table 5–41: Output items (Error in opening the configuration file of the functionality for specifying the classes to be excluded from the Explicit Memory Management functionality)

Output items	Output contents	Meaning
`<ctime>`	`<letters>`	Specifies the time at which the configuration file failed to open. The time is output in the same format as the time output by the extended VerboseGC functionality. If the `HitachiOutputMilliTime` option is set up, the time is output in milliseconds.
`<TYPE>`	`SYS|USR|DEF`	Outputs the type of the configuration file that could not be opened and read. `SYS` indicates the configuration file provided with the system, `USR` indicates the configuration file with the file path specified in the JavaVM invocation option, and `DEF` indicates the configuration file existing in the default file path of the JavaVM invocation option.
`<FILENAME>`	`<letters>`	Outputs the name of the configuration file that could not be opened (does not include the directory name).

(d) Example of output

An example of output is as follows:

```
[ENO]<Fri Aug 10 17:41:51 2012> failed to open file. [USR] [file=javamove.cf
g]
```

You can check the following details in this example of output:

- The configuration file with the file path specified in the JavaVM invocation option failed to open on August 10, 2012 (Friday) at 17:41:51.

(10) Error in parsing the configuration file of the functionality for specifying the classes to be excluded from the Explicit Memory Management functionality

If there is a line in which the parsing of the configuration file of the functionality for specifying the classes to be excluded from the Explicit Memory Management functionality has failed, an error message is output.

(a) Output trigger

The error message is output if there is a line in which the parsing of the configuration file of the functionality for specifying the classes to be excluded from the Explicit Memory Management functionality has failed. If a coding format error exists in multiple lines of a file, the log is output multiple times.

(b) Output format

```
[ENO]<ctime> parsed error line. [TYPE] [file=<FILENAME> line=<LINENO>]
```

(c) Output items

The following table describes the items specified in *(b) Output format*.

Table 5–42: Output items (Parsing error in the automatic allocation configuration file for Explicit Memory Management)

Output items	Output contents	Meaning
<ctime>	<letters>	Specifies the time at which the parsing of the configuration file of the functionality for specifying the classes to be excluded from the Explicit Memory Management functionality failed. The time is output in the same format as the time output by the extended verboseGC functionality. If the HitachiOutputMilliTime option is set up, the time is output in milliseconds.
<TYPE>	SYS|USR|DEF	Outputs the type of configuration file for which the file parsing failed. SYS indicates the configuration file provided with the system, USR indicates the configuration file with the file path specified in the JavaVM invocation option, and DEF indicates the configuration file existing in the default file path of the JavaVM invocation option.
<FILENAME>	<letters>	Outputs the name of the configuration file for which the file parsing failed (does not include the directory name).
<LINENO>	<const>	Outputs the line number for which the parsing has failed.

(d) Example of output

An example of output is as follows:

```
[ENO]<Fri Aug 10 17:41:51 2012> parsed error line. [USR] [file=javamove.cfg
line=25]
```

You can check the following details in this example of output:

- The parsing of the configuration file with the file path specified in the JavaVM invocation option failed in the 25th line on August 10, 2012 (Friday) at 17:41:51.

5.11.4 Contents output when output level is verbose

This subsection describes the contents output for each event when you specify `verbose` in the log output level. `Verbose` is a log output level to output detail information required for the error analysis.

Supplement

In `verbose`, besides contents output for `normal`, detail log is output for contents that are not output in `normal`. The log output would cost some overheads and there might be the degradation in throughput, if verbose is specified for normal operations.

(1) Initialization of Explicit memory block

When you initialize the Explicit memory block newly, the name, ID and type of initialized Explicit memory block is output.

(a) Output trigger

Initialization of Explicit memory block.

(b) Output format

```
[EVO]<ctime>[Created]["<EM_NAME>" eid=<EID>(<EM_PTR>)/<EM_TYPE>]
```

(c) Output items

The following table describes each item indicated in *(b) output trigger*.

Table 5–43: Output items (Initialization of Explicit memory block)

Output items	Output contents	Meaning
<ctime>	<letters>	Indicates the date and time on which Explicit memory block was initialized. Output in the format same as extended `verbosegc` information. Output in milli-seconds unit when `-XX:+HitachiOutputMilliTime` option is set.
<EM_NAME>	<letters>	Output name of initialized Explicit memory block. The output contents are uncertain when multi byte characters are included in name of Explicit memory block (Usually garbled and output).
<EID>	<const>	Output ID of initialized Explicit memory block.
<EM_PTR>	<ptr>	Output value that indicates internal status of Explicit memory block.
<EM_TYPE>	R|B	Output type of initialized Explicit memory block. R indicates Explicit memory block used internally in Application Server. B indicates the type of the Explicit memory block in JavaVM.

(d) Example of output

Output example:

```
[EVO]<Tue Jul 24 01:23:51 2007>[Created]["BasicExplicitMemory-2" eid=2(0x123
4568)/B]
```

You can confirm the following contents in this output example:

- Output trigger is the initialization of Explicit memory block executed on July 24. 2007 (Tuesday) 1:23:51.
- The name of initialized Explicit memory block is `BasicExplicitMemory-2`.

(2) Initialization failure of Explicit memory block

Output when an attempt to initialize Explicit memory block failed while initializing Explicit memory block. The cause for failure is that the Explicit memory blocks have reached the maximum count. Output the number of Explicit memory blocks and stack trace of Java program that failed in initialization.

This log is output on multiple lines. This log is output asynchronously with the execution of the Java program. Hence, there are cases when some other log is output within the lines of this log. However, other logs are not output within one line.

(a) Output trigger

The output is triggered when Explicit memory block reaches the upper limit of Explicit memory blocks and when an attempt to initialize the Explicit memory block fails.

(b) Output format

```
[EVO]<ctime>[Creation failed][EH: <EH_USED>(<EH_GARB>)/<EH_TOTAL>/<EH_MAX>][
E/F/D: <AC_NUM>/<FL_NUM>/<DA_NUM>][Thread: <TH_PTR>]
[EVO][Thread: <TH_PTR>] at <FRAME><SOURCE>
...
```

(c) Output items

The following table describes each item indicated in *(b) output trigger*.

Table 5–44: Output contents (Initialization failure of Explicit memory block)

Output items	Output contents	Meaning
`<ctime>`	`<letters>`	Indicates failure date and time of initialization of Explicit memory block. Output in the format same as extended `verbosegc` information. Output in milli-seconds unit when `-XX:+HitachiOutputMilliTime` option is set.
`<EH_USED>`	`<const>K`	Output used size of Explicit heap when failed in initialization of Explicit memory block. The unit is kilo bytes.
`<EH_GARB>`	`<const>K`	The internal status of Explicit heap is output.
`<EH_TOTAL>`	`<const>K`	Output secured size of Explicit heap when failed in initialization of Explicit memory block. The unit is kilo bytes.
`<EH_MAX>`	`<const>K`	Output Explicit heap maximum size. The unit is kilo bytes.

Output items	Output contents	Meaning
<AC_NUM>	<const>	Output number of Explicit memory blocks whose sub status is Enable when failed in initialization of Explicit memory block.
<FL_NUM>	<const>	Normally 0 is output
<DA_NUM>	<const>	Output number of Explicit memory blocks whose sub status is Disable when failed in initialization of Explicit memory block.
<TH_PTR>	<ptr>	Output thread ID of threads failed in initialization of Explicit memory block. Thread ID is same as tid output to thread dump.
<FRAME>	<letters>.<letters>	Output one frame in stack trace when failed in initialization of Explicit memory block. Output all class names and method names by delimiting them by a ".".
<SOURCE>	(<letters>:<const>)| (Native Method)| (Unknown Source)	Output the source file names where methods output to <FRAME> are described and the line numbers match the stack trace. Output the file name and line numbers by delimiting them by a ":". Output as "(Native Method)" for native method. Output as "(Unknown Source)" when you cannot acquire source file name.

(d) Example of output

Output example.

```
[EVO]<Tue Jul 24 01:23:51 2007>[Creation failed][EH: 12000K(0K)/15000K/30000
K][E/F/D: 65535/0/0][Thread: 0x00035a60]
[EVO][Thread: 0x00035a60] at ExplicitMemory.registerExplicitMemory(Native Me
thod)
[EVO][Thread: 0x00035a60] at BasicExplicitMemory.<init>(Unknown Source)
[EVO][Thread: 0x00035a60] at AllocTest.test(AllocTest.java:64)
[EVO][Thread: 0x00035a60] at java.lang.Thread.run(Thread.java:2312)
```

You can confirm the following contents in this output example:

- Output trigger is the failure in initialization of Explicit memory block occurred on July 24, 2007 (Tuesday) 1:23:51. The maximum number of Explicit memory blocks is 65535 that already exist and hence failed in initialization of new Explicit memory block.
- On the 64th line of AllocTest.java, try initializing Explicit memory block by executing constructor of BasicExplicitMemory class.

(3) Disable of sub status of Explicit memory block

Output the information of Explicit memory block in which the usage status and sub status are changed to Disable, when the sub status of Explicit memory block is changed to Disable. This log is output on multiple lines and output asynchronously with the execution of Java program. Hence, there are cases where some other log is output within the lines of this log. However, other logs are not output within one line.

(a) Output trigger

Output is triggered when sub status of Explicit memory block is Disable and when you cannot map the object to respective Explicit memory block.

(b) Output format

```
[EVO]<ctime>[Alloc failed(Disable)][EH: <EH_USED>(<EH_GARB>)/<EH_TOTAL>/<EH_
MAX>][E/F/D: <AC_NUM>/<FL_NUM>/<DA_NUM>][cause:<CAUSE>]\
["<EM_NAME>" eid=<EID>/<EM_TYPE>: <EM_USED>(<EM_GARB>)/<EM_TOTAL>][Thread: <
TH_PTR>]
[EVO][Thread: <TH_PTR>] at <FRAME><SOURCE>
...
```

#

The Underlined part is output only for New.

(c) Output items

The following table describes each item indicated in *(b) output trigger*.

Table 5–45: Output items (Disable sub status of Explicit memory block)

Output items	Output contents	Meaning
<ctime>	<letters>	Indicates the date and time when sub status of Explicit memory block with <EID> is changed to Disable. Output in the format same as extended verbosegc information. Output in milli-seconds unit when -XX:+HitachiOutputMilliTime option is set.
<EH_USED>	<const>K	Output used size of Explicit heap when sub status of Explicit memory block with <EID> is Disable. The unit is kilo bytes.
<EH_GARB>	<const>K	Output value indicating internal status of Explicit heap.
<EH_TOTAL>	<const>K	Output secured size of Explicit heap when sub status of Explicit memory block with <EID> is Disable. The unit is kilo bytes.
<EH_MAX>	<const>K	Output Explicit heap maximum size. The unit is kilo bytes.
<AC_NUM>	<const>	Output number of Explicit memory blocks that are Enable after changing the sub status of Explicit memory block with <EID> to Disable.
<FL_NUM>	<const>	Normally 0 is output
<DA_NUM>	<const>	Output number of Explicit memory blocks that whose sub status is Disable after changing the sub status of Explicit memory block with <EID> to Disable.
<CAUSE>	New|GC|Full GC	Output the process that cause Disable of sub status. The output contents and measures for the cause are as follows: • "New" The cause is direct generation of object in Explicit heap by newInstance(). • "GC" The cause is moving the object to Explicit heap by copy GC. • "Full GC" The cause is moving the object to Explicit heap by Full GC.
<EM_NAME>	<letters>	Output name of Explicit memory block whose sub status is Disable. The output contents are uncertain when multi byte characters are included in name of Explicit memory block (Usually garbled and output).

Output items	Output contents	Meaning
<EID>	<const>	Output ID of Explicit memory block whose sub status is Disable.
<EM_TYPE>	R|B|A	Output type of Explicit memory block whose sub status is Disable. R indicates Explicit memory block used internally in Application Server. B indicates the type of the Explicit memory block in JavaVM. A indicates the Explicit memory block specified using the automatic allocation configuration file.
<EM_USED>	<const>K	Output used size of Explicit memory block whose sub status is Disable. The unit is kilo bytes.
<EM_GARB>	<const>K	Output value indicating internal status of Explicit memory block.
<EM_TOTAL>	<const>K	Output secured size of Explicit memory block whose sub status is Disable. The unit is kilo bytes.
<TH_PTR>	<ptr>	Output thread ID of thread executing object generation to Explicit heap as the sub status is Disable. Thread ID is same as the tid output to thread dump. This item is output only when <CAUSE> is "New".
<FRAME>	<letters>.<letters>	Output one frame in stack trace output as a result of direct object generation when sub status is Disable due to direct object generation in Explicit heap. Output all class names and method names by delimiting them by a ".". This item is output only when <CAUSE> is "New".
<SOURCE>	(<letters>:<const>)| (Native Method)| (Unknown Source)	Output the source file names where methods output to <FRAME> are described and the line numbers that match the stack trace. Output the file name and line numbers by delimiting them by a ":". Output as "(Native Method)" for native method. Output as "(Unknown Source)" when you cannot acquire source file name. This item is output only when <CAUSE> is "New".

(d) Example of output

Output example.

```
[EVO]<Tue Jul 24 01:23:51 2007>[Alloc failed(Disable)][EH: 12000K(1258K)/150
00K/30000K][E/F/D: 321/0/1][cause:GC]\
["ReferenceExplicitMemory-3" eid=3/R: 108K(20K)/108K]
```

You can confirm the following contents in this output example:

- Output trigger is Disable of sub status of Explicit memory block on July 24, 2007 1:23:51.
- For Explicit heap, 12000K is used and 15000K is secured.
- Maximum size of Explicit heap is 30000K.
- There are only 322 valid Explicit memory blocks in the entire Explicit heap. In these blocks, the blocks whose sub status is Enable are 321 blocks and the blocks whose sub status is Disable is 1 block.
- The migration of objects to the Explicit memory block during GC is the process as a result of which the sub status is Disabled.
- The Explicit memory blocks whose sub status is Disable is when ID is "3" and name of Explicit memory block is "ReferenceExplicitMemory-3".
- In "ReferenceExplicitMemory-3", 108k memory is already used.

(4) Object generation to Explicit memory block

Output when object is generated directly in Explicit memory block using `ExplicitMemory.newInstance()`.

(a) Output trigger

Output is triggered when object is generated in Explicit memory block.

(b) Output format

```
[EVS]<ctime>[EH: <EH_USED_BF>-><EH_USED_AF>(<EH_TOTAL>/<EH_MAX>)][E/F/D: <AC
_NUM>/<FL_NUM>/<DA_NUM>][cause:<CAUSE>]\
["<EM_NAME>" eid=<EID>/<EM_TYPE>: <EM_USED_BF>-><EM_USED_AF>(<EM_TOTAL>)]
```

(c) Output items

The following table describes each item indicated in *(b) output trigger*.

Table 5–46: Output items (Object generation to Explicit memory block)

Output items	Output contents	Meaning
<ctime>	<letters>	Indicates the date and time when object is generated in Explicit memory block. Output in the format same as extended `verbosegc` information. Output in milli-seconds unit when `-XX:+HitachiOutputMilliTime` option is set.
<EH_USED_BF>	<const>K	Output used size of Explicit heap before object generation. The unit is kilo bytes.
<EH_USED_AF>	<const>K	Output used size of Explicit heap after object generation. The unit is kilo bytes.
<EH_TOTAL>	<const>K	Output secured size of Explicit heap after object generation. The unit is kilo bytes.
<EH_MAX>	<const>K	Output maximum size of Explicit heap. The unit is kilo bytes.
<AC_NUM>	<const>	Output the number of Explicit memory blocks whose sub status is Enable after object generation.
<FL_NUM>	<const>	Normally 0 is output
<DA_NUM>	<const>	Output number of Explicit memory blocks whose sub status is Disable after object generation.
<CAUSE>	New	Always output as "New". Indicates that migration of object to Explicit memory block is executed from Java program.
<EM_NAME>	<letters>	Output name of Explicit memory block where object is generated. The output contents are uncertain when multi byte characters are included in name of Explicit memory block (Usually garbled and output).
<EID>	<const>	Output ID of Explicit memory block where object is generated.
<EM_TYPE>	R|B|A	Output type of Explicit memory block where object is generated. R indicates Explicit memory block used internally in Application Server. B indicates the type of the Explicit memory block in JavaVM. A indicates the Explicit memory block specified using the automatic allocation configuration file.

Output items	Output contents	Meaning
<EM_USED_BF>	<const>K	Output used size of Explicit memory block that is generation destination before object generation. The unit is kilo bytes.
<EM_USED_AF>	<const>K	Output used size of Explicit memory block that is generation destination after object generation. The unit is kilo bytes.
<EM_TOTAL>	<const>K	Output secured size of Explicit memory block that is generation destination after object generation. The unit is kilo bytes.

(d) Example of output

Output example.

```
[EVS]<Thu Oct 21 14:55:50 2007>[EH: 150528K->150529K(150532K/1048576K)][E/F/
D: 200/0/0][cause:New]["BEM" eid=2/B: 30K->31K(32K)]
```

You can confirm the following contents in this output example:

- Output trigger is the generation of object to Explicit memory block executed on October 21, 2007 (Thursday) 14:55:50.
- The used size of Explicit heap changed from 150528K to 150529K as object is generated in Explicit heap.
- Secured size of Explicit heap after object generation to Explicit heap is 150532K. Maximum size is 1048576K.
- The number of Explicit memory blocks whose sub status is Enable after object generation to Explicit heap, is 200.
- Changed the used size of memory block named "BEM" from 30K to 31K. Secured size of BEM after object generation is 32K.

(5) Moving to Explicit memory block (Output detail information)

Output detail information for moving object to Explicit memory block. Besides the output contents described in *5.11.3(1) GC occurrence (Output Explicit heap usage status)*, output the usage status of all Explicit memory blocks where the objects are to be moved.

(a) Output trigger

Output is triggered when object moves to Explicit memory block as a result of occurrence of GC.

(b) Output format

```
<Explicit heap usage status when GC occurred>#
[EVS]{["<EM_NAME>" eid=<EID>/<EM_TYPE>: <EM_USED_BF>-><EM_USED_AF>(<EM_TOTAL
>)]}{1,5}
...
```

Note:

Linefeed information of Explicit memory block after output of every five blocks.

#

For output items, see *5.11.3(1) GC occurrence (Output Explicit heap usage status)*.

(c) Output items

The following table describes each item indicated in *(b) output trigger*.

Table 5–47: Output items (Moving to Explicit memory block (Output detail information))

Output items	Output contents	Meaning
Explicit heap usage status when GC occurred		See *5.11.3(1) GC occurrence (Output Explicit heap usage status)*.
<EM_NAME>	<letters>	Output name of Explicit memory block where the object is to be moved when GC occurred. The output contents are uncertain when multi byte characters are included in name of Explicit memory block (Usually garbled and output).
<EID>	<const>	Output ID of Explicit memory block where the object is to be moved when GC occurred.
<EM_TYPE>	R|A	Output type of Explicit memory block where the object is to be moved whenGC occurred. R or A is output. R indicates the type of the Explicit memory block in JavaVM. A indicates the Explicit memory block specified by using the automatic allocation configuration file.
<EM_USED_BF>	<const>K	Output used size before occurrence of GC of Explicit memory block where the object is to be moved when GC occurred. The unit is kilo bytes.
<EM_USED_AF>	<const>K	Output used size after occurrence of GC of Explicit memory block where the object is to be moved when GC occurred. The unit is kilo bytes.
<EM_TOTAL>	<const>K	Output secured size after occurrence of GC of Explicit memory block where the object is to be moved when GC occurred. The unit is kilo bytes.

(d) Example of output

Output example.

```
[ENS]<Thu Oct 21 14:55:50 2007>[EH: 150528K->162816K(162816K/1048576K)][E/F/
D: 200/0/0][cause:GC][CF: 0]
[EVS]["REM2" eid=2/R: 0K->88K(128K)]["REM3" eid=3/R: 30K->230K(256K)]["REM6
" eid=6/R: 30K->2030K(2048K)]\
["Session1" eid=8/R: 30K->2530K(2548K)]["Session2" eid=10/R: 30K->2530K(2548
K)]
[EVS]["Session3" eid=12/R: 30K->5030K(5048K)]
```

You can confirm the following contents in this output example:

- Output trigger is the moving of object to Explicit heap in GC occurred on October 21, 2007 (Thursday) 14:55:50.
- Used size of Explicit heap changed from 150528K to 162816K.
- Secured size of Explicit heap after executing GC is 162816K. Maximum size is 1048576K.
- There are 200 Explicit memory blocks whose sub status is Enable after executing GC.
- Type of GC is copy GC.
- There are minimum five Explicit memory blocks where the objects are to be moved. Perform one linefeed after output of five blocks.
- Following is the breakdown for moving 2288k to (162816K-150528K) explicit heap:
  - Move 88K in Explicit memory block "REM2"(eid=2).
  - Move 200K in Explicit memory block "REM3"(eid=3).
  - Move 2000K in Explicit memory block "REM6"(eid=6).
  - Move 2500K in Explicit memory block "Session1"(eid=8).

- Move 2500K in Explicit memory block "Session2"(eid=10).
- Move 5000K in Explicit memory block "Session3"(eid=12).

Moved to Explicit memory blocks used internally in all Application Servers.

(6) Explicit memory block explicit release process (Output detail information)

Output detail information when the explicit release process of Explicit memory block occurred. Besides the contents output to *5.11.3(2) Explicit memory block explicit release process*, output the information of all the explicitly released Explicit memory blocks.

While explicitly releasing the Explicit memory block, if there is a Java heap overflow, output the information by adding information of Explicit memory blocks explicitly released before overflow. The information described here is not output to the output contents of *5.11.3(3) Java heap overflow in Explicit memory block explicit release process*.

(a) Output trigger

The Explicit release process of Explicit memory block.

(b) Output format

```
<Explicit memory block explicit release process>#
[EVS]{["<EM_NAME>" eid=<EID>/<EM_TYPE>: <EM_TOTAL>]}{1,5}
...
```

Note:

Linefeed information of Explicit memory block after every five blocks.

#

For details on the output item, see *5.11.3(2) Explicit memory block explicit release process*.

(c) Output items

The following table describes each item indicated in *(b) output trigger*.

Table 5–48: Output items (Explicit memory block explicit release process (Output detail information))

Output items	Output contents	Meaning
Information output by release process of Explicit memory block		See *5.11.3(2) Explicit memory block explicit release process*.
<EM_NAME>	<letters>	Output name of the explicitly released Explicit memory block. The output contents are uncertain when multi byte characters are included in name of Explicit memory block (Usually garbled and output).
<EID>	<const>	Output ID of the explicitly released Explicit memory blocks.
<EM_TYPE>	R|B|A	Output type of the explicitly released Explicit memory blocks. R indicates Explicit memory block used internally in Application Server. B indicates the type of the Explicit memory block in JavaVM. A indicates the Explicit memory block specified using the automatic allocation configuration file.

Output items	Output contents	Meaning
<EM_TOTAL>	<const>K	Output secured memory size (memory size released explicitly in the explicit release process) of the explicitly released Explicit memory blocks. The unit is kilo bytes.

(d) Example of output

Output example.

```
[ENS]<Tue Jul 24 01:23:51 2007>[EH: 150528K->149528K(162816K/1048576K), 0.11
29602 secs][E/F/D: 523/0/0]\
[DefNew::Eden: 0K->0K(243600K)][DefNew::Survivor: 0K->0K(17400K)][Tenured: 1
03400K->103400K(556800K)][cause:Reclaim]
[EVS]["REM2" eid=3/R: 300K]["BEM3" eid=5/B: 300K]["BEM1" eid=7/B: 400K]
```

You can confirm the following contents in this output example:

- Output trigger is the explicit release process of Explicit memory block occurred on July 24, 2007 (Friday) 1:23:51.
- The used size of Explicit heap reduced from 150528K to 149528K by the explicit release process of Explicit memory block.
- The secured size of Explicit heap after Explicit memory block explicit release process is 162816K. Maximum size is 1048576K.
- The explicit release process of Explicit memory block took 0.1129602 seconds.
- The number of Explicit memory blocks, whose sub status is `Enable` after Explicit memory block explicit release process, are 523 blocks.
- The memory size of each area of Java heap is not changed as a result of the explicit release process of Explicit memory block. In other words, no objects were moved to Java heap.
- The following Explicit memory blocks were target for the explicit release process:
  - Explicit memory block "REM2"(eid=3) whose secured memory size is 300K
  - Explicit memory block "BEM3"(eid=5) whose secured memory size was 300K
  - Explicit memory block "BEM1"(eid=7) whose secured memory size was 400K

  REM2 is the Explicit memory block used internally in Application Server. BEM3 and BEM1 are Explicit memory blocks used by application.

(7) Release reservation of Explicit memory block by finalize

Output when an Explicit memory block corresponding to the finalize method of the `ExplicitMemory` class is reserved for release as a result of the disconnection of a reference to an `ExplicitMemory` instance.

(a) Output trigger

Output is triggered when release of Explicit memory block is reserved by `ExplicitMemory.finalize()` method.

(b) Output format

```
[EVO]<ctime>[Finalized]["<EM_NAME>" eid=<EID>/<EM_TYPE>: <EM_TOTAL>]
```

(c) Output items

The following table describes each item indicated in *(b) output trigger*.

Table 5–49: Output items (Release reservation of Explicit memory block by finalize)

Output items	Output contents	Meaning
`<ctime>`	`<letters>`	Indicates the reserved date and time for release. Output in the format same as extended `verbosegc` information. Output in milli-seconds unit when `-XX:+HitachiOutputMilliTime` option is set.
`<EM_NAME>`	`<letters>`	Output name of Explicit memory block reserved for release. The output contents are uncertain when multi byte characters are included in name of Explicit memory block (Usually garbled and output).
`<EID>`	`<const>`	Output ID of Explicit memory block reserved for release.
`<EM_TYPE>`	`B`	Output type of Explicit memory block reserved for release. B indicates the type of the Explicit memory block in JavaVM. Indicates Explicit memory block used by the application if Application Server version is earlier than 08-50.
`<EM_TOTAL>`	`<const>K`	Output size secured (Memory size released by release process) by Explicit memory blocks reserved for release. The unit is kilo bytes.

(d) Example of output

Output example.

```
[EVO]<Tue Jul 24 01:23:51 2007>[Finalized]["REM1" eid=3/R: 512K]
```

You can confirm the following contents in this output example:

- Output trigger is reservation for release of Explicit memory block "REM1"(eid=3) on July 24, 2007 (Friday) 1:23:51. Release and reservation is executed by `finalize()`.

(8) Automatic allocation to Explicit Memory Management

A log message is output, when the specified class succeeds in automatic allocation to Explicit Memory Management.

(a) Output trigger

The log message is output, when the specified class succeeds in automatic allocation to Explicit Memory Management.

(b) Output format

```
[EVA]<ctime> creation in explicit memory is succeeded. [class=<CLASSNAME>]
```

(c) Output items

The following table describes the items specified in *(b) Output format*.

Table 5–50: Output items (Automatic allocation to Explicit Memory Management)

Output items	Output contents	Meaning
<ctime>	<letters>	Indicates the occurrence time when the specified class succeeded in automatic allocation to Explicit Memory Management. Output with the format same as would output for the extended verboseGC functionality. Output in milliseconds, when the HitachiOutputMilliTime option is set.
<CLASSNAME>	<letters>	Outputs the fully qualified class name of the class that succeeded in automatic allocation to Explicit Memory Management.

(d) Example of output

An output example is as follows:

```
[EVA]<Tue Jul 24 01:23:51 2007> creation in explicit memory is succeeded. [c
lass=jp.co.sample.Main]
```

You can check the following details in this output example:

- When the Explicit Memory Management functionality succeeds in automatic allocation of the class jp.co.sample.Main to Explicit Memory Management on July 24, 2007 (Tuesday) at 1:23:51.

5.11.5 Contents Output when Output Level is Debug

This subsection describes the contents output for each event when you specify debug in log output level.

Supplement

Use debug, for detail information besides the contents output to verbose, while implementing by debugging. In the logs output within debug, there are some logs with high overheads than that of the logs that require execution of Full GC for output.

(1) Migration of object to Java heap by the explicit release of Explicit memory block

If you want to refer an object in Explicit memory block from the Explicit memory block that is not the target for explicit release, move that object to Java heap for explicitly releasing the Explicit memory block. This log does output of information of object moved to Java heap and object of the reference source.

(a) Output trigger

Output is triggered when an object is moved to Java heap in the explicit release process of Explicit memory block.

(b) Output format

```
[EDO][eid=<EID>: Reference to <REFED_NAME>(<REFED_PTR>), total <R_SIZE>]
[EDO] <REF_NAME>(<REF_PTR>)<REF_GEN>
```

Note:

Output for each object referred by Explicit memory block that is not the target for explicit release.

(c) Output items

The following table describes each item indicated in *(b) output trigger*.

Table 5–51: Output items (Migration of objects to Java heap as a result of the explicit release of Explicit memory block)

Output items	Output contents	Meaning
<EID>	<const>	Output ID of the Explicit memory block that has objects referred from other Explicit heap and that are not target for explicit release in the Explicit memory block explicit release process.
<REFED_NAME>	<letters>	Output all class names of objects (Objects shown in<REF_NAME>(<REF_PTR>)) referred from objects other than Explicit heap that are not target for explicit release in Explicit memory block explicit release process.
<REFED_PTR>	<ptr>	Output memory address before migration of objects indicating <REFED_NAME> to Java heap.
<R_SIZE>	<const>K	Output total size of objects to be moved to Java heap by referring from <REF_NAME>(<REF_PTR>). Total size is the value including the objects in the Explicit memory block that are to be released explicitly and objects that are indirectly referred from <REF_NAME>(<REF_PTR>)#. The unit is kilo bytes.
<REF_NAME>	<letters>|JVM	Output all class names of objects referring <REFED_NAME>(<REFED_PTR>). Output as "JVM" when reference source is stack or internal JavaVM.
<REF_PTR>	<ptr>	Output object indicating <REF_NAME>, stack or memory address in JavaVM.
<REF_GEN>	(eid=<EID>)|(DefNew)|(Tenured)|(JVM)	Output area belonging to <REF_NAME>(<REF_PTR>) or generation name. Output ID of Explicit memory block for Explicit memory block. Output "JVM" when referred from stack or JavaVM.

#

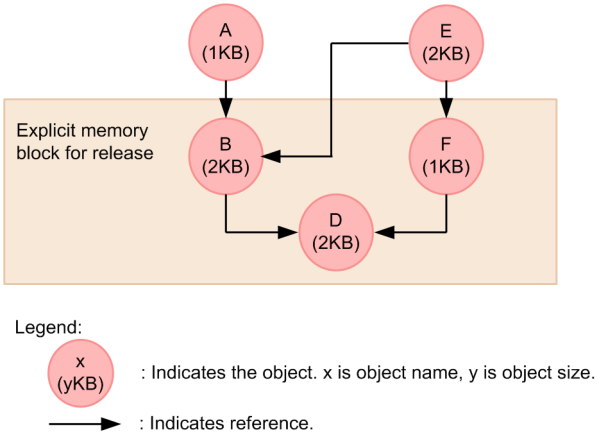
Also refer to the following supplement:

Supplement

This supplement gives an overview when there are many reference routes to an object moving to Java heap.

The description is provided with following figure as an example.

Figure 5–9: Example where there are multiple reference routes to object moving to Java heap



Output when object to be moved to Java heap is referred from multiple objects

There are cases when the objects to be moved to Java heap are referred from multiple objects. In the figure, there are two types; A->B and E->B of reference routes for object B.

In such cases, the information related to reference of A->B, E->B is output separately in logs.

<R_SIZE> calculation when there is indirect referring from multiple objects

Output value that includes objects in the Explicit memory block that are to be released explicitly and objects that are referred indirectly from <REF_NAME>(<REF_PTR>) in <R_SIZE>. However, the calculation method differs when object to be moved to Java heap is indirectly referred from multiple objects.

In the figure, object D is indirectly referred by the following three types of routes:

- A->B->D
- E->B->D
- E->F->D

When referred by multiple routes, the size of object is added to <R_SIZE> reference relation calculated initially. For example, when the reference relation is output in the following A->B, E->B, E->F sequence, the size of object D is added in <R_SIZE> of A->B. In such cases, the output of <R_SIZE> of each reference relation would be as follows:

- A->B
  Total of size of object B and object D is 4K.
- E->B
  2k that is the size of object B.
- E->F
  1K that is the size of object F.

(d) Example of output

Output example.

```
[EDO][eid=5: Reference to java.lang.HashMap$Entry(0x1234568), total 125K]
[EDO] java.util.HashMap$KeyIterator(0x1134428)(eid=1)
[EDO][eid=5: Reference to ClassA(0x1234580), total 19250K]
[EDO] ClassV(0x1234468)(Tenured)
[EDO][eid=5: Reference to ClassZ(0x1234680), total 12K]
[EDO] ClassU(0x1233468)(DefNew)
[EDO][eid=9: Reference to JP.co.Hitachi.soft.jvm.BBB(0x1034428), total 1250
K]
[EDO] JVM(0x23456780)(JVM)
```

You can check the following contents in this output example:

- Object of `java.lang.HashMap$Entry` that is in the Explicit memory block of eid=5 is referred from object of `java.util.HashMap$KeyIterator` that is in the Explicit memory block of `eid=1` which is not the explicit release target. As a result, object of 125Kbyte is moved to Java heap.
- Object of `ClassA` that is in Explicit memory block of `eid=5` is referred from object of `ClassV` in Tenured area. As a result, an object of 19,250 Kbyte is moved to Java heap.
- Object of `ClassZ` in Explicit memory block of `eid=5` is referred from object of `ClassU` in New area. As a result, object of 12Kbyte is moved to Java heap.

- Object of `JP.co.Hitachi.soft.jvm.BBB` that is in Explicit memory block of `eid=9` is referred from stack or internal JavaVM. As a result, an object of 1,250 kilobyte is moved to Java heap.

(2) Initialization of Explicit memory block (Output detail information)

Output detail information when you initialize a new Explicit memory block. Besides the output contents in *5.11.4(1) Initialization of Explicit memory block*, output the stack trace of Java program executing initialization process. This log is output asynchronously with the execution of Java program on multiple lines. Hence, there are cases where some other log is output within the lines of this log. However, other logs are not output within one line.

(a) Output trigger

Initialization of Explicit memory block.

(b) Output format

```
<Information(verbose) of initialization of Explicit memory block>#[Thread: <
TH_PTR>]
[EDO][Thread: <TH_PTR>] at <FRAME><SOURCE>
...
```

#

For output items, see *5.11.4(1) Initialization of Explicit memory block*.

(c) Output items

The following table describes each item indicated in *(b) output trigger*.

Table 5–52: Output items (Initialization of Explicit memory block(Output detail information))

Output items	Output contents	Meaning
Information(`vervose`) when initialization of Explicit memory block		See *5.11.4(1) Initialization of Explicit memory block*.
`<TH_PTR>`	`<ptr>`	Output thread ID of thread to initialize Explicit memory block. Thread ID is same as tid output to thread dump.
`<FRAME>`	`<letters>.<letters>`	Output one frame in stack trace when initializing Explicit memory block. Output all class names and method names by delimiting them by a ".".
`<SOURCE>`	`(<letters>:<const>)|(Native Method)|(Unknown Source)`	Output the source file names where methods output to `<FRAME>` are described and line numbers match the stack trace. Output file names and line numbers by delimiting them by a ":". Output as "(Native Method)" when native method. Output as "(Unknown Source)" when you cannot acquire the source file name.

(d) Example of output

Output example.

```
[EVO]<Tue Jul 24 01:23:51 2007>[Created]["BasicExplicitMemory-2" eid=2(0x123
4568)/B][Thread: 0x00035a60]
[EDO][Thread: 0x00035a60] at ExplicitMemory.registerExplicitMemory(Native Me
thod)
[EDO][Thread: 0x00035a60] at BasicExplicitMemory.<init>(Unknown Source)
```

```
[EDO][Thread: 0x00035a60] at AllocTest.test(AllocTest.java:64)
[EDO][Thread: 0x00035a60] at java.lang.Thread.run(Thread.java:2312)
```

You can check the following contents in this output example:

- Output trigger is initialization of memory block executed on July 24, 2007 1:23:51. Name of Explicit memory block is "BasicExplicitMemory-2". ID of Explicit memory block is eid=2.
- Try initializing Explicit memory block by executing constructor of `BasicExplicitMemory` class on 64th line of AllocTest.java.

(3) Details of automatic release processing of the Explicit memory block (Output detailed information)

Output the detailed information when the automatic release processing of the Explicit memory block occurs. Besides the output contents described in the subsection *5.11.3(4) Automatic release processing of the explicit memory block*, output the EID information of the Explicit memory block executing the automatic release processing. The automatic release processing of the Explicit memory block might include the automatic release processing without the movement of objects, many-to-one automatic release processing, and one-to-one automatic release processing. Therefore, the information about EID without object movement, many-to-one EID, and one-to-one EID is output.

(a) Output trigger

The log is output when the automatic release processing for the explicit memory block occurs.

(b) Output format

```
[EDO][migrate:(<EID_DEL>{,<EID_DEL>}*|)/(<EID_MBF>{,<EID_MBF>}*-><EID_MAF>|)
/(<EID_MIG>{,<EID_MIG>}*|)]
```

(c) Output items

The following table describes the items specified in *(b) Output format*.

Table 5–53: Output items (Details of automatic release processing of the Explicit memory block (output detailed information))

Output items	Output contents	Meaning
<EID_DEL>	<const>	Outputs EID of the Explicit memory blocks without moving the objects, among the Explicit memory blocks released by the automatic release processing of the Explicit memory block.
<EID_MBF>	<const>	Outputs EID used before reserving the automatic release of the Explicit memory block where many-to-one automatic release processing occurred, among the Explicit memory blocks released by the automatic release processing of the Explicit memory block.
<EID_MAF>	<const>	Outputs EID of the Explicit memory blocks generated by the many-to-one automatic release processing, among the Explicit memory blocks generated by the automatic release processing of the Explicit memory block.
<EID_MIG>	<const>	Outputs EID of the Explicit memory blocks where the one-to-one automatic release processing occurred, among the Explicit memory blocks released by the automatic release processing of the Explicit memory block.

(d) Example of output

An output example is as follows:

```
[EVS]<Tue Jul 14 02:31:22 2009>[EH: 256512K->256128K(256256K/1048576K), 0.11
24626 secs][E/F/D: 423/0/0][target:584K/384K/200K]\
[cause:Migrate]
[EDO][migrate:()/(2,4,6,9->10)/(1,8)]
```

You can check the following details in this output example:

- The output trigger is the automatic release processing of the Explicit memory block in the GC that occurred on July 14, 2009 (Tuesday) at 2:31:22.
- The used size of Explicit memory changed from 256,512K to 256,128K due to the automatic release processing.
- The secured size of Explicit memory after the automatic release processing is 256,256K and the maximum size is 1,048,576K.
- The time taken for the automatic release processing was 0.1124626 seconds.
- There are 423 Explicit memory blocks whose sub-state is Enable after the automatic release processing.
- The automatic release processing was executed for the Explicit memory used size 584K.
- The Explicit memory blocks with IDs (2, 4, 6, 9) are moved to the Explicit memory block with ID (10).
- The Explicit memory blocks with IDs (1, 8) are moved to the same IDs (1, 8).

6 Troubleshooting Procedure

This chapter describes the troubleshooting procedure for Application Server.

6.1 Organization of this chapter

This chapter describes the main problems that occur in Application Server and how to troubleshoot the problems.

The following table describes the organization of this chapter.

Table 6–1: Organization of this chapter (Troubleshooting procedure)

Category	Title	Reference section
Explanation	List of main problems	*6.2*
	Processes that output logs	*6.3*
	Troubleshooting during setup	*6.4.1*
	Troubleshooting during operations	*6.4.2*
	Troubleshooting the server management commands	*6.4.3*
	Examples of troubleshooting during operations	*6.5*

6.2 List of main problems

This section describes the locations to check the main causes of problems and logs required for actions, for the different times at which problems occur.

This subsection describes the main causes and actions for problems, for each of the following times:

- During installation
- During server setup
- During server startup
- During application startup
- During operations
- During server or application maintenance

The following is a description of each of the above problems:

6.2.1 Main problems occurring during installation

The following table describes the main problems that occur during installation.

Table 6–2: List of main problems occurring during installation

Tool reporting the problem	Event	Main effects	Main causes	Checking location
Installer	Output of error message (dialog box or command prompt)	Installation will be interrupted. In some cases, un-installation is required.	There is a problem with the environment (such as the OS or disk).	• Dialog box • `install.log`

Legend:
--: Not applicable.

Reference note

For details on the notes related to installation and un-installation, see *Appendix I Notes on installation and un-installation* in the *uCosminexus Application Server System Setup and Operation Guide*.

6.2.2 Main problems occurring during server setup

The following table describes the main problems that occur during server setup.

Table 6–3: List of main problems occurring during server setup

Tool reporting the problem	Event	Main effects	Main causes	Checking location	Reference location
Setup Wizard	Output of error message (command prompt)	The setup operation is interrupted. In some cases, un-setup is required.	The possible causes are as follows: • There is a problem with the environment (such	• Command error • Setup Wizard log • Manager log	*6.4.1*

Tool reporting the problem	Event	Main effects	Main causes	Checking location	Reference location
			as the OS, network, memory, disk) • Operation error • Settings error	• J2EE server log • Web server log • OS	
SmartComposer	Output of error message (command prompt)			• Command error • Manager log • J2EE server log • Web server log • OS	
Management portal	Output of error message (GUI)			• Displayed log • Manager log • J2EE server log • Web server log • OS	

6.2.3 Main problems occurring during server startup

The following table describes the main problems that occur when you start a server.

Table 6–4: List of main problems occurring during server startup

Tool reporting the problem	Event	Main effects	Main causes	Checking location	Reference
SmartComposer	Output of error message (command prompt)	The deploy operation is interrupted.	The possible causes are as follows: • There is a problem with the environment (such as the OS, network, memory, disk) • Operation error • Settings error	• Command error • Manager log • J2EE server log • Web server log • OS	*6.4.1*
Management portal	Output of error message (GUI)			• Displayed log • Manager log • J2EE server log • Web server log • OS	

6.2.4 Main problems occurring during application startup

The following table describes the main problems that occur when you start an application.

Table 6–5: List of main problems occurring during application startup

Tool reporting the problem	Event	Main effects	Main causes	Checking location	Reference
SmartComposer (for compatibility)	Output of error message (command prompt)	The deploy operation is interrupted.	The possible causes are as follows: • There is a problem with the environment (such	• Command error • Manager log • J2EE server log • Web server log • OS	*6.4.1*

Tool reporting the problem	Event	Main effects	Main causes	Checking location	Reference
Management portal	Output of error message (GUI)		as the OS, network, memory, disk) • Operation error • Settings error • There is a problem with the created application • There is a problem between an application and Application Server	• Displayed log • Manager log • `admin` log • J2EE server log • Web server log • OS	
Server management commands#	Output of error message (command prompt)			• Command error • `admin` log • J2EE server log • Application	*6.4.3*

#

When you use the server management commands, the action to be taken for the problems varies. Check the action described at the *Reference* column indicated in the table.

6.2.5 Main problems occurring during operations

The following table describes the main problems that occur during operations.

Table 6–6: List of main problems occurring during operations

Tool reporting the problem	Event	Main effects	Main causes	Checking location	Reference
Management portal	Output of error message (GUI)	The business cannot continue.	The possible causes are as follows: • There is a problem with the environment (such as the OS, network, memory, disk) • Resource depletion in the server (such as OOM) • There is a problem with the integrated system (such as DB) • Lengthy processing on an application • HA switching	• Displayed log • Manager log • J2EE server log • Web server log • OS • JavaVM, container resources • PRF trace • Thread dump • Application	*6.4.2*
(Client)#	• Output of business errors • Browser-specific errors (such as server not found, 404, 500, 503) • No response • Delayed response (performance deterioration)	The business might not be continued.		• Displayed log • Manager log • J2EE server log • Web server log • OS • JavaVM, container resources • PRF trace • Thread dump • Application	

#

Appears as an event on the client using the service.

6.2.6 Main problems occurring during server/application maintenance

The following table describes the main problems that occur during server/application maintenance.

Table 6–7: List of main problems occurring during server/application maintenance

Tool reporting the problem	Event	Main effects	Main causes	Checking location	Reference
SmartComposer	Output of error message (command prompt)	• The server settings cannot be changed • The application cannot be updated	The possible causes are as follows: • There is a problem with the environment (such as the OS, network, memory, disk) • Operation error • Settings error • There is a problem with the created application • There is a problem between an application and Application Server	• Command error • Manager log • J2EE server log • Web server log • OS	*6.4.1*
Management portal	Output of error message (GUI)			• Displayed log • Manager log • `admin` log • J2EE server log • Web server log • OS	
Server management commands#	Output of error message (command prompt)			• Command error • `admin` log • J2EE server log • OS • Application	*6.4.3*

#

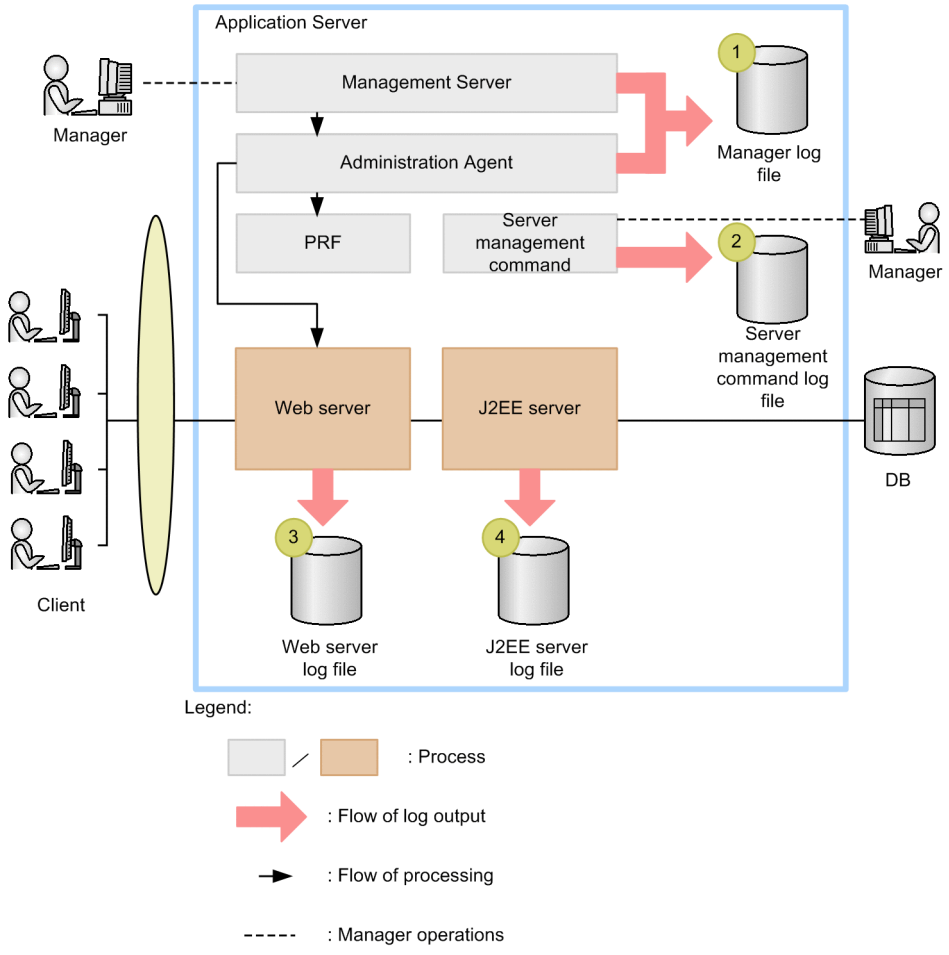
The action to be taken for the problems varies when you use the server management commands. Check the action described at the *Reference* column indicated in the table.

6.3 Processes that output logs

Multiple logs are output by Application Server. You reference the logs to identify the cause of a problem, and then take action.

This section describes the logs that are mainly used for troubleshooting and the processes that output the logs. The following figure shows the logs that are mainly used for troubleshooting and the processes that output the logs.

Figure 6–1: Processes that output logs



Note: Apart from the log file shown here, setup wizard log is also checked.

Four types of logs are mainly used for troubleshooting with Application Server. The following table describes each of the logs.

Table 6–8: Log types and output contents

Number in the figure	Log type	Output contents
1	Manager logs	Logs output by Management Server or Administration Agent. This type of log is output during the setup, operations, and maintenance by using the Cosminexus Manager functionality.
2	Server management command logs	This type of log is output during the server management command operations.
3	Web server logs	This type of log is output by Cosminexus HTTP Server.
4	J2EE server logs	This type of log is output by the J2EE server.

During troubleshooting, you check these logs and then take action. The following sections describe specific troubleshooting procedure for each tool used.

6.4 Overview of troubleshooting

6.4.1 Troubleshooting during setup

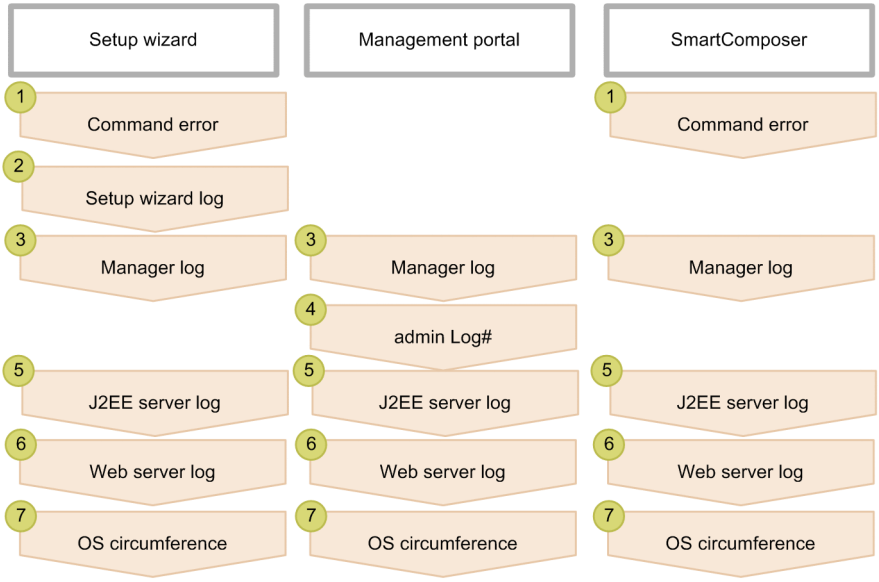
This section describes the actions for problems that occur during setup.

Setup refers to the following times:

- During server setup
- During server startup
- During application startup
- During server/ application maintenance

The following figure shows the procedure of checks during setup.

Figure 6–2: Procedure of checks for the tools being used (During setup)



This is a necessary check if an error occurs at the time of application start.

Implement the required checks as per the procedure shown in the figure for the tools being used. The following subsections describe the details for the respective checks.

Reference note

Use the management portal to check the Manager logs. For details on the Management portal, see the *uCosminexus Application Server Management Portal User Guide*.

(1) Checking the command errors

Tip

This check is required when you use the following tools:

- Setup Wizard
- SmartComposer

The errors are displayed in the Setup Wizard and Smart Composer windows. Check the displayed error contents, and then take action. For details on the actions for the error messages, see the manual *uCosminexus Application Server Messages*.

(2) Checking the Setup Wizard logs

Tip

This check is required when you use the following tool:

- Setup Wizard

Check the Setup Wizard logs. The output destination of the Setup Wizard logs is specified in `setup.log.dir` of `setup.cfg`. For details on `setup.cfg`, see *8.2.18 setup.cfg (Setup file for the Setup Wizard)* in the *uCosminexus Application Server Definition Reference Guide*.

(3) Checking the Manager logs

Tip

This check is required when you use the following tools:

- Setup Wizard
- SmartComposer

Check the logs output by Manager. The following table lists and describes the log files to be checked.

Log file name#	Log description	Checking method
mngsvr?.log	Management Server log	Check the contents of the error message ID (message with message type E) for the time at which the error message is output by Cosminexus Manager.

#:

? indicates the number of log files.

For details on the default output destination of the log files, see *4.3.1 Acquiring the Cosminexus Component Container Logs*.

(4) Checking the admin logs

Tip

This check is required when you use the following tools:

- Management portal

Note that this check is necessary when an error occurs while an application is being started.

Check the server management command logs.

The server management command logs include the message log, exception log, and maintenance log. These logs output information such as the operating status of the server management commands, the standard output and standard error output information output in an application, and the information used by the maintenance personnel to analyze the Component Container errors.

For details on the default output destination and the names of the log files, see the description related to the server management command logs in *4.3.1 Acquiring the Cosminexus Component Container Logs*.

(5) Checking the J2EE server logs

Tip

This check is required when you use the following tools:

- Setup Wizard
- Management portal
- SmartComposer

Check the contents of the log files output by the J2EE server. The following table lists and describes the log files to be checked.

Table 6–9: List of log files to be collected

Log file name#	Log description	Checking method
`cjmessage?.log`	Log for J2EE server operations	Check the contents of the error message ID (message with message type E) for the time at which the error message is output by Cosminexus Manager.
`cjexception?.log`	Exception information log for J2EE server operations	
`web_servlet?.log`	Web servlet log (messages output in JSP/ Servlet)	
`user_out?.log`	User output log (standard output of messages in applications)	
`user_err?.log`	User error log (standard error output of messages in applications)	
`javalog?.log`	Logs for JavaVM maintenance information and GC information	
`hs_err?`	JavaVM error report file	Send to maintenance personnel.

#:

? indicates the number of log files.

For details on the default output destination of the log files, see *4.3.1 Acquiring the Cosminexus Component Container Logs*.

For details on how to acquire `hs_err`, see *4.11 JavaVM Output Message Logs (Standard Output or Error Report File)*.

(6) Checking the Web server logs

Tip

This check is required when you use the following tools:

- Setup Wizard
- Management portal
- SmartComposer

Check the log files output by the Web server. The following table lists and describes the log files to be checked.

Table 6–10: List of log files to be collected

Log file name#	Log description	Checking method
`processConsole?.log`	Console logs for Manager (Error information when the Web server starts)	Check the contents of the error message ID (message with message type E) for the time at which the error message is output by Cosminexus Manager.
`error?`	Web server error log	Check the contents of the error message (`emerg`, `alert`, and `crit`) for the time at which the error message is output by Cosminexus Manager.

#:

? indicates the number of log files.

For details on the default output destination of the console logs for Manager, see *4.11 JavaVM Output Message Logs (Standard Output or Error Report File)*.

For details on the default output destination of the Web server error logs, see *6.2.4 Directives that start with E, F, G, H, and I* in the *uCosminexus Application Server HTTP Server User Guide*.

(7) OS peripheral checks

Tip

This check is required when you use the following tools:

- Setup Wizard
- Management portal
- SmartComposer

Check the memory size and operating status of the machine on which Application Server is installed.

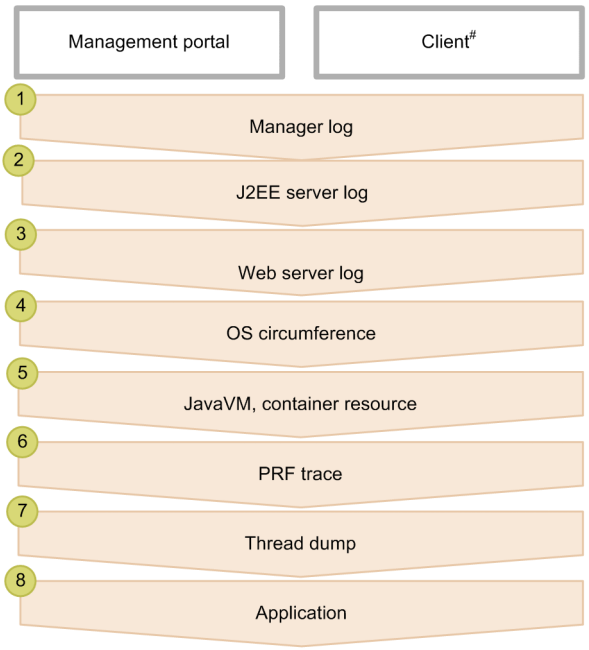
6.4.2 Troubleshooting during operations

This section describes the actions to be taken for the problems occurring during operations.

"During operations" indicates the time immediately after the server startup is complete or when the server is running.

The following figure shows the procedure of checks during the operations of the tools being used.

Figure 6–3: Procedure of checks for the tools being used (during operations)



#: It emerges as the phenomenon of the client side using the service.

Implement the required checks as per the procedure shown in the figure for the tools being used. The following subsections describe the details for the respective checks.

(1) Checking the Manager logs

Check the log files output by Manager. For details on the Manager log checks, see *6.4.1(3) Checking the Manager logs*.

(2) Checking the J2EE server logs

Check the log files output by the J2EE server. For details on the J2EE server log checks, see *6.4.1(5) Checking the J2EE server logs*.

(3) Checking the Web server logs

Check the log files output by Web server. For details on the Web server log checks, see *6.4.1(6) Checking the Web server logs*.

(4) OS peripheral checks

Check the memory size and operating status of the machine on which the Application Server is installed.

(5) Checking JavaVM and container resources

Check whether there are errors in JavaVM (occurrence of OutOfMemoryError) and in resources. For details on the action to be taken when an error occurs in JavaVM, see *2.5.4 If JavaVM Terminates abnormally*.

(6) Checking the PRF trace

Check the trace based performance analysis and verify if there are any bottlenecks or if the processing is delayed at any location. For details on the trace based performance analysis, see *7. Performance Analysis by Using Trace Based Performance Analysis*.

(7) Checking the thread dump

Check the thread dump and verify if there are any deadlocks or if there are any errors in the Java programs. For details on the information output to a thread dump, see *5.5 JavaVM Thread Dump*.

(8) Checking the applications

Check the contents of the applications in which you think errors might have occurred. Request the creators of the application to perform the check.

6.4.3 Troubleshooting the server management commands

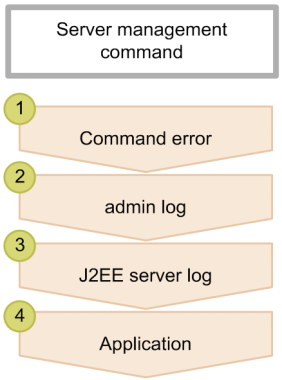
This section describes the actions to be taken for the problems that occur when the server management commands are used.

The actions for problems described in this section are as follows:

- During application startup
- During server/ application maintenance

The following figure shows the procedure of checks using the server management commands.

Figure 6–4: Procedure of checks using the server management commands



The details of the respective operations are as follows:

(1) Checking the command errors

The errors are displayed on the command prompt. Check the displayed error contents, and then take action.

(2) Checking the admin logs

Check the server management command logs. For details on the default output destination of the log files, see the description related to the server management command logs in *4.3.1 Acquiring the Cosminexus Component Container Logs*.

(3) Checking the J2EE server logs

Check the log files output by the J2EE server. For details on the J2EE server log checks, see *6.4.1(5) Checking the J2EE server logs*.

(4) Checking the applications

Check the contents of the applications in which you think errors might have occurred. Request the creators of the application to perform the check.

6.5 Examples of troubleshooting during operations

This section describes the procedure of troubleshooting for the following problems as examples of troubleshooting during operations:

- Process down
- Delayed response

Important note

This section specifies the default path settings of Application Server. Also, the OS used is Windows.

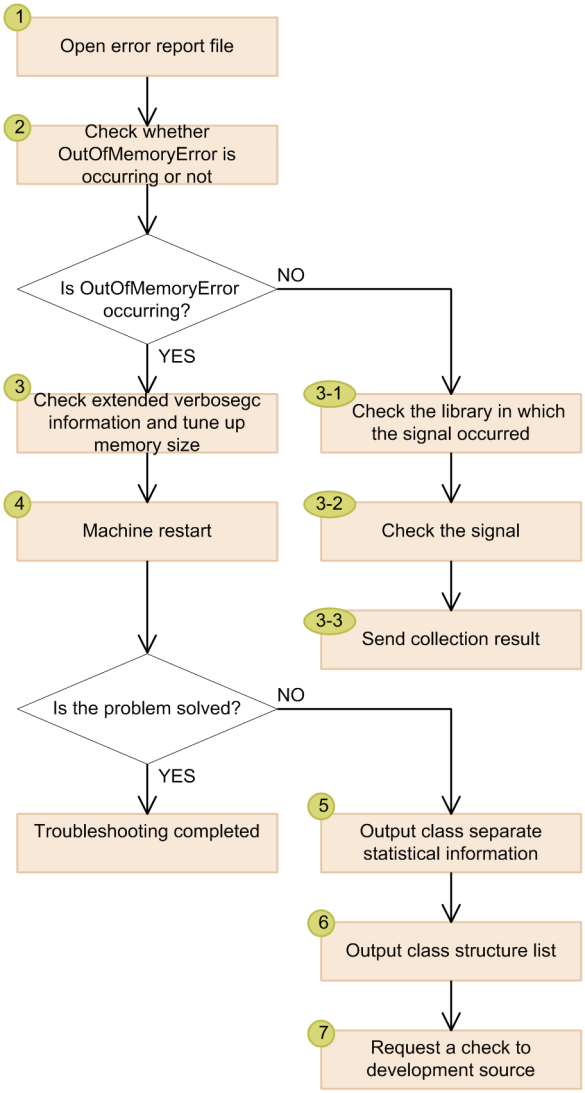
6.5.1 Troubleshooting when a process is down

This subsection describes troubleshooting when a process is down.

(1) Flow of actions when a process is down

The following figure shows the flow of troubleshooting when a process is down.

Figure 6–5: Flow of actions when a process is down



The details of the processing shown in the figure are described in subsection (2).

(2) Flow of actions when a process is down

The following points describe the operations according to the contents of the process down flow:

1. Open the error report file (`hs_err_pid`*process-ID*`.log`)

   Open the error report file that was output around the time the error occurred and compare it with the Manager log.

   - Output destination of the error report file and the output file name

     `C:\Program Files\Hitachi\Cosminexus\CC\server\public\ejb\`*server-name*`\hs_err_pid`*server-process-ID*`.log`

2. Check whether OutOfMemoryError occurs

   If OutOfMemoryError has occurred, `java.lang.OutOfMemoryError occurred.` is displayed.

   The following are examples of the output of the error report file and the action to be taken when OutOfMemoryError has occurred and when OutOfMemoryError has not occurred.

   When OutOfMemoryError has occurred

   Output example

```
  :
#
# java.lang.OutOfMemoryError occurred.
# JavaVM aborted because of specified -XX:+HitachiOutOfMemoryAbort opti
ons.
# Please check Javacorefile:D:\Cosminexus\CC\server\public\ejb\MyJ2EESe
rver\javacore2100.120124144835.txt
#
```

The part shown in bold with a colored background is the string that is output when OutOfMemoryError has occurred.

Action

You need to examine the error report file. Go to step 3.

When OutOfMemoryError has not occurred

Output example

```
 :
#
# A fatal error has been detected by the Java Runtime Environment
#
# EXCEPTION_ACCESS_VIOLATION (0xc0000005) at pc=0x0000000008303b00, pid
=1356, tid=2604
#
# Java VM: Java HotSpot(TM) 64-Bit Server VM (25.20-b23-CDK0970-2015012
7 mixed mode windows-amd64)
# Problematic frame:
# V [C:\Program Files\Hitachi\Cosminexus\jdk\jre\bin\server\jvm.dll+0x3
03b00]
#
```

Action

The following checks are required. Perform the following steps. Note that the step numbers correspond to *Figure 6-5*.

3-1 Check the library where the signal occurred

Check the red portion in the example of output, and make a note of the down library name and the down native function.

In the example of output, the library name is `NativeCrash.dll` and the down native function is `(null)+0x77C785BA`.

3-2 Check the signal

If the location specified after "An unexpected error has been detected by HotSpot Virtual Machine: " is down due to signal 6 (SIGABRT, SIGIOT), the invocation destination library must be checked because the library is aborted by a higher abort function.

The above example is not aborted.

In this example, request a detailed check to the Oracle support service.

3-3 Send the collected results to the helpdesk

Send the library you have noted and the signal contents to the module developer or to the helpdesk based on the purchase agreement and request a check.

3. Check the extended verbosegc information and tune the memory size

Check the extended verbosegc information and adjust the Java heap memory size.

Using the set value of the `-Xmx` option as an indicator, specify 1.5 to 2 times the value. However, note that the hard disk memory size must not be exceeded.

4. Restart the machine

   Restart the machine and check if the operations can be executed without any problems.

   Tip

   Troubleshooting is complete when you restart the machine and can perform operations without any problems.

   If the operations are not performed correctly even after the machine is restarted, go to step 5.

5. Output the class-wise statistics

   Check the JavaVM GC log (`javalog[`*n*`].log`), or operation log (`HJVMStats_`*YYYYMMDDhhmmTZ*`.csv`), for every 10 MB to 20 MB memory increased, execute the `jheapprof` command to output the class-wise statistics (txt file).

   Repeat the process until the memory is increased to 100 MB or more.

   If the command execution interval is short, the leaked memory size is also small, so it is difficult to find the class for which the memory has been increased.

   By increasing the execution interval and incrementing the increase in memory, the class from which the memory is leaking is made to stand out.

   Example of execution

   ```
   % jheapprof -p 2463
   ```

   Note

   `2463` in the example of execution becomes the process ID. When you execute the command, specify the process ID with the problem.

6. Output the class structure list

   List and output the class structure containing the class picked up in the previous step as a member, create the class list to be checked, and then list up the check targets.

   Example of execution

   ```
   % jheapprof -class org.apache.catalina.loader.WebappClassLoder -p 2463
   ```

   Note

   `2463` in the example of execution becomes the process ID. When you execute the command, specify the class with the problem.

   `org.apache.catalina.loader.WebappClassLoder` in the example of execution becomes the class name. Use this class to check OutOfMemoryError caused by an insufficient Metaspace area.

7. Request the developer to perform the check

   Send the information collected in the previous step to the developer and request a check.

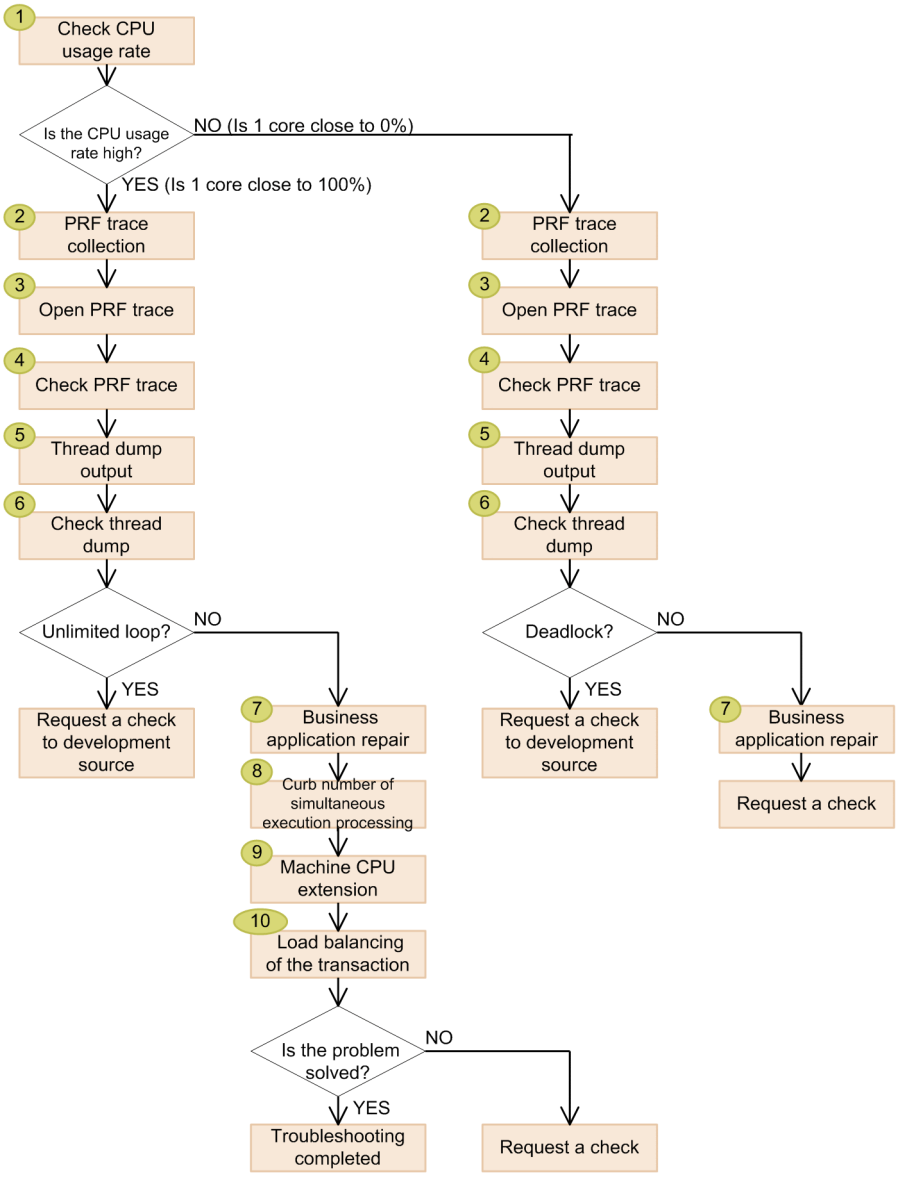
6.5.2 Troubleshooting when a response is delayed

This subsection describes troubleshooting for a delayed response.

(1) Flow of actions when a response is delayed

The following figure shows the flow of troubleshooting when a response is delayed.

Figure 6–6: Flow of actions when a response is delayed



The details of the processing shown in the figure are described in subsection (2).

(2) Flow of actions when a response is delayed

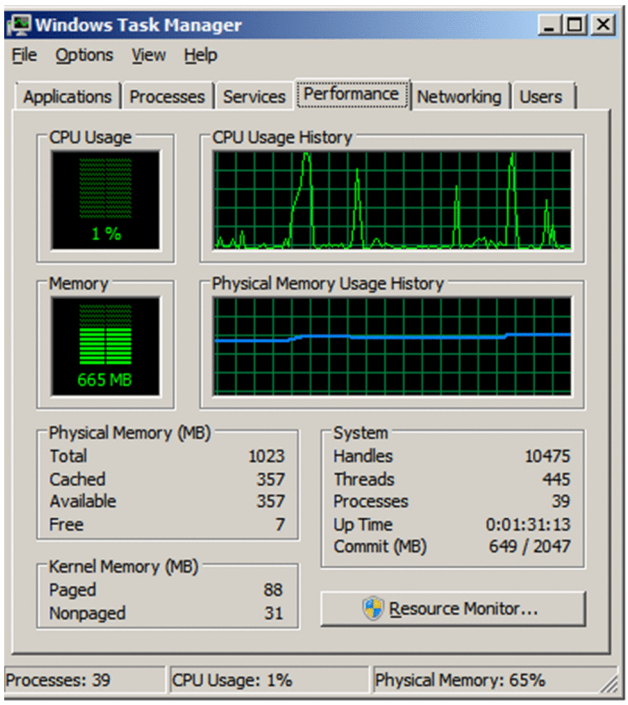
The following points describe the operations according to the contents of the delayed response flow:

1. Check the CPU usage

   Check the CPU usage of the applicable process.

   The following is an example display of CPU usage with the task manager.

Figure 6–7: CPU usage



Tip

If 1core is close to 100%

This includes cases that run into an infinite loop and recursive invocation. A CPU bottleneck is a possible cause. Go to step 2 and proceed with the check.

If 1core is close to 0%

A possible cause is a non-responding or deadlocked back-end process, based on the reason that the back-end process does not return a response. Go to step 2 and proceed with the check.

2. Acquire the PRF trace

Execute the mngsvrutil command to output the PRF trace.

Example of execution

```
mngsvrutil -m 123.45.67.89 -u admin2 collect allPrfTraces
```

3. Open the PRF trace

Open the PRF trace.

Output destination

```
C:\Program Files\Hitachi\Cosminexus\manager\log\prf
```

File name

The file is output with the following file names for the trace information to be collected.

Note that the date and time at which the PRF trace was collected is displayed in *date-and-time*.

Performance tracer types	File name
All the performance tracers running on the hosts in the management domain	*management-domain-name*`-`*date-and-time*`.zip`
All the performance tracers running on a specific host	*host-name*`-`*date-and-time*`.zip`

Performance tracer types	File name
Specific performance tracer	*logical-server-name*-*date-and-time*.zip

4. Check the PRF trace

   Check the Time column in PRF trace and find the processing that requires a long period of time.

   The PRF trace is a trace information that outputs events across processes and effective data for performance analysis or error analysis.

   Figure 6–8: Example of output of PRF trace

PRF	Event	Date	Time	Time(msec/usec/nsec)	RootAP CommNo.
Rec	0x8000	2011/12/14	17:21:31	080/855/000	0x000000000000d613
:	:	:	:	:	:
Rec	0x8c35	2011/12/14	17:21:31	086/582/000	0x000000000000d613
Rec	0x8cd0	2011/12/14	17:21:31	086/593/000	0x000000000000d613
Rec	0x8cd1	2011/12/14	17:21:42	661/097/000	0x000000000000d613
Rec	0x8c22	2011/12/14	17:21:42	061/282/000	0x000000000000d613
Rec	0x8c23	2011/12/14	17:21:43	663/449/000	0x000000000000d613
:	:	:	:	:	:

   In the example, there is a gap of 11 minutes after the SQL statement is issued. Furthermore, the execution of the SQL statement has not ended. Therefore, a problem might have occurred in the database while the SQL statement was being executed.

   Note that the PRF trace is easy to check if you use spreadsheet software.

5. Output the thread dump

   Execute the `mngsvrutil` command to output the thread dump.

   Example of execution

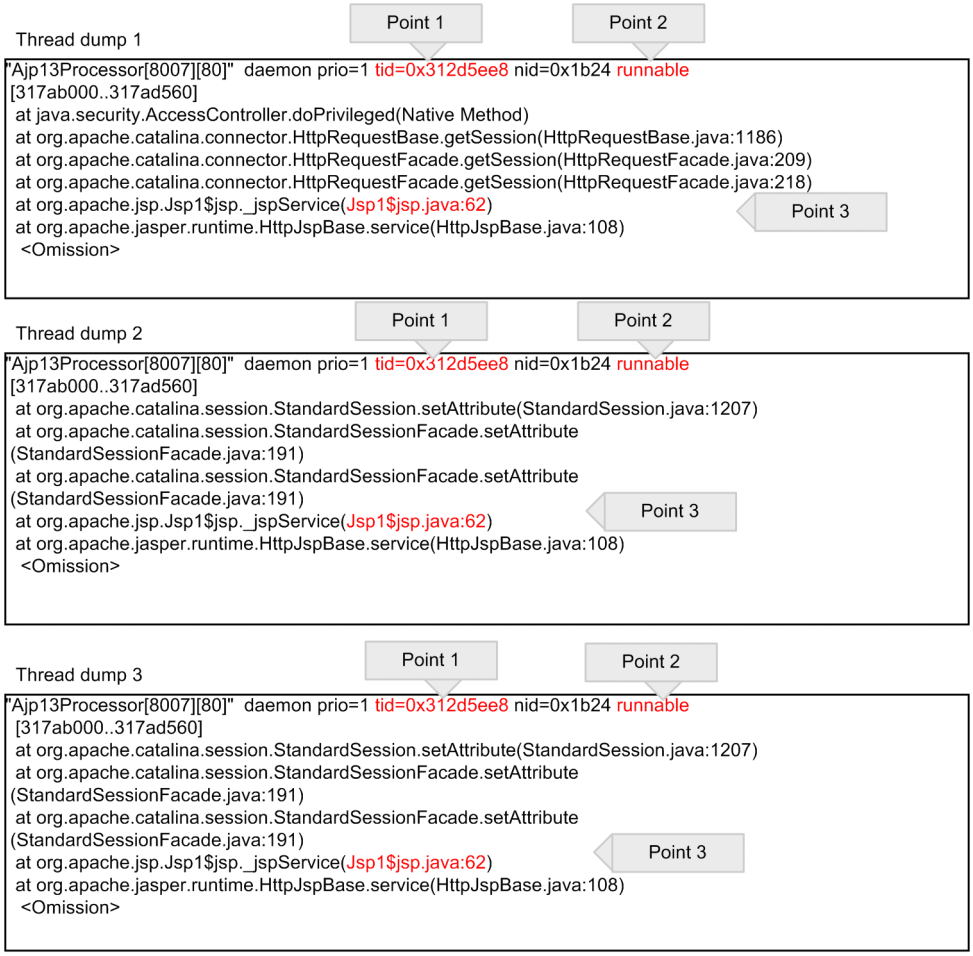
   ```
   mngsvrutil -m 123.45.67.89 -u admin2 dump server
   ```

6. Check the thread dump

   For an infinite loop

   The following figure shows an example of the thread dump output and the check points in the case of an infinite loop.

Figure 6–9: Example of thread dump output (infinite loop)



Output the thread dump multiple times, observe the time series, and perform a comparative check of the stack trace of the threads with the same `tid` in each thread dump.

Point 1

If the thread attribute is `runnable`, this thread is executable. This thread is participating in the increased CPU usage (if the attribute is `waiting for monitor entry`, the thread is not executable and so does not increase the CPU usage).

Point 2

All the thread attributes with the same `tid` are `runnable` in multiple thread dump files.

The threads might be running for a long period of time.

Point 3

If a specific line in the same method is being executed repeatedly, an infinite loop might be suspected.

Tip

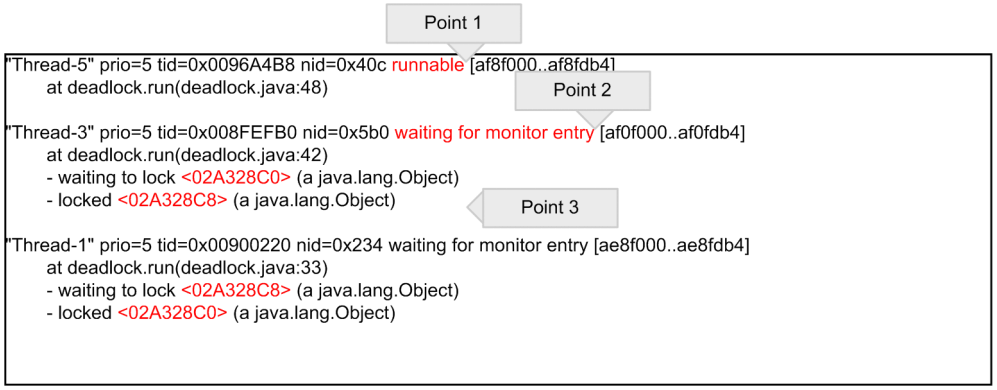
If an infinite loop is suspected in the checks until now, request the developer to perform the check.

If an infinite loop is not suspected, go to step 7.

For a deadlock

The following figure shows an example of thread dump output and the check points in the case of a deadlock.

Figure 6–10: Example of thread dump output (deadlock)



The above figure shows an example of thread dump when a deadlock occurs.

The thread attributes are output after `nid:...` in the example of output.

Find the thread with the attribute `waiting for monitor entry`.

Check the contents of "-waiting to lock..." and "-locked...". There is a deadlock if the threads are waiting to acquire a lock for the areas that are mutually locked.

Point 1

If the thread attribute is `runnable`, this thread is executable, and so this thread is irrelevant to a deadlock.

Point 2

If the thread attribute is `waiting for monitor entry`, it indicates that this thread is waiting to acquire a lock.

This thread might have caused the deadlock.

Point 3

If a thread has acquired a lock, and if the thread is waiting for a lock at Point 2, there is a high possibility that the thread is causing the deadlock.

Compare the addresses of the locked objects to detect the deadlock for a thread applicable to Point 2 and Point 3.

In the example, Thread-3 has acquired the <02A328C8> lock and is waiting to acquire <02A328C0>.

On the other hand, Thread-1 has acquired the <02A328C0> lock and is waiting to acquire <02A328C8>. This shows that Thread-3 and Thread-1 are in a deadlock.

Tip

If a deadlock is suspected in the checks until now, request the developer to perform the check.

If a deadlock is not suspected, go to step 7.

7. Improve the business application. Remove redundant processing

   Based on the results of checks on the PRF trace and the thread dump, check and take action if you suspect delays in the business application.

Tip

If the problem is resolved, the troubleshooting process ends at this point.

If the problem is not resolved and if the CPU usage is high, go to step 8.

If the problem is not resolved and if the CPU usage is low, request the helpdesk to check, based on the purchase agreement.

8. Reduce the parameters with concurrently executing threads and control the number of concurrently executing processing
   The pending requests might accumulate, but you must wait for some time for the processing.

9. Upgrade the machine CPU
   Note the additional middleware license costs when you upgrade the CPU.

10. Add more machines and distribute the load of the transactions
    Note the additional hardware and software license costs when you add machines.

> **Tip**
>
> If the problem is resolved, the troubleshooting process is complete.
>
> If the problem is not resolved, request the helpdesk to check, based on the purchase agreement.

7 Performance Analysis by Using Trace Based Performance Analysis

The trace based performance analysis is the functionality to collect the performance analysis information (trace information) output by each functionality of Application Server when processing the requests from the client, and the performance analysis information (trace information) output by the processing of applications.

You can analyze the system and application performance based on this information. This chapter describes methods for analyzing the performance of the system and applications using the trace based performance analysis. For details on the collection of performance analysis information by the trace based performance analysis (trace collection point) and acquisition range of the information (PRF trace collection level), see the chapter *8. Trace Collection Points and PRF Trace Collection Levels of the Trace Based Performance Analysis*.

7.1 Organization of this chapter

By using the trace based performance analysis, you can monitor the operational status of each software type that configures the system and the trace information output by each server during request processing, thus you can check the processing performance of the whole system. If you monitor the processing performance, you can determine whether the study of an Application Server bottleneck and implementation of performance tuning is required. Furthermore, by analyzing the trace information output by the processing of the applications, you can check and compare the application performance. If you analyze the performance, you can check for application bottlenecks and determine whether the application requires improvement.

The following table describes the organization of this chapter.

Table 7–1: Organization of this chapter (Trace based Performance Analysis)

Category	Title	Reference location
Explanation	Overview of the trace based performance analysis	*7.2*
	Collecting the trace based performance analysis file by using Management Server	*7.3*
Implementation	Implementation for collection of root application information of trace based performance analysis	*7.4*
Setting	Settings of execution environment	*7.5*
Operation	Logs output when the user-extended trace based performance analysis is executed	*7.6*
	Analysis operation of the processing performance by using the trace based performance analysis file	*7.7*
Notes	Notes on using the user-extended trace based performance analysis	*7.8*

7.2 Overview of the trace based performance analysis

The trace based performance analysis functionality uses the performance analysis information output by the Application Server functionality and the application processing to analyze the performance of Application Server and application processing.

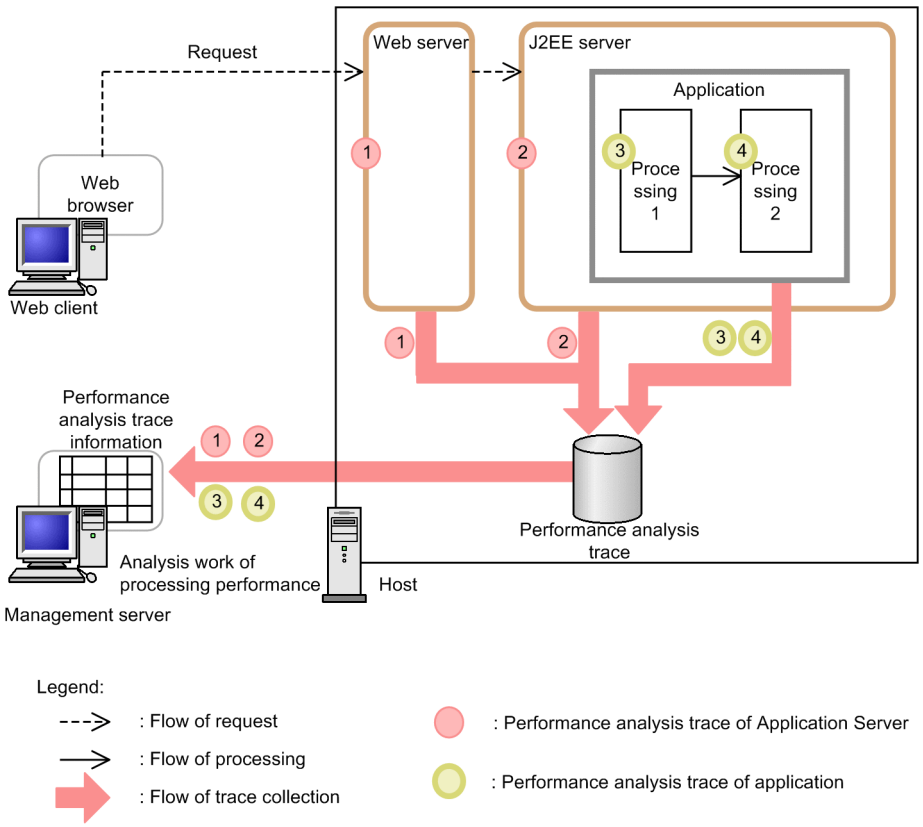
The trace based performance analysis functionality includes the following types based on the performance analysis information to be used:

- Trace based performance analysis of Application Server
- Trace based performance analysis of applications

Note that in both the cases you use the component software Cosminexus Performance Tracer to analyze the processing performance.

The following figure gives an overview of the trace based performance analysis.

Figure 7–1: Overview of the trace based performance analysis



Points 1 and 2 in the figure indicate the trace based performance analysis of Application Server, and points 3 and 4 indicate the trace based performance analysis of applications. You can collectively analyze both the trace based performance analysis as the trace based performance analysis information.

7.2.1 Overview of the trace based performance analysis of Application Server

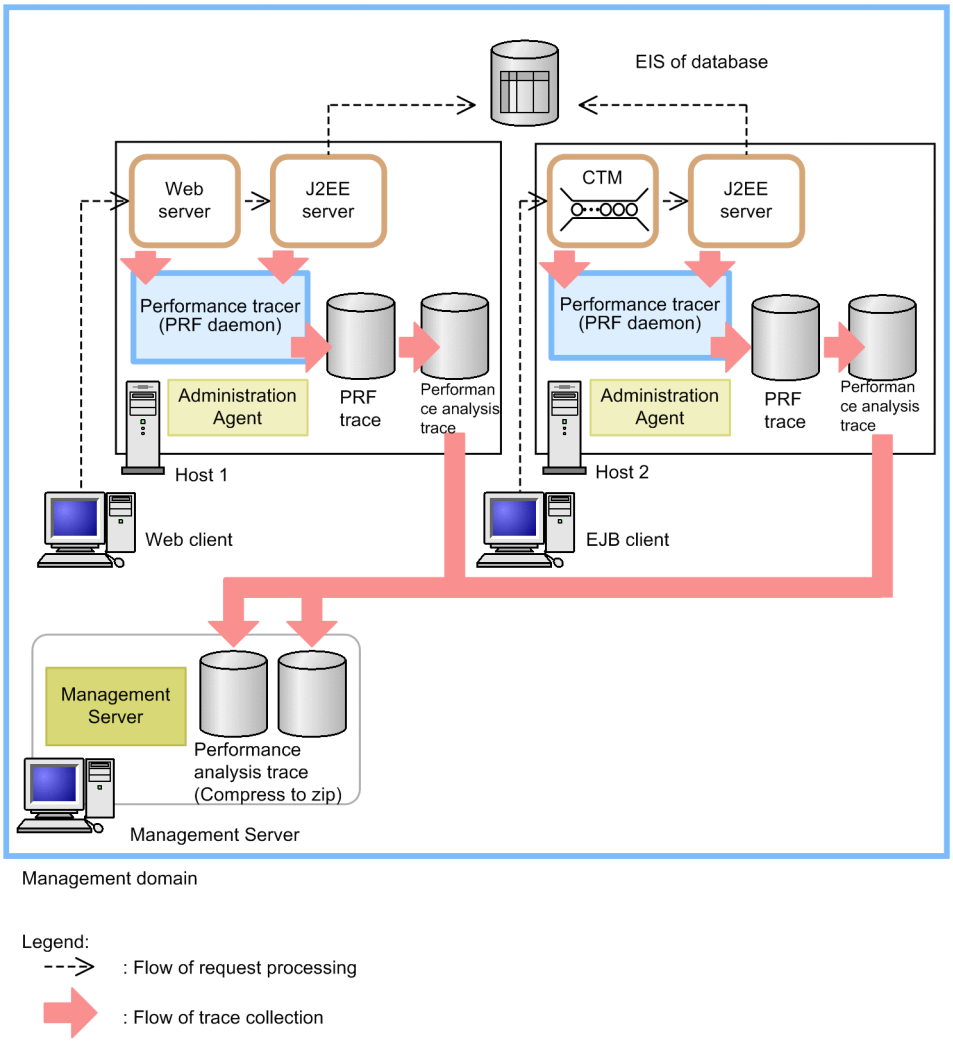
The trace based performance analysis of Application Server is a functionality that analyzes the Application Server performance by using the performance analysis information (trace information) output by the Application Server functionality in the processes that handle the requests from the client, and the information used to determine the session life cycle (hereafter, the trace based performance analysis of Application Server will be referred to as *trace based performance analysis*). This analysis enables you to analyze the Application Server bottlenecks, to improve the efficiency of troubleshooting by checking the extent to which request processing has been attained when an error occurs, and to understand the session and global session information life cycles.

(1) Collecting the trace information of the trace based performance analysis

The trace information of the trace based performance analysis collects the performance analysis information output in a series of processing of requests until the client reaches the components of the EIS such as a database, and until the processing results are returned to the client.

The following figure shows an overview of the trace information collection by the trace based performance analysis.

Figure 7–2: Overview of trace information collection of trace based performance analysis



When a request is sent from the Web client or the EJB client, the Web server, J2EE server, and CTM output the trace information to the buffer at determined processing points. If constant output information is collected, the trace is output in the trace file (*PRFTrace file*) depending on the *performance tracer* (**PRF daemon**). The point at which the trace is output is called the *trace get point*. You can set the trace collection levels (standard or detailed) in the performance tracer. The trace collection levels set in the performance tracer are called PRF trace collection levels.

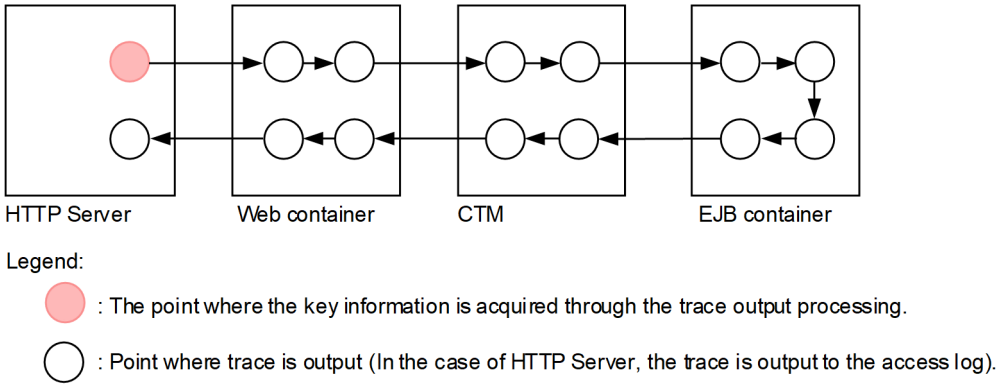
When you use Management Server for operations, you can collect the trace based performance analysis file by editing the PRF trace file in the text format. On the basis of the collected trace based performance analysis file, the operations administrator can perform the entire performance and bottleneck analysis in the management domain. For details on how to collect the trace based performance analysis file and the information that is output, see *7.3 Collecting the trace based performance analysis file by using Management Server*.

(2) Working of Trace based performance analysis

You can use the trace based performance analysis to collect the trace information across multiple nodes and processes in an event within the system. This helps to trace the process in a sequence of processes in which there is a bottleneck.

To obtain the trace in an event, set a uniform key for the sequence of processes of an event in the trace based performance analysis. The information of the key is added to the trace that is output at the trace acquisition point in an event. The sequence of processes can be traced in this way.

Figure 7–3: Overview of trace output depending on the trace based performance analysis



The EJB container 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 describes the execution environment of the application and the applicable function layer. In the trace based performance analysis, trace information is output at the entrance and the exit of the following function layer.

Table 7–2: 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 the batch application
CTM	Y	--
Web container	Y	--
EJB container	Y	--
Timer Service	Y	--
JNDI	Y	Y

Function layer	Execution environment of the application	
	Execution environment of the J2EE application	Execution environment of the batch application
JTA	Y	Y
JCA container	Y	Y
DB Connector	Y	Y
RMI (communication processing)#1	Y	Y
OTS	Y	Y
Standard output, standard error output, and user log	Y	Y
DI	Y	--
Batch application execution functionality	--	Y#2
JPA	Y	--
TP1 inbound integrated function	Y	--
Cosminexus JMS Provider	Y	--
JavaMail	Y	--
CDI	Y	--
JSF 2.2	Y	--
JAX-RS	Y	--
Java Batch	Y	--

Legend:

Y: Applicable

--: Not applicable

#1

You can control the collection of the layer information for the function layer of RMI (communication processing). In such cases, you have to set `control` in the trace collection level. For details on the setting methods, see *cprfstart (start PRF daemon)* in the *uCosminexus Application Server Command Reference Guide* or *cprflevel (display or change the PRF trace collection level)* in the *uCosminexus Application Server Command Reference Guide*.

#2

The trace information is output immediately before (immediately before invoking main method) executing the batch application and immediately after terminating the batch application. The trace information is not output at the execution of `cjexecjob` and `cjkilljob` command.

In addition to these function layers, the trace based performance analysis outputs the trace for the start process and stop process of the J2EE server, as well as when transaction timeout occurs, and when a session is generated or cancelled.

The trace information contains information such as the process ID used to get the trace information, eventID that indicates the get point, and the IP address of the client application that gets the trace get date or the trace information.

Reference note

In addition to these function layers, you can also obtain the PRF trace in the following function layers with the configuration software and related programs of Application Server:

- Cosminexus Web Services - Base
- uCosminexus TP1 Connector
- TP1/Client/J
- TP1/MQ Access
- Cosminexus RM
- HCSC server
- HCSC server (Object Access adapter)
- Service Coordinator Interactive Workflow
- HCSC server (file adapter)
- HCSC server (Message Queue adapter)
- HCSC server (FTP adapter)
- JAX-WS Engine
- Elastic Application Data store

The key information of the trace information consists of the following elements:

Configuration of the key information

- Process ID used to obtain the key information
- IP address of the host that invoked the process of acquiring the key information
- Communication number allocated to the I/O process (PRF daemon) of the PRF trace
  If the PRF daemon is not running, the time is returned as the communication number. It is important, however, to ensure that the PRF daemon is running, because the communication number may not be unique.

The following two types of key information are added to the PRF trace:

- Root application information
  This is the information obtained during the process that is executed first in the sequence of the processes in every event.
  - In the case of J2EE application
    It is the information obtained in the HTTP Server, the NIO HTTP server, or the EJB client.
  - In the case of Batch application
    It is the information obtained immediately before executing the batch application.
- Client application information
  In case of J2EE application, this is the information that is set for each processing that invokes the following Enterprise Beans:
  - Invoking the EJB container from the Web container
  - Invoking the EJB container from the EJB client
  - Invoking the EJB container from the EJB container

  For batch applications, it is the information set immediately before executing batch application.

(3) Configuration of the trace based performance analysis

The trace based performance analysis consists of the following programs:

- PRF trace output library

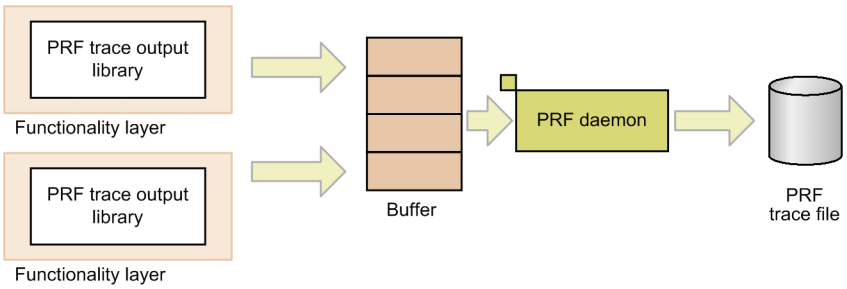
  This program is embedded in each function layer of Application Server. The PRF trace output by each function layer is output in the buffer that is created in the shared memory.

- PRF daemon

  This is the I/O process to output the PRF trace that is output in the buffer to a file, after a certain amount of PRF trace is collected. At least one PRF daemon is invoked on each host that obtains the PRF trace. We recommend that you deploy one PRF daemon on one host.

The following figure shows the relationship between the PRF trace output library and the PRF daemon.

Figure 7–4: Relation between PRF trace output library and PRF daemon



The buffer area output by trace is created when the PRF daemon starts, depending on the PRF trace output library. The buffer area is created in the common memory. Any buffer area that was created in the previous invocations of the PRF daemon that is still remaining is re-used. The buffer area is not deleted when the PRF daemon that was invoked previously ends abnormally.

When the PRF daemon ends normally, the buffer data in the buffer area is output to a PRF trace file and the buffer area is deleted.

When the buffer area is insufficient, the message of KFCT26999-W might output and the PRF trace might not output at all. Therefore, you must tune the buffer size when the message is output.

(4) Troubleshooting by acquiring the Trace Information

The use of trace information in troubleshooting errors is explained below.

You can output the information in the trace based performance analysis and use it for troubleshooting as follows:

- If a transaction in a J2EE application times out or if a timeout occurs when receiving a response on the reverse proxy of the HTTP Server, you can use the root application information that is output in the trace based performance analysis to identify the transaction or request that timed out.
- If a failure occurs while connecting to the database, you can use the connection ID that is output in the trace based performance analysis to identify the connection in which the error occurred.

7.2.2 Overview of the trace based performance analysis of applications

The trace based performance analysis of applications is a functionality that analyzes the application processing performance by using the performance analysis information (trace information) output when the application processing

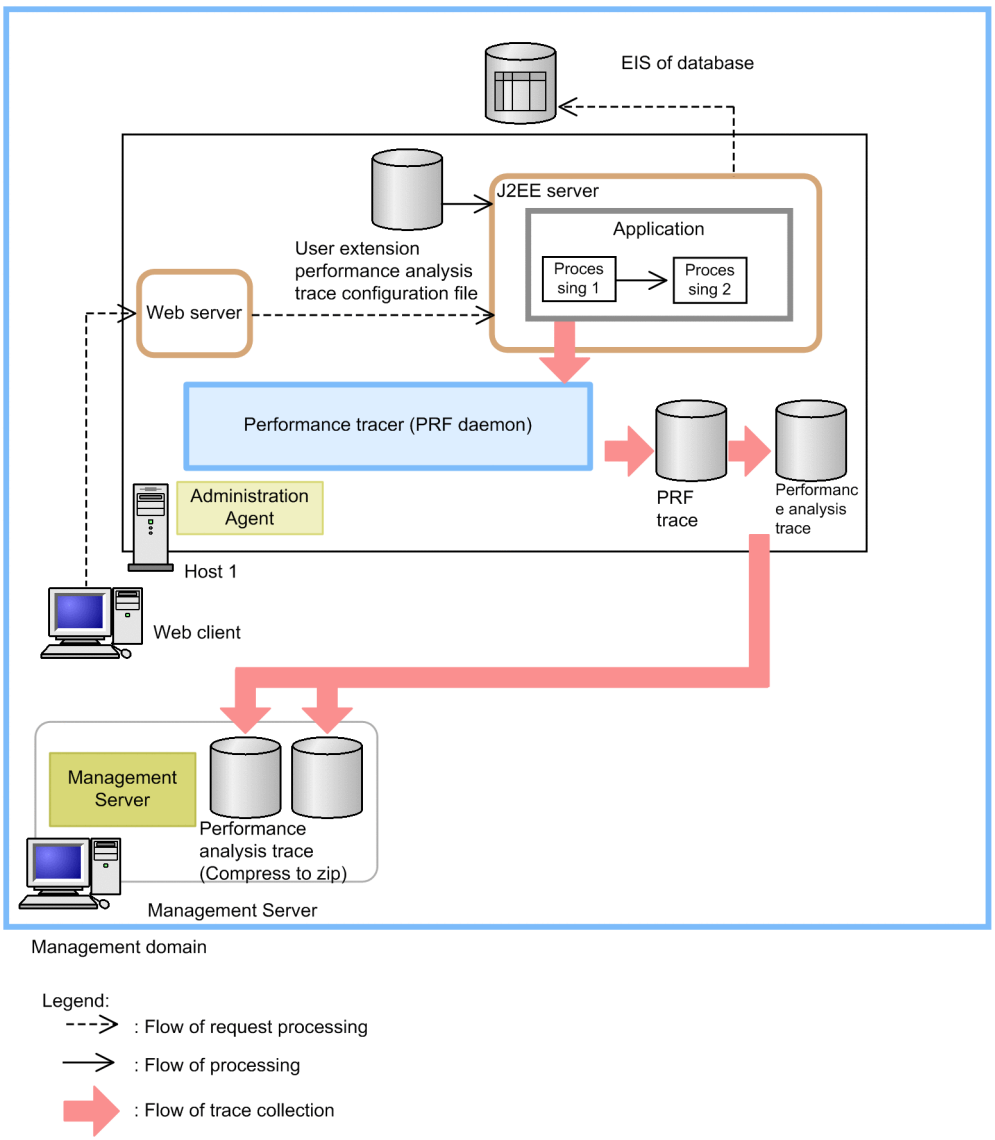
to be traced is executed in the processes from the start up to the termination of an application (hereafter, the trace based performance analysis of applications is referred to as the *user-extended trace based performance analysis*). The user-extended trace based performance analysis presumes the use of the trace based performance analysis of the Application Server. This analysis enables you to identify the processing bottlenecks and to improve the efficiency of troubleshooting by checking the extent to which the processing has been attained when an error occurs. Furthermore, you specify the processing for which you want to acquire the performance analysis information in the configuration file for the user-extended trace based performance analysis. You can efficiently check and compare the application performance because the application need not acquire the performance analysis information.

(1) Collecting the trace information of the user-extended trace based performance analysis

The trace information of the user-extended trace based performance analysis collects the performance analysis information that is output when the processing specified in the configuration file for the user-extended trace based performance analysis is executed.

The following figure gives an overview of the trace information collection for the user-extended trace based performance analysis.

Figure 7–5: Overview of the trace information collection for the user-extended trace based performance analysis



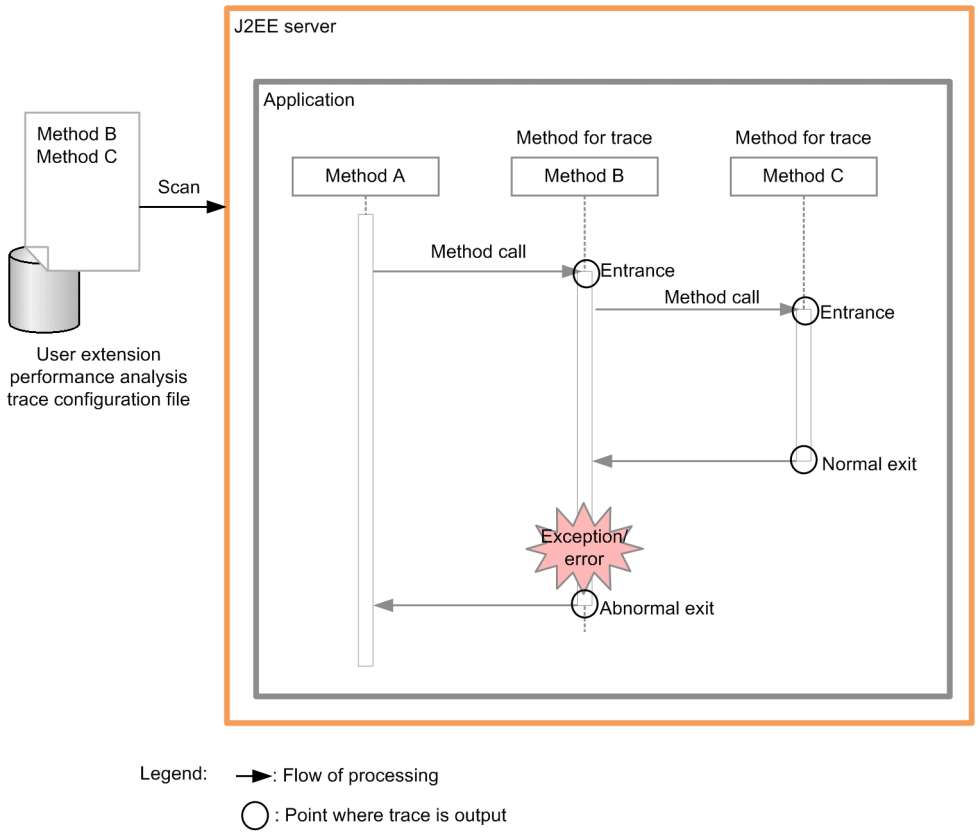
When you start the J2EE server, the configuration file for the user-extended trace based performance analysis is read. When the processing (trace collection point) specified in the configuration file for the user-extended trace based performance analysis is invoked, the trace information is output to a buffer. When a certain amount of information is collected, the performance tracer (PRF daemon) outputs the information to a trace file (PRF trace file).

When you use Management Server for operations, you can collect the trace based performance analysis file by editing the PRF trace file in the text format. On the basis of the collected trace based performance analysis file, the operations administrator can perform the entire performance and bottleneck analysis in the management domain. For details on how to collect the trace based performance analysis file and the information that is output, see *7.3 Collecting the trace based performance analysis file by using Management Server*.

(2) Working of the user-extended trace based performance analysis

In the user-extended trace based performance analysis, you specify the names of the methods that will be used to acquire the trace information, in the configuration file for the user-extended trace based performance analysis. The following figure gives an overview of trace output for the specified methods.

Figure 7–6: Overview of trace output by the user-extended trace based performance analysis



If you enable the user-extended trace based performance analysis, the user-extended trace based performance analysis reads the configuration file for the user-extended trace based performance analysis. If the application is executed and if the methods (methods to be traced) specified in the configuration file for the user-extended trace based performance analysis are invoked, the trace information is output to the following locations:

- Method entry

  Trace information immediately after the method is started.
- Normal exit of method

  Trace information just before the method terminates normally.
- Abnormal exit of method

  Trace information immediately after an exception or error occurs in the method. However, the exceptions or errors thrown at the method invocation source are excluded.

(3) Configuration of the user-extended trace based performance analysis

The user-extended trace based performance analysis is configured from the following elements.

Note that the user-extended trace based performance analysis uses the class load hook processing of the instrumentation functionality and rewrites the applications to be traced in order to output the trace based performance analysis.

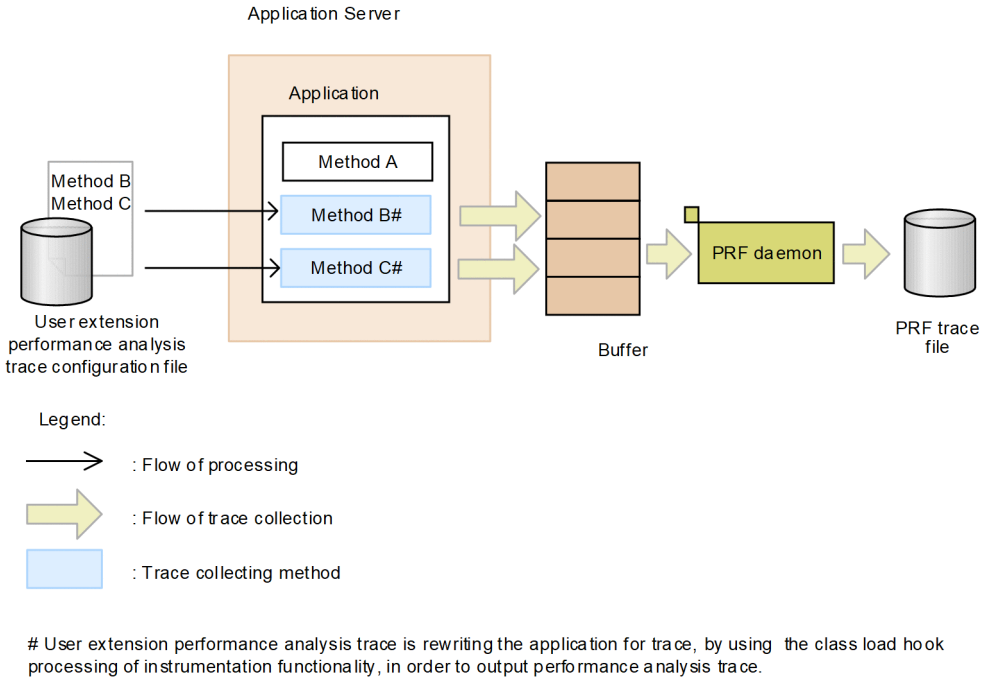
- **Configuration file for the user-extended trace based performance analysis**

  You set up the information about the methods to be traced by the user-extended trace based performance analysis by using the configuration file for the user-extended trace based performance analysis. For details on the contents of the configuration file for the user-extended trace based performance analysis, see *7.5.3 Settings for the methods to be traced by the user-extended trace based performance analysis*.
- **PRF daemon**

This is the I/O process to output the PRF trace that is output in the buffer to a file, after a certain amount of PRF trace is collected. At least one PRF daemon is invoked on each host that acquires the PRF trace. We recommend that you allocate one PRF daemon to one host.

The following figure shows the relationship between the configuration file for the user-extended trace based performance analysis and the PRF daemon.

Figure 7–7: Relationship between the configuration file for the user-extended trace based performance analysis and the PRF daemon



User extension performance analysis trace is rewriting the application for trace, by using the class load hook processing of instrumentation functionality, in order to output performance analysis trace.

7.3 Collecting the trace based performance analysis file by using Management Server

When you use Management Server for operations, you can collect the contents of the trace file output to each host on the management server in a batch by using the management commands (`mngsvrutil`) of Management Server. Note that the PRF trace file output by the PRF daemon is a binary file. Management Server collects the file obtained by issuing instructions to Administration Agent, editing the PRF trace file in a text-format (CSV) file, and then compressing it (ZIP format). The PRF trace output by the hosts in the management domain can be collected on Management Server.

Note that the trace file edited in the text format is called the *trace based performance analysis file*.

For details on the management commands, see *mngsvrutil (Management Server management command)* in the *uCosminexus Application Server Command Reference Guide*.

This section describes about the collection methods, output destination and the information of the trace based performance analysis file.

7.3.1 How to collect a trace based performance analysis file

When collecting the trace based performance analysis file, you have to use the management command (`mngsvrutil`) of the Management Server. Specify the subcommand `collect` to the `mngsvrutil` command. In the case of collecting a trace based performance analysis file, select the target for collecting the trace information. The following are the targets for collecting the trace information when using management command:

- All the performance tracer that operate on the host in the management domain.
- All the performance tracer that operate on a specific host
- An identified performance tracer

The execution format and example of each case is described below:

When all the performance tracer that operate on the host in the management domain is the target

Execution format

```
mngsvrutil-m Host-name-of-Management-Server [:Port-Number]-u Management
-user-ID -p Management-password collect allPrfTraces
```

Execution example

```
mngsvrutil -m mnghost -u user01 -p pw1 collect allPrfTraces
```

When all the performance tracer that operate on specific host is the target

Execution format

```
mngsvrutil-m Host-name-of-Management-Server -[:Port-Number]-u Managemen
t-user-ID -p Management-password -t Host-name -k host collect prfTrace
```

Execution example

```
mngsvrutil -m mnghost -u user01 -p pw1 -t host01 -k host collect prfTra
ce
```

When an identified performance tracer is the target

Execution format

```
mngsvrutil -m Host-name-of-Management-Server -u[:Port-Number] Managemen
t-user-ID -p Management-password -t Logical-performance-trace-user -k l
ogicalServer collect prfTrace
```

Execution example

```
mngsvrutil -m mnghost -u user01 -p pw1 -t ID01 -k logicalServer collec
t prfTrace
```

7.3.2 Output destination of trace based performance analysis files

- In Windows
  log-output-directory-of-the-Manager\prf
- In UNIX
  log-output-directory-of-the-Manager/prf

Note that a trace based performance analysis file is output based on the trace information to be collected by the following file names.

Table 7–3: File names of trace based performance analysis files

Target for collecting trace information	File name
All the performance tracer that operate on the host in the management domain.	*Management-domain-name* - *date-and-time*#1 .zip
All the performance tracer that operate on a specific host.	*Host-name* - *date-and-time*#1 .zip
An identified performance tracer	*Logical-server-name*#2 - *date-and-time*#1 .zip

#1
The date when the trace based performance analysis file is collected is displayed.

#2
The name of the identified performance tracer is displayed.

7.3.3 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 describes the information output to trace based performance analysis file (CSV format) from the performance tracer. The collection items differ in CTM and the other function layers. Note that the output items are different depending on each collection point such as the existence of the additional information. For details on the items output for each collection point, see the chapter *8. Trace Collection Points and PRF Trace Collection Levels of the Trace Based Performance Analysis*.

Table 7–4: Information output to the trace based performance analysis file (for the trace based performance analysis)

Trace information header	Description	Range of values
PRF	Record status of that process (normal and abnormal).	Either of the following is output: Normal: `Rec` Abnormal: `ErrRec`
Process	Process ID of the process that acquired the trace information.	A decimal number of ten digits is output.
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.
Trace	Trace sequence number of the thread in the process that acquired the trace information.	A decimal number of ten digits is output.
ProcessName	Process name	A string[#2] displaying the process up to 32 characters is output.
Event	Event ID showing the trace collection point.	A hexadecimal number of six digits (include 0x also in six digits) is output[#3].
Date	Date when the trace information is acquired.	The date is output in the yyyy/mm/dd format. yyyy: Year mm: Month dd: Day
Time	Time when the trace information is acquired (Hours: Minutes: Seconds).	The time is output in the hh:mm:ss format. hh: Hours mm: Minutes ss: Seconds
Time(msec/usec/nsec)	Time when the trace information is acquired (milli seconds/micro seconds/nanoseconds).	The time is output in the ms/us/ns format. ms: Milli seconds us: micro seconds ns: nanoseconds
Rc	Return code.	A hexadecimal number of sixteen digits (include "0x" also in sixteen digits) is output. Normal:0 Abnormal:1 (or other than 0)
ClientAP IP[#4]	IP address of the client application that acquired the trace information.	The IP address is output in the aaa.bbb.ccc.ddd format.
ClientAP PID[#4]	Process ID of the client application that acquired the trace information.	A decimal number of ten digits is output.
ClientAP CommNo.[#4]	Communication number of the client application that acquired the trace information.	A hexadecimal number of 18 digits (include 0x also in 18 digits) is output.
RootAP IP[#5]	IP address of a root application that acquired the trace information.	The IP address is output in the aaa.bbb.ccc.ddd format.
RootAP PID[#5]	Process ID of a root application that acquired the trace information.	A decimal number of ten digits is output.
RootAP CommNo.[#5]	The communication number of the root application that acquired the trace information.	A hexadecimal number of eighteen digits (include "0x" also in eighteen digits) is output.
SendSCD IP[#5]	IP address of the request source CTM.	The IP address is output in the aaa.bbb.ccc.ddd format.

Trace information header	Description	Range of values
SendSCD PID[#6]	Process ID of the request source CTM.	A decimal number of ten digits is output.
ReceiveSCD IP[#6]	IP address of the request destination CTM	The IP address is output in the aaa.bbb.ccc.ddd format.
ReceiveSCD PID[#6]	Process ID of the request destination CTM.	The decimal number of ten digits is output.
INT	Interface name for each collection point.	A string[#7] up to 33 characters is output.
OPR	Operation information related to the collection point.	A string[#7] up to 33 characters is output.
LookupName[#6]	Lookup name.	A string[#7] up to 33 characters is output.
OPT[#8]	Additional information for each collection point.	A hexadecimal number string up to 514 characters is output.
ASCII	ASCII character output of the additional information for each collection point.	The OPT contents is output as ASCII character strings within 514 characters.

#1

There are cases when the hash value of a thread is not output in the trace information acquired by CTM.

#2

The process name is decided as described below:

- **In the case of EJB client applications**

  The name specified in the system property `ejbserver.server.prf.processName` of the EJB client application.

  If this system property is not specified or null character is specified in this property, the process name will be EJBClient.

- **In the case of J2EE server, batch server, or Web container server**

  The server name will be the process name.

- **In case of CTM**

  It is the name of each CTM process.

#3

An event ID is allocated to each trace collection point of the function layer. For details, see *8. Trace Collection Points and PRF Trace Collection Levels of the Trace Based Performance Analysis*.

#4

This is a component of client application information.

#5

This is a component of root application information. At the trace point that is output to the Web container, there are cases when the IP address, process ID, and communication number of the root application that acquired the trace information are output as 0.0.0. 0/0/0x0000000000000000. For details, see *7.7.7 Investigating the Log Using the Root Application Information*.

Note that when you operate an HTTP server in a dual stack environment of IPv4 and IPv6, the IPv4 address is output to the process ID of the root application. Also, if the client address is IPv6 and is 33 characters or more, edit and output the trace information event IDs 0x8000 and 0x8100.

- For event ID 0x8000: *32-characters-from-the-beginning* + *
- For event ID 0x8100: *16-characters-from-the-beginning* + * + *16-characters-from-behind*

In such cases, if both event IDs 0x8000 and 0x8100 are referenced, you can obtain the entire client address.

#6

The information that is output only for CTM. "****" is displayed in the layer other than CTM. In the systems executing batch application, the schedule group name is displayed.

#7

If the interface name, operation name, and lookup name exceeds 32 characters, the name is changed to 33 characters and is output with one of the following methods. For details, see *8. Trace Collection Points and PRF Trace Collection Levels of the Trace Based Performance Analysis*.

First 32 characters + *

First 16 characters + * + last 16 characters

* + last 32 characters

#8

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.

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.

7.3.4 Output information of the trace based performance analysis file (for the user-extended trace based performance analysis)

The user-extended trace based performance analysis collects the trace information output by the application processing.

For details on the items output in the trace based performance analysis, see *7.3.3 Output information of the trace based performance analysis file (for the trace based performance analysis)*.

Note that the items output at each collection point, such as presence or absence of the operation information, are different. For details on the items output at the collection points, see *8.23 Trace collection points of an application*.

7.4 Implementation for collection of root application information of trace based performance analysis

This section describes an overview and implementation methods of the functionality of acquiring root application information of the trace based performance analysis.

root application information is the information that is acquired by the HTTP Server, the NIO HTTP server, or the EJB client among the information acquired by trace based performance analysis functionality. You can implement the functionality of acquiring the character string expression of the root application information in the J2EE application and batch application using API. When implemented, it would be helpful in troubleshooting when the trouble occurs since the functionality of acquiring the character string expression of the root application information can be compared with the trace based performance analysis file at the optional timing when the acquired character string expression is recorded in log file.

Use `CprfTrace` class for implementing the functionality of acquiring root application information of trace based performance analysis. To use `CprfTrace` class, specify the following path in the class path and compile.

- In Windows
  Cosminexus-installation-directory`\CC\lib\ejbserver.jar`
- In Unix
  `/opt/Cosminexus/CC/lib/ejbserver.jar`

For details on the investigation of using the root application information, see the subsection *7.7.7 Investigating the Log Using the Root Application Information*.

7.5 Settings of execution environment

This section describes the settings required for using the trace based performance analysis and user-extended trace based performance analysis.

Different items will be set up for the trace based performance analysis and the user-extended trace based performance analysis. The following table describes the settings required for the trace information to be acquired.

Table 7–5: Settings required for the trace information to be acquired

Settings	Trace information to be acquired		Reference location
	Trace based performance analysis	User-extended trace based performance analysis	
Settings for using the trace based performance analysis	Y	Y	*7.5.1*
Settings for using the user-extended trace based performance analysis	--	Y	*7.5.2*
Settings for the methods to be traced by the user-extended trace based performance analysis	--	Y	*7.5.3*

Legend:

Y: Settings are required.

--: Settings are not required.

7.5.1 Settings for using the trace based performance analysis

The following settings are required when using the trace based performance analysis:

- Management Server
- Performance tracer

Important note

Note the following points when you specify the environment variables in AIX:

- With the Performance Tracer execution environment, set up `early` in the environment variable `PSALLOC`. If this value is not set and if a memory shortage occurs, the operations might not function correctly.
- Set up `early` in the environment variable `PSALLOC` specifying the early paging space allocation in AIX. For the points to be considered when you estimate the paging space for the early paging space allocation, see the AIX manual *System Management Concepts: Operation Systems and Devices*.
- With the Performance Tracer execution environment, set up `true` in the environment variable `NODISCLAIM`. If you specify `early` in the environment variable `PSALLOC`, and do not specify `true` in the environment variable `NODISCLAIM`, the responses, throughput, and CPU usage might deteriorate greatly.
- To extend the user data area and shared memory area to be used with Performance Tracer, set up `MAXDATA=0x40000000` in the environment variable `LDR_CNTRL`. Allocate 1 gigabyte memory.
- With the Performance Tracer execution environment, set up `ON` in the environment variable `EXTSHM`. If no value is set up, sometimes the shared memory cannot be referenced.

(1) Management Server settings

The setting of the Management Server is executed by `mserver.properties` (Management Server environment setting file). The setting parameters are as follows:

- `com.cosminexus.manager.tracelog.size`

  Specify the number of files of the trace based performance analysis file collected by Management Server.

For details on `mserver.properties` and the key, see *8.2.6 mserver.properties (Management Server environment settings file)* in the *uCosminexus Application Server Definition Reference Guide*. For details on the trace based performance analysis file, see the section *7.3 Collecting the trace based performance analysis file by using Management Server*.

(2) Setting of performance trace

The setting of the performance tracer (PRF daemon) is implemented by the Easy Setup definition file. Specify the definition of trace based performance analysis in the `<configuration>` tab of the logical performance trace (performance-tracer) of the Easy Setup definition file.

The following table describes the definition of the trace based performance analysis in the Easy Setup definition file.

Table 7–6: Definition of trace based performance analysis in the Easy Setup definition file

Parameter to specify	Setting contents
`PrfTraceLevel`	The functionality (Web server, J2EE server, CTM) of Application Server outputs in the buffer, specifies trace collection level of the performance trace.
`PrfTraceCount`	Specifies the number of files of the PRF trace of performance trace.
`PrfTraceFileSize`	Specifies the file size of the performance tracer. Set the value consisting the relation of `PrfTraceFileSize` ≥ PrfTraceBufferSize .
`PrfTraceBufferSize`	Specifies the buffer size of the performance tracer. Set the value consisting the relation of `PrfTraceFileSize` ≥ PrfTraceBufferSize .

Note:

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

7.5.2 Settings for using the user-extended trace based performance analysis

To use the user-extended trace based performance analysis, you must specify the following settings before you start the server:

- J2EE server
- Batch server
- EJB client application
- Settings for the methods to be traced by the user-extended trace based performance analysis

(1) J2EE server settings

Implement the J2EE server settings using the Easy Setup definition file.

Define the user-extended trace based performance analysis in the JavaVM startup parameter (`add.jvm.arg`) in the `<configuration>` tag of the logical J2EE server (`j2ee-server`) in the Easy Setup definition file.

The following table describes the definition of the user-extended trace based performance analysis in the Easy Setup definition file.

Table 7–7: Definition of the user-extended trace based performance analysis in the Easy Setup definition file

Item	Specified parameter	Settings
Settings to enable the user-extended trace based performance analysis	`jvm.userprf.Enable`	To use the user-extended trace based performance analysis, specify the settings to enable.
Settings to specify the configuration file for the user-extended trace based performance analysis	`jvm.userprf.File`[#1]	Specify the file path of the user-extended trace based performance analysis configuration file to be used for the user-extended trace based performance analysis.
Settings related to application rewriting	`jvm.userprf.Limit`	Specify the maximum number of methods to be rewritten from the methods specified in the configuration file for the user-extended trace based performance analysis. Among the methods specified in the configuration file for the user-extended trace based performance analysis, the methods exceeding this value are not traced.
	`jvm.userprf.Trace`	Specify whether log will be output when the class files specified in the configuration file for the user-extended trace based performance analysis are successfully rewritten. Note that if the rewriting of the class files fails, the log is necessarily output.
Settings related to the output information of the user-extended trace based performance analysis	`jvm.userprf.ExtendedSetting`[#2]	Specify whether the trace target can be specified for packages or classes in addition to methods.
	`jvm.userprf.LineNumber`	Specify whether the line number of the line that the method executes last in the case of a normal exit, will be output to the trace information.
	`jvm.userprf.ThrowableName`	Specify whether the exception or error class name will be output to the trace information.
	`jvm.userprf.ThrowableNameEditMethod`	Specify how the name will be edited when the exception or error class name exceeds the limit of 32 characters.
	`jvm.userprf.LogLevel`	Specify the trace output level of the user-extended trace based performance analysis.

#1

The default values are as follows:

In Windows

JDK-installation-directory`\usrconf\userprf.cfg`

In UNIX
`/opt/Cosminexus/jdk/usrconf/userprf.cfg`

#2

The amount of methods to be traced increases due to the specification format of the user-extended trace based performance analysis configuration file permitted by setting up the `jvm.userprf.ExtendedSetting` property. Therefore, note that the amount of trace that is output increases and affects the system performance. For details on the notes for using the user-extended trace based performance analysis, see *7.8 Notes on using the user-extended trace based performance analysis*.

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*.

An example of definition is as follows:

```
 ...
<param>
<param-name>add.jvm.arg</param-name>
<param-value>-Djvm.userprf.Enable=true</param-value>
<param-value>-Djvm.userprf.File=Cosminexus-installation-directory/CC/server/
usrconf/ejb/real-server-name/userprf.cfg</param-value>
<param-value>-Djvm.userprf.LogLevel=class</param-value>
</param>
 ...
```

Reference note

You can also specify the J2EE server settings in the Start Parameter Settings window (defining the logical J2EE server) of the management portal. For details on how to specify the settings on the management portal, see *10.8.23 Startup parameter settings (J2EE server)* in the *uCosminexus Application Server Management Portal User Guide*.

(2) Batch server settings

Implement the batch server settings using the Easy Setup definition file.

Define the user-extended trace based performance analysis in the JavaVM startup parameter (`add.jvm.arg`) in the `<configuration>` tag of the logical J2EE server (`j2ee-server`) in the Easy Setup definition file. For details on the parameter values to be specified, see *(1) J2EE server settings*.

(3) EJB client application settings

Implement the settings for an EJB client application, which is running with the `cjclstartap` command, using the option definition file for EJB client applications (`usrconf.cfg`).

Define the user-extended trace based performance analysis in the JavaVM startup parameter (`add.jvm.arg`) of the option definition file for EJB client applications (`usrconf.cfg`). For details on the parameter values to be specified, see *(1) J2EE server settings*.

An example of definition in the option definition file for EJB client applications (`usrconf.cfg`) is as follows:

```
 ...
add.jvm.arg=-Djvm.userprf.Enable=true
add.jvm.arg=-Djvm.userprf.File=user-definition-file-storage-directory#/userpr
f.cfg
```

```
add.jvm.arg=-Djvm.userprf.LogLevel=class
 ...
```

#

The directory specified in the user definition file storage destination environment variable (CJCLUSRCONFDIR). If the user definition file storage destination environment variable is not set, the current directory is referenced.

(4) Settings for the user-extended trace based performance analysis configuration file

To use the user-extended trace based performance analysis, you must create the configuration file for the user-extended trace based performance analysis and specify the settings for the methods to be traced.

Specify the settings for the methods to be traced in the UserPrfText parameter in the <configuration> tag of the logical J2EE server (J2EE-Server) in the Easy Setup definition file.

An example of definition is as follows:

```
 ...
<param>
<param-name>UserPrfText</param-name>
<param-value>
<![CDATA[
org.apache.struts.action.Action.execute(*), struts, true
Info.getInfo(*), struts, true
]]>
</param-value>
</param>
 ...
```

For details on how to create the configuration file for the user-extended trace based performance analysis, see *7.5.3 Settings for the methods to be traced by the user-extended trace based performance analysis*.

Reference note

You can also set up the configuration file for the user-extended trace based performance analysis in the Start Parameter Settings window (defining the logical J2EE server) of the management portal or a user optional file (file specified in the jvm.userprf.File property).

7.5.3 Settings for the methods to be traced by the user-extended trace based performance analysis

Use the configuration file for the user-extended trace based performance analysis to specify the settings for the methods to be traced by the user-extended trace based performance analysis. You can code the contents of the configuration file for the user-extended trace based performance analysis in one of the following files or window:

- Easy Setup definition file
- Start Parameter Settings window (defining the logical J2EE server) of the management portal
- Default file or any user-specified file (file specified in the jvm.userprf.File property)

(1) Format of description

The format for coding the configuration file for the user-extended trace based performance analysis is as follows:

```
specification-format#, identity-ID, subclass-flag [, [event-ID][,[trace-colle
ction-level]]] [# Comment]
```

#1

You can omit the items enclosed within [ ].

#2

To trace all the methods with matching class names and package names, set up `true` in the value of the `jvm.userprf.ExtendedSetting` property.

#

Specify *specification-format* in one of the following formats:

- To trace a method with matching method name and argument types
  package-name.*class-name*.*method-name* (type name of the method argument)
- To trace methods with matching method names
  package-name.*class-name*.*method-name*(*)
- To trace all the methods with matching class names
  package-name.*class-name*
- To trace all the methods with matching package names
  package-name.*

Example of coding format:

```
com.sample.Test.method(),TEST1,false,0xae02,A
com.sample.Test.method(),TEST1,false,0xae02
com.sample.Test.method(),TEST1,false,0xae02,
com.sample.Test.method(),TEST1,false,,A
com.sample.Test.method(),TEST1,false
com.sample.Test.method(),TEST1,false,
com.sample.Test.method(),TEST1,false,,
```

(2) Description items

The items coded in the configuration file for the user-extended trace based performance analysis are as follows:

Specification-format

Specify the method to be traced.

package-name

Specify the package name of the class or interface of the method to be traced.

class-name

Specify the class name or interface name of the method to be traced. If you specify the interface name instead of the class name, and if `true` is specified in the subclass flag, the method implementing that interface is traced.

method-name

Specify the method to be traced.

type-name-of-the-method-argument

Specify the argument type of the method to be traced using the fully qualified name.

identity-ID

Specify the character string for identifying the method to be traced.

The characters you can use are the ASCII characters from 0x21(!) to 0x7e(~). However, you cannot specify 0x22(") , 0x23(#), and 0x2c(,).

The identity ID is output to the PRF trace file. Note that up to 32 characters are output to the PRF trace file. The 33rd and subsequent characters are omitted and * (asterisk) is output as the 33rd character.

Important note

If a subclass exists in the class defining the method to be traced and if the method to be traced is not overridden in that subclass, the specified identity ID is output for the superclass method to be traced.

subclass-flag

Specify whether to trace the methods of the class that has an inheritance relationship with the classes or interfaces of the specified method, using `true` or `false`.

- If you specify `true`, the specified method and the methods overriding the specified method are traced.
- If you specify `false`, only the specified method is traced, and the methods overriding the specified method are not traced.

event-ID

Specify the point for acquiring the trace information (trace collection point) using a hexadecimal value (0xae02 to 0xae7e and 0xc000 to 0xcffe). The default value is `0xae00`.

This value is output in the trace information that is output at the method entry, and this value + 1 is output in the trace information that is output at the method exit.

trace-collection-level

Specify `A`, `B`, `C`, or their respective lower cases in the trace collection level of the methods to be traced. The default value is `A`.

- `A`: Standard level
- `B`: Advanced level
- `C`: Maintenance level

Reference note

The trace collection level is the same as the PRF trace collection level of the `cprflevel` command. All the methods to be traced reference the levels of the Java VM layers. For details on the PRF collection levels and layers, see *cprflevel (display or change the PRF trace collection level)* in the *uCosminexus Application Server Command Reference Guide*.

comment

A comment begins with # (hash mark). All the characters from # (hash mark) up to the end of the line are assumed to be a comment.

(3) Rules of description

The rules for coding the configuration file for the user-extended trace based performance analysis are as follows:

- You can use only single-byte characters in the code.
- A space character becomes a single-byte space character (`0x20`) or a tab character (`\t` or `0x09`). Note that the space characters are ignored when the configuration file for the user-extended trace based performance analysis is read.
- Code one method to be traced on one line.
- You can specify up to 2,048 characters in 1 line. This character count includes the spaces and comments.
- The end of the line is indicated by 1 or more continuous linefeed characters (`\n` or `0x0A`) or carriage return characters (`\r` or `0x0D`).
- If the description in the configuration file for the user-extended trace based performance analysis is incorrect, a message describing the incorrect contents is displayed. Also, if invalid values are coded for the items, a message indicating incorrect contents is output, and the settings for that line are disabled.

  For details on the message, see *7.6 Logs output when the user-extended trace based performance analysis is executed*.
- With the user-extended trace based performance analysis, you cannot specify the JavaVM classes and Cosminexus classes in the methods to be traced. The following packages are applicable:
  - Classes beneath `java`
  - Classes beneath `javax`
  - Classes beneath `com.hitachi`
  - Classes beneath `JP.co.Hitachi`

  Therefore, when you want to specify a package name by specifying the `jvm.userprf.ExtendedSetting` property, code the package name such that only the application classes are included.
- You cannot specify the following methods as the methods to be traced:
  - Non-existent package name, class name, and method name
  - native method
  - abstract method
  - Classes in JavaVM and the methods of those classes

    (Example) Classes of packages beginning with `java` and `javax`
  - Classes specified in `-Xbootclasspath` and the methods of those classes
  - Classes in Cosminexus
- You can specify the following methods as the methods to be traced using the following code:
  - You can specify the constructor using the same method name as the class name, or `<init>`.

    (Example) Specify as follows for the constructor of the `MyMain` class:

    `MyMain.MyMain()` or `MyMain.<init>()`
  - If you specify a method with the same name as a non-constructor class name, whether a constructor or a method has been specified cannot be determined, therefore, the constructor and method are traced.
  - When specifying a method with variable length arguments, describe the variable length arguments as an array.

    (Example) Specification of a method with variable length argument

    Example of correct specification: `com.sample.Test.method(java.lang.String[])`

    Example of incorrect specification: `com.sample.Test.method(java.lang.String...)`
  - Specify the name of a nested class using '$' (dollar sign) as a delimiter instead of '. ' (period).

    (Example) Specification of a nested class

    Example of correct specification: `com.sample.Test$NestClass`

Example of incorrect specification: `com.sample.Test.NestClass`

- You can specify a non-generics and non-parameter class name (raw type).
  (Example) Specification of a non-generics class
  Example of correct specification: `com.sample.Test.method()`
  Example of incorrect specification: `com.sample.Test<java.lang.String,java.lang.Object>.method()`

- The methods to be traced differ when you specify a class name and when you specify an interface name in *class-name* in *specification-format*.
  The following table describes the methods to be traced based on the class or interface specification.

Table 7–8: Methods to be traced based on the class or interface specification

Class or interface	Method to be traced	
	When false is specified in the subclass flag	When true is specified in the subclass flag
Class	Method of the specified class	The methods of the specified class and the methods overriding these methods
Interface	None	The methods of the class#1 that directly implements the specified interface and the class#2 that indirectly implements the interface, and the methods overriding these methods

#1
This class implements the interface specified by using `implements` when the class is declared.

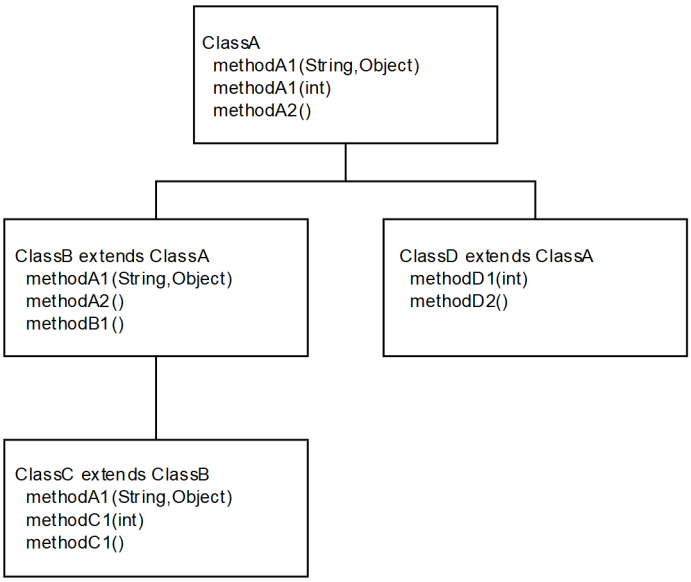
#2
This subclass of the class that directly implements the specified interface, or the class that directly implements the interface that inherits the specified interface.

(4) Example of description of the configuration file for the user-extended trace based performance analysis

This section describes the examples of coding of the configuration file for the user-extended trace based performance analysis for each method to be traced.

Note that the examples of code assume that the package name is `com.sample`, and that the application has the class structure shown in the following figure.

Figure 7–8: Example of the class structure of the application



(a) To trace a method with matching method name and argument types

The following is an example of coding the user-extended trace based performance analysis configuration file when you want to trace a method with a matching method name and argument types:

- **If the subclass flag is** `false`

```
com.sample.ClassA.methodA1(java.lang.String,java.lang.Object),1000,false
```

If the subclass flag is `false`, the coded method with a matching method name and argument types is traced.

Method to be traced

- `methodA1(String,Object)` of the `ClassA` class

The event ID is not set for the method to be traced, so if you invoke the method to be traced, the default value `0xae00` is output as the event ID at the method entry, `0xae01` is output at the method exit. Also, `1000` is set in the identity ID, so `1000` is output as the identity ID.

- **If the subclass flag is** `true`

```
com.sample.ClassA.methodA2(),2000,true
```

If the subclass flag is `true`, in addition to the method in which the method name and argument types match the code, the methods overriding the coded method are also traced.

Methods to be traced

- `methodA2()` of the `ClassA` class
- `methodA2()` of the `ClassB` class that overrides `methodA2()` of the `ClassA` class

The event ID is not set for the method to be traced, so if you invoke the method to be traced, the default value `0xae00` is output as the event ID at the method entry, `0xae01` is output at the method exit. Also, `2000` is set in the identity ID, so `2000` is output as the identity ID for all the methods to be traced.

(b) To trace methods with matching method names

The following is an example of coding the user-extended trace based performance analysis configuration file when you want to trace methods with matching method names:

- **If the subclass flag is** `false`

```
com.sample.ClassA.methodA1(*),methodA1,false,0xae30
```

If the subclass flag is `false`, all the methods matching with the coded method names are traced.

Methods to be traced

- `methodA1(String,Object)` of the `ClassA` class
- `methodA1(int)` of the `ClassA` class

If the method to be traced is invoked, `0xae30` is output as the event ID at the method entry, and `0xae31` is output at the method exit. Also, `methodA1` is output as the identity ID for all the methods to be traced.

- **If the subclass flag is** `true`

```
com.sample.ClassA.methodA1(*),methodA1,true,0xae30
```

If the subclass flag is `true`, all the methods matching with the coded method names are traced. Also, in addition to the coded method, the methods overriding the coded method are also traced.

Methods to be traced

- `methodA1(String,Object)` and `methodA1(int)` of the `ClassA` class
- `methodA1(String,Object)` of the `ClassB` class that overrides `methodA1(String,Object)` of the `ClassA` class
- `methodA1(String,Object)` of the `ClassC` class that overrides `methodA1(String,Object)` of the `ClassB` class

If the method to be traced is invoked, `0xae30` is output as the event ID at method entry, and `0xae31` is output at method exit. Also, `methodA1` is output as the identity ID for all the methods to be traced.

(c) To trace all the methods with matching class names

The following is an example of coding the user-extended trace based performance analysis configuration file when you trace all the methods with matching class names, omitting the methods and arguments.

- **If the subclass flag is** `false`

```
com.sample.ClassA,TEST01,false
```

If the subclass flag is `false`, all the methods with the coded class name (`ClassA` class) are traced.

Methods to be traced

`methodA1(String,Object)`, `methodA1(int)`, and `methodA2()` of the `ClassA` class

If the method to be traced is invoked, 0xae00 is output at method entry and 0xae01 is output at method exit because the event ID is omitted. Also, `TEST01` is output as the identity ID for all the methods.

- **If the subclass flag is** `true`

```
com.sample.ClassB,TEST02,true
```

If the subclass flag is `true`, all the methods with the coded class name (`ClassB` class), and all the methods that override these methods are traced.

Methods to be traced

- `methodA1(String,Object)` of the `ClassB` class
- `methodA2()` of the `ClassB` class
- `methodB1()` of the `ClassB` class

- `methodA1(String,Object)` of the `ClassC` class that overrides `methodA1(String,Object)` of the `ClassB` class

If the method to be traced is invoked, 0xae00 is output at method entry and 0xae01 is output at method exit because the event ID is omitted. Also, `TEST02` is output as the identity ID for all the methods.

(d) To trace all the methods with matching package names

The following is an example of coding the user-extended trace based performance analysis configuration file when you trace all the methods of all the classes with matching package names, omitting the class names, method names, and arguments.

Important note

This specification also includes the sub-packages as trace targets. When a sub-package is to be traced, the trace is output when the target method is invoked.

If the `com.sample` package has a sub-package, all the methods of all the classes of that sub-package are also traced.

- **If the subclass flag is `false`**

```
com.sample.*,6000,false
```

If the subclass flag is `false`, all the methods of all the classes in the coded package (`com.sample`) are traced.

Methods to be traced

All the methods of `ClassA`, `ClassB`, `ClassC`, and `ClassD` are traced.

If the method to be traced is invoked, 0xae00 is output at method entry and 0xae01 is output at method exit because the event ID is omitted. Also, `6000` is output as the identity ID for all the methods.

- **If the subclass flag is `true`**

```
com.sample.*,6000,false
```

If the subclass flag is `true`, all the methods of all the classes in the coded package (`com.sample`), and all the methods that override these methods are traced.

Methods to be traced

All the methods of `ClassA`, `ClassB`, `ClassC`, and `ClassD`, and all the methods that override these methods are traced.

If the method to be traced is invoked, 0xae00 is output at method entry and 0xae01 is output at method exit because the event ID is omitted. Also, `6001` is output as the identity ID for all the methods.

(5) Notes on creating the configuration file for the user-extended trace based performance analysis

The precautions to be taken when you create a configuration file for the user-extended trace based performance analysis are as follows:

- If you specify a different event ID or identity ID in multiple lines for the same method in the configuration file for the user-extended trace based performance analysis, the settings coded first in the configuration file for the user-extended trace based performance analysis will have priority.

- If you code multiple methods as targets in the configuration file for the user-extended trace based performance analysis, all the target methods are output with the same event ID or identity ID. In this case, if the method name that is output exceeds the number of characters that can be output, you might not be able to identify the method. Specify the settings so that one method can be identified.
- If you want to trace an interface, specify `true` in the subclass flag. If you specify `false` in the subclass flag, the trace information is not output because the methods to be traced do not exist in an interface.
- Use the specification format where multiple methods are the targets for user-extended analysis trace only when you want to understand the application operations.

  In the code of the configuration file for the user-extended trace based performance analysis, if `true` is specified in the subclass flag, and you specify the specification format by which the methods with matching method names are traced, many unintended methods become targets of the user-extended trace based performance analysis, and identifying the cause of performance deterioration might become difficult. To identify the cause of performance deterioration, we recommend that you limit the targets for user-extended trace based performance analysis by specifying `false` in the subclass flag of the configuration file for the user-extended trace based performance analysis, or use the specification format by which the methods with matching method name and argument types are traced.

7.6 Logs output when the user-extended trace based performance analysis is executed

When you execute the user-extended trace based performance analysis, the following logs are output to the JavaVM log file:

- Logs for reading the configuration file for the user-extended trace based performance analysis
- Logs for application rewriting

The following points describe the respective logs.

7.6.1 Logs for the reading of the configuration file for the user-extended trace based performance analysis

The following table describes the logs output when the configuration file for the user-extended trace based performance analysis is read during the user-extended trace based performance analysis.

Table 7–9: Logs output the configuration file for the user-extended trace based performance analysis is read

Output format	Output contents	Explanation
`[UPR`[#1]`]<DATE>Setting file not found.<file=FILEPATH>`	• `DATE`[#2] The date and time at which an attempt to read the configuration file for the user-extended trace based performance analysis failed. • `FILEPATH` The absolute file path of the configuration file for the user-extended trace based performance analysis that could not be read.	The default configuration file, or the configuration file for the user-extended trace based performance analysis specified in the `jvm.userprf.File` property does not exist.
`[UPR`[#1]`]<DATE>Failed to open setting file.<file=FILEPATH>`	• `DATE`[#2] The date and time at which an attempt to read the configuration file for the user-extended trace based performance analysis failed. • `FILEPATH` The absolute file path of the configuration file for the user-extended trace based performance analysis that could not be read.	The configuration file for the user-extended trace based performance analysis cannot be opened or read.
`[UPR`[#1]`]<DATE>Failed to parse setting file.<file=FILEPATH><line=LINE>`	• `DATE`[#2] The date and time at which an error was detected in the configuration file for the user-extended trace based performance analysis. • `FILEPATH` The absolute file path of the configuration file for the user-extended trace based performance analysis. • `LINE`	There is a description format error in the contents of the configuration file for the user-extended trace based performance.

Output format	Output contents	Explanation
	The line number in the configuration file for the user-extended trace based performance analysis.	
[UPR[#1]]<DATE>Event ID is invalid value.<file=FILEPATH><eventID=EventID>	• DATE[#2] The date and time at which an error was detected in the configuration file for the user-extended trace based performance analysis. • FILEPATH The absolute file path of the configuration file for the user-extended trace based performance analysis. • EventID: The event ID specified in the user-extended trace based performance analysis configuration file.	The event ID of the configuration file for the user-extended trace based performance analysis is outside the range of valid values.
[UPR[#1]]<DATE>No valid settings in setting file.<file=FILEPATH>	• DATE[#2] The date and time at which invalid settings were detected in the configuration file for the user-extended trace based performance analysis. • FILEPATH The absolute file path of the configuration file for the user-extended trace based performance analysis.	The settings in the configuration file for the user-extended trace based performance analysis are invalid.
[UPR[#1]]<DATE>User Extended PRF started successfully.<file=FILEPATH>	• DATE[#2] The date and time at which the reading of the configuration file for the user-extended trace based performance analysis was successful. • FILEPATH The absolute file path of the configuration file for the user-extended trace based performance analysis that was read properly.	The user-extended trace based performance analysis configuration file is read correctly, and the user-extended trace based performance analysis is now valid.

#1

An identifier indicating that the log was output by the user-extended trace based performance analysis.

#2

Output in the same format as the extended verbosegc information.

7.6.2 Logs for application rewriting

The user-extended trace based performance analysis uses the instrumentation functionality and rewrites the application classes when the class is loaded, in order to output the trace based performance analysis. The following table describes the logs output at this time.

Table 7–10: Logs for application rewriting

Output format	Output contents	Explanation
[UPR[#1]]<DATE>BCI process failed.<class=CLASS>	• DATE[#2] The date and time at which an attempt to rewrite failed.	An attempt to rewrite the class specified in the configuration file for the user-extended

Output format	Output contents	Explanation
	• CLASS The fully qualified class name for which rewriting failed.	trace based performance analysis failed.
[UPR[#1]]<DATE>BCI process finished successfully.<class=CLASS>	• DATE[#2] The date and time at which rewriting was successful. • CLASS The fully qualified class name for which rewriting was successful.	The class specified in the configuration file for the user-extended trace based performance analysis was successfully rewritten. Note that this log is output when true is specified in the jvm.userprf.Trace property.
[UPR[#1]]<DATE>BCI count exceeded a limit.	• DATE[#2] The date and time at which the rewriting count exceeded the upper limit.	An attempt was made to rewrite the methods to be traced, exceeding the number specified in the jvm.userprf.Limit property. Take one of the following actions: • Reduce the number of target methods of the user-extended trace based performance analysis. • Increase the value specified in the jvm.userprf.Limit property. Note that this log is output when the value specified in the jvm.userprf.Limit property is exceeded for the first time.

#1

An identifier indicating that the log was output by the user-extended trace based performance analysis.

#2

Output in the same format as the extended verbosegc information.

7.7 Analysis operation of the processing performance by using the trace based performance analysis file

This section describes about the analysis operation of the processing performance using the trace based performance analysis file.

7.7.1 Overview of the Operation for Analyzing the Processing Performance

You can analyze the processing performance of Application Server based on the trace information output from the function layer such as the EJB container 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.

Note that the trace information of each function layer is output to the PRF trace file in the binary format. When analyzing the processing performance of Application Server, use the trace based performance analysis file where PRF trace file is converted in to text format (CSV format).

The description of the operation of analyzing the processing performance of Application Server is as follows:

Reference note

Maintenance level trace information

The PRF trace collection level includes the maintenance level in addition to the standard level and detailed level. The maintenance level is a level for acquiring the maintenance information in the case of an error. Do not specify the maintenance level in normal cases.

For how to set up the maintenance level in PRF trace collection levels, see *cprflevel (display or change the PRF trace collection level)* in the *uCosminexus Application Server Command Reference Guide*.

(1) Collecting a trace based performance analysis file

Collect the trace based performance analysis files used for analyzing the processing performance of Application Server, which is output using the management command `mngsvrutil`. For details on how to collect the trace based performance analysis file and on the output destinations and output information of the trace based performance analysis file, see *7.3 Collecting the trace based performance analysis file by using Management Server*.

(2) Analyzing the processing performance of Application Server using a trace based performance analysis file

To analyze the performance using a trace based performance analysis file, display the file in an application program that can edit CSV format files, and use the filtering or sorting function as required.

For example, when the information is output in the CSV format, you can filter the output items as 'event-wise', or 'process-wise', in the application programs that can edit files in the CSV format, list the locations to be focused, and analyze the information according to your requirement.

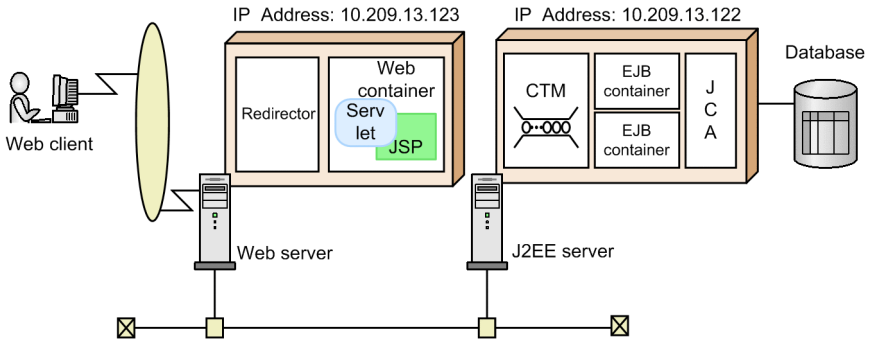
How to use the trace based performance analysis file is described below by giving an example.

- For details on analyzing the response time of the Web server, see *7.7.2 Analyzing the Response Time of a Web Server*.

- For details on investigating the processing status of a request in Application Server, see *7.7.3 Investigating the Processing Status of a Request in an Application Server*.
- For details on investigating the life cycle of the session, see *7.7.4 Investigating the Life Cycle of a Session*.
- For details on identifying the transaction in which the timeout occurs, see *7.7.5 Identifying the Transaction in Which a Timeout Occurred*.
- For details on identifying of the request for which time-out occurs, see *7.7.6 Identifying the Request for Which Timeout Occurred*.
- For details on the log investigation where root application information is used, see *7.7.7 Investigating the Log Using the Root Application Information*.
- For details on identifying the connection where the trouble occurred, see *7.7.8 Identifying the Connection in Which an Error Occurred*.
- For details on the investigation about the location of the problem associated to Trace based performance analysis and thread dump, see *7.7.9 Investigation about the location of the problem associated to the trace based performance analysis file and thread dump*.

Note that the collection of trace based performance analysis files in such an environment as the following Web client configuration environment is a prerequisite.

Figure 7–9: Example of Web client configuration



7.7.2 Analyzing the Response Time of a Web Server

This subsection 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 can output the time taken to process the request in the access log of the Web server. For details, see the directive in *6.2.3(6) CustomLog {file-name | pipe} {"format" | label-name} [env=[!]environment-variable]* in the *uCosminexus Application Server HTTP Server User Guide*.

The following is an example of the output of the access log specified in the default configuration file.

Output example

```
192.168.10.10 - - [12/Feb/2020:20:52:16.352 +0900] "GET /index.html HTTP/1.1
" 200 53 0.013 11340 "192.168.10.10/5636/0x0000000000000022"
192.168.10.10 - - [12/Feb/2020:20:52:19.439 +0900] "GET /wait.pl HTTP/1.1" 5
04 319 3.004 12548 "192.168.10.10/5636/0x0000000000000023"
```

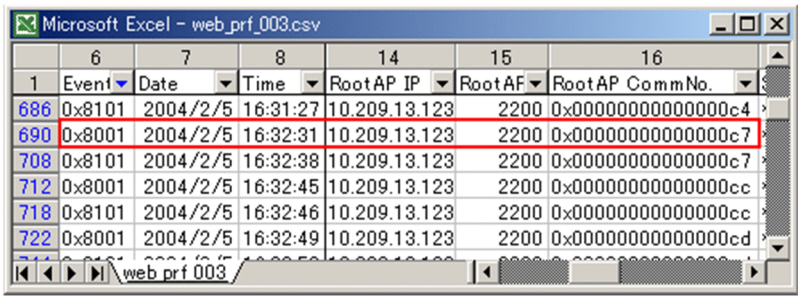
The output example shows that it took 0.013 seconds to process the request to `index.html` and 3.004 seconds to process the request to wait.pl. In addition, by comparing the root application information output in each message with the root application information output in the trace based performance analysis file, you can analyze the processing time for each functionality layer.

7.7.3 Investigating the Processing Status of a Request in an Application Server

This subsection gives an example to describe how to investigate whether the request received by a Web server from a client is processed on an application server.

In the following figure, from the trace based performance analysis files collected on the Web server you will understand that the request is received on 2004/2/5 16:32:31.

Figure 7–10: Example of the trace based performance analysis file collected in the Web server

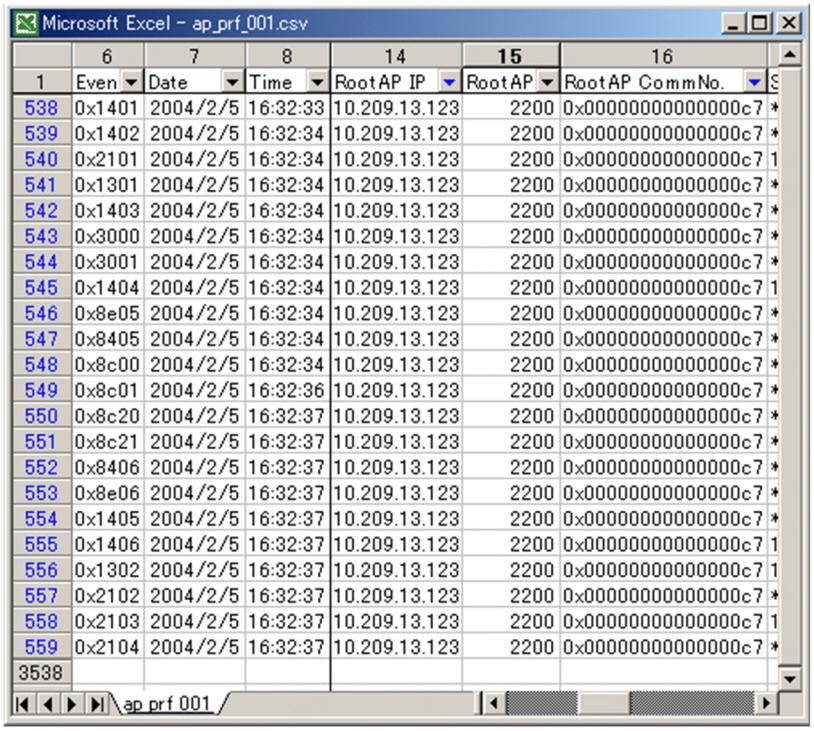


	6	7	8	14	15	16
1	Event	Date	Time	RootAP IP	RootAP	RootAP CommNo.
686	0x8101	2004/2/5	16:31:27	10.209.13.123	2200	0x00000000000000c4
690	0x8001	2004/2/5	16:32:31	10.209.13.123	2200	0x00000000000000c7
708	0x8101	2004/2/5	16:32:38	10.209.13.123	2200	0x00000000000000c7
712	0x8001	2004/2/5	16:32:45	10.209.13.123	2200	0x00000000000000cc
718	0x8101	2004/2/5	16:32:46	10.209.13.123	2200	0x00000000000000cc
722	0x8001	2004/2/5	16:32:49	10.209.13.123	2200	0x00000000000000cd

You use the root application information of this request as a key to investigate trace based performance analysis files collected on Application Server.

The following figure shows an example of filtering the trace based performance analysis files collected in Application Server with RootAP IP:10.209.13.123, RootAP ID:2200, and RootAP CommNo.:0x00000000000000c7 as keys.

Figure 7–11: Example of the filtered trace based performance analysis file collected in Application Server



Microsoft Excel - ap_prf_001.csv

	6	7	8	14	15	16	
1	Even	Date	Time	RootAP IP	RootAP	RootAP CommNo.	S
538	0x1401	2004/2/5	16:32:33	10.209.13.123	2200	0x00000000000000c7	*
539	0x1402	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	*
540	0x2101	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	1
541	0x1301	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	*
542	0x1403	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	*
543	0x3000	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	*
544	0x3001	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	*
545	0x1404	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	1
546	0x8e05	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	*
547	0x8405	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	*
548	0x8c00	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	*
549	0x8c01	2004/2/5	16:32:36	10.209.13.123	2200	0x00000000000000c7	*
550	0x8c20	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	*
551	0x8c21	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	*
552	0x8406	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	*
553	0x8e06	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	*
554	0x1405	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	*
555	0x1406	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	1
556	0x1302	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	1
557	0x2102	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	*
558	0x2103	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	1
559	0x2104	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	*
3538							

ap_prf_001

In this way, if you filter the trace based performance analysis files collected on an application server with a root application information as a key, you can investigate the processing of a series of requests in an application server.

7.7.4 Investigating the Life Cycle of a Session

This subsection describes how to investigate the life cycle of a session using a trace based performance analysis.

You can investigate the life cycle of a session by using the trace (Session trace) that is output when the PRF trace collection level is set as details.

You can investigate session trace information by using the session ID in the output trace based performance analysis file as a key.

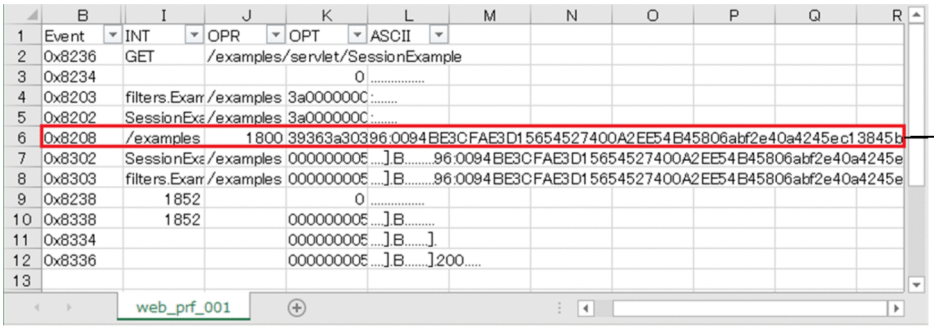
The session ID is output to the option area of the event, output as the session trace information in the following format:

```
Number of session characters: Session ID
```

For example, if there is a session ID called as abc123, in the option area 6:abc123 is output. When the session ID is a blank character, 0: is output as the number of session characters. When the session ID cannot be acquired, nothing is output.

The following figure gives an example of trace based performance analysis file in which the session trace information is output. In this example, the trace information of a request to generate a session, a request to use the session, and a request to destroy the session are output in the validity period of one session. Also, a request to generate a session, a request to use the session, and a request to destroy the session are shown separately. In reality, the trace information in these three examples is output one after another. In these windows, all items are related to the session trace.

Figure 7–12: Example of trace based performance analysis file where the session trace information is output (request part where the session is created)



	B	I	J	K	L	M	N	O	P	Q	R
1	Event	INT	OPR	OPT	ASCII						
2	0x8236	GET	/examples/servlet/SessionExample								
3	0x8234			0	.....						
4	0x8203	filters.Exam	/examples	3a0000000	:.....						
5	0x8202	SessionExa	/examples	3a0000000	:.....						
6	0x8208	/examples	1800	39363a30396:0094BE3CFAE3D15654527400A2EE54B45806abf2e40a4245ec13845b							
7	0x8302	SessionExa	/examples	000000005	....].B.....96:0094BE3CFAE3D15654527400A2EE54B45806abf2e40a4245e						
8	0x8303	filters.Exam	/examples	000000005	....].B.....96:0094BE3CFAE3D15654527400A2EE54B45806abf2e40a4245e						
9	0x8238	1852		0	.....						
10	0x8338	1852		000000005	....].B.....						
11	0x8334			000000005	....].B.....]						
12	0x8336			000000005	....].B.....]200.....						
13											

web_prf_001

1.

1. In the ASCII example of the 0x8208 event, the generated session ID is output. 1800 in the option column indicate the session validity period.

The items that you can confirm in this trace file are described here.

Confirming the validity period of the session

When generating a session, the 0x8208 event is output. When destroying a session, the 0x8209 event is output. You can confirm the validity range of the session from the date and time of these acquisitions. Also, in the case of the 0x8208 event, you can even confirm the validity period (seconds) of the session generated from the operation name.

The same ID is output to the event 0x8209 of the request to destroy the session until the session is destroyed. You can confirm the date and time of generating the session that is destroyed from the operation name.

7.7.5 Identifying the Transaction in Which a Timeout Occurred

This subsection describes how to identify the transaction that is timed out using the message or trace based performance analysis information file, if a timeout occurs in the transaction of a J2EE application or batch application.

The subsection describes the following two methods:

- Method of identifying the transaction using messages
- Method of identifying the transaction depending on the output contents of the trace based performance analysis file

(1) Method of identifying the transaction using messages

If a timeout occurs in a transaction, the KDJE31002-W and KDJE50080-W messages are output. The following information is output in these messages:

KDJE31002-W

- J2EE application name or the class name of the batch application that started the transaction.
- Hash code of the instance and the class name of J2EE component (Enterprise Bean, servlets, or JSPs) that started the transaction
- Maintenance information
- Root application information of the trace based performance analysis

You can confirm the location where the transaction timeout occurred in the trace based performance analysis by matching and confirming the root application information of the trace based performance analysis output to the message with the root application information that is output to the trace based performance analysis file.

You can confirm the trace information before and after the timeout for checking the process contents of the transaction in which a timeout occurred.

KDJE50080-W

- IP address, process ID, and root application information
- Connection ID of the connection that executed the SQL
- Name of the method that executed SQL
- Whether the method that executed SQL is running
- SQL executed last

You can refer to the SQL information that is output in the message, and can investigate the location where the timeout occurred.

For details on messages, see *KDJE31002-W* and *KDJE50080-W* in the *uCosminexus Application Server Messages*.

(2) Method of identifying the transaction depending on the output contents of the trace based performance analysis file

In the trace based performance analysis file, the trace information is output at the trace collection point immediately before or after the occurrence of timeout in the transaction.

Check the processing contents of the following event IDs.

Table 7–11: Trace based performance analysis that is output when timeout occurs in the transaction

Event ID	Description
0x8819	This information is output in the process immediately before a timeout occurs in the transaction. The root application information of the transaction that is timed out is output to the interface name.
0x8820	This information is output to the process immediately after a transaction is timed out.
0x8C41	This information is output when the SQL for investigating an error is output.

7.7.6 Identifying the Request for Which Timeout Occurred

This subsection describes how to use trace based performance analysis to identify a request that has timed out if a timeout occurs while receiving a response on the reverse proxy of a Web server.

If a timeout occurs while receiving a response on the reverse proxy, the message AH01102 is output to the error log of the Web server, with the detailed information indicating that a timeout has occurred. Based on the thread ID and time included in this message, search the Web server request log for a proxy trace. The following information is output in the proxy trace:

- Connection status with the backend server
- IP address of the connection destination
- Port number of the connection destination
- Root application information of the trace based performance analysis

By comparing the root application information of the trace based performance analysis output in the message of the proxy trace with the root application information output in the trace based performance analysis file, you can check where in the trace based performance analysis the request timeout occurred.

You can also check the URI of the request that has timed out by comparing the corresponding root application information with the root application information output to the access log of the Web server. Based on the information, identify the request that has timed out.

7.7.7 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 7–12: Trace point to which the new root application information is assigned

Event ID (process contents)	Process contents
0x8236	When acquiring a request, when completing the request header analysis

#

If a Web server is connected, the root application information obtained by the HTTP Server is assigned. If the root application information was not issued because the HTTP Server or the PRF trace output library used by the HTTP Server failed to load, new root application information is assigned by 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'

- 0x8237
  When data is read from Web client.
- 0x8238
  When data is written to the Web client.

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 the implementation of acquiring the root application when the J2EE application or batch application is started, see the section *7.4 Implementation for collection of root application information of trace based performance analysis*. For details on the APIs, see the *uCosminexus Application Server API Reference Guide*.

7.7.8 Identifying the Connection in Which an Error Occurred

When HiRDB or Oracle is used as a database, you can compare the connection ID output to the trace based performance analysis file with the connection ID output to the log file and trace file , HiRDB client, and Oracle client for confirming the connection in which an error has occurred. For details on how to confirm, see *Appendix B. Identifying the Connection in Which an Error Has Occurred When Connecting to a Database*.

7.7.9 Investigation about the location of the problem associated to the trace based performance analysis file and thread dump

If slow down or hang-up occurs in a J2EE application or batch application, you can investigate the location of the error by correlating the trace based performance analysis file with the thread dump.

Use the thread ID that is output to the trace based performance analysis file and nativeID (thread IDs of the OS level) of the thread information output to the thread dump, for correlating the trace based performance analysis file with the thread dump. This subsection describes how to identify the corresponding thread dump from the trace based performance analysis file.

The following are the steps to identify the corresponding thread dump from the trace based performance analysis file:

1. Collect the trace based performance analysis file and thread dump.

   For details on the collection methods of trace based performance analysis file, see *7.3.1 How to collect a trace based performance analysis file*. For details on the collection method of thread dump, see *4.7 JavaVM thread dump*.

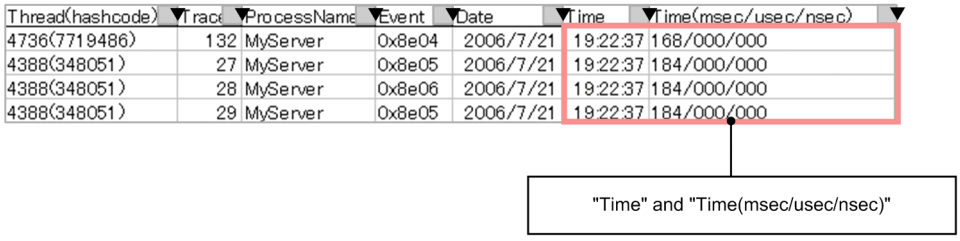
2. Select the trace based performance analysis file and thread dump to be used.

   Select the trace based performance analysis file and thread dump to be used, based on the date and time when the trace based performance analysis file and thread dump are output. For details on the output time, see the following information:

   Trace based performance analysis file

   Time and Time(msec/usec/nsec)

   The following figure shows the Time and Time (msec/usec/nsec) of the trace based performance analysis file.



   Thread dump

   Date and time output at the end of the file name and file.

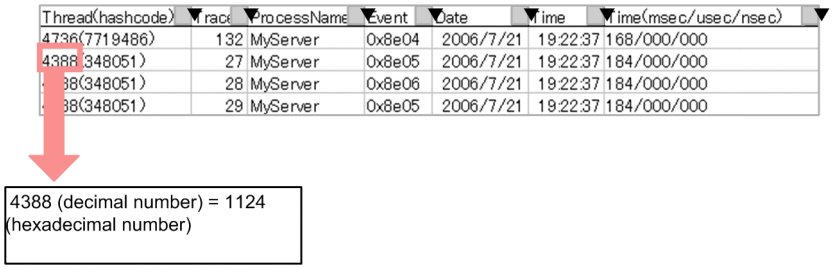
   The following is an example showing the date and time output at the end of a file:

```
 ...
 ...
Full thread dump completed. Fri Jul 21 19:22:47 2006
```

3. Convert the value of Thread (hash code), corresponding to the event ID of the trace based performance analysis file that you want to investigate, in to a hexadecimal number.

   - In Windows or AIX

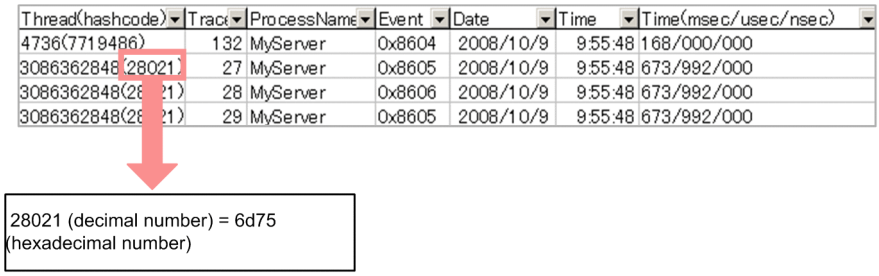
     Change the value of the thread (thread ID) from decimal numbers to hexadecimal numbers.



Thread(hashcode)	Trace	ProcessName	Event	Date	Time	Time(msec/usec/nsec)
4736(7719486)	132	MyServer	0x8e04	2006/7/21	19:22:37	168/000/000
4388(348051)	27	MyServer	0x8e05	2006/7/21	19:22:37	184/000/000
8(348051)	28	MyServer	0x8e06	2006/7/21	19:22:37	184/000/000
8(348051)	29	MyServer	0x8e05	2006/7/21	19:22:37	184/000/000

   - In Linux

     Change the value of the `hashcode` (hash code) from decimal numbers to hexadecimal numbers.



Thread(hashcode)	Trace	ProcessName	Event	Date	Time	Time(msec/usec/nsec)
4736(7719486)	132	MyServer	0x8604	2008/10/9	9:55:48	168/000/000
3086362848(28021)	27	MyServer	0x8605	2008/10/9	9:55:48	673/992/000
3086362848(28021)	28	MyServer	0x8606	2008/10/9	9:55:48	673/992/000
3086362848(28021)	29	MyServer	0x8605	2008/10/9	9:55:48	673/992/000

4. Specify the thread information corresponding to the thread matching to the value of `Thread (hashcode)` that is converted into a hexadecimal number in step 3.

   - In Windows or AIX

     The thread dump `nid` searches for the thread that matches with the value (`1124`) of the `Thread` (Thread ID) that is converted into hexadecimal numbers.

```
 ...
 ...
"VBJ ThreadPool Worker" daemon prio=5 jid=0x00054f93 tid=0x04cef380 nid
=0x1124 in Object.wait() [0x0632f000..0x0632fd18]
 stack=[0x06330000..0x062f5000..0x062f1000..0x062f0000]
 [user cpu time=0ms, kernel cpu time=15ms] [blocked count=1, waited cou
nt=29]
at java.lang.Object.wait(Native Method)
 ...
 ...
```

   - In Linux

     The thread dump `jid` searches for the thread that matches with the value (`6d75`) of `hashcode` (hash code) that is converted into hexadecimal numbers.

```
 ...
 ...
"main" prio=1 jid=0x00006d75 tid=0x00201d70 nid=0x1e51 waiting on condi
tion [0x00000000..0xbfe80488]
```

```
 stack=[0xbfe87000..0xbfc8c000..0xbfc88000..0xbfc87000]
 [user cpu time=1320ms, kernel cpu time=4280ms] [blocked count=5, waite
d count=4]
 ...
 ...
```

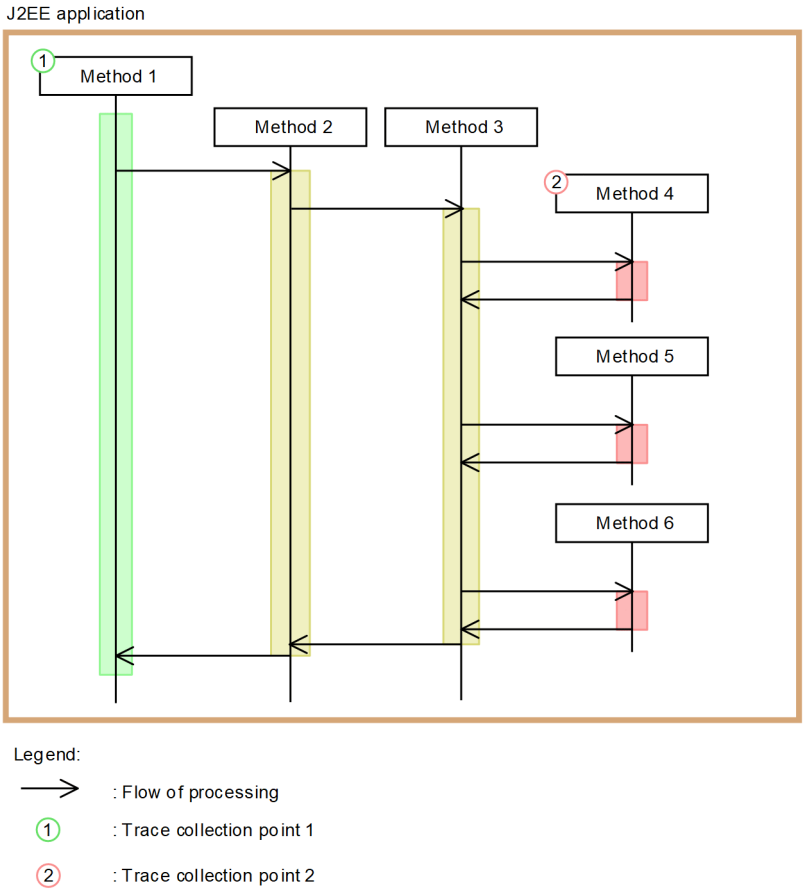
7.8 Notes on using the user-extended trace based performance analysis

This section describes the notes on using the user-extended trace based performance analysis.

- If the methods to be traced are invoked frequently, it might be difficult to identify the method that causes performance deterioration, due to the performance deterioration caused by the processing overhead of the user-extended trace based performance analysis. Therefore, in the methods to be traced, specify a method that serves as the entry to the processing that can identify the cause of deterioration.

  For example, in the case of applications with a method invocation relationship as shown in the following figure, take method 1 (trace collection point 1) that serves as an entry to processing as the target. If you specify the methods 4, 5, and 6 of the trace collection point 2, and if the processing of the user-extended trace based performance analysis for these methods takes time, the performance measurement of this application might be affected.

  Figure 7–13: Method invocation relationship in an application



- If you use the user-extended trace based performance analysis for a method invoked before the root application information is acquired, the root application information might be output with 0.
- If you increase the frequency of invoking the method to be traced and specify a property that increases the amount of trace output, the amount of PRF trace file output increases. In this case, increase the number of PRF trace files or the PRF trace file size. For details on how to change the number and size of PRF trace files, see *10.3.1 Setting up the performance tracer* in the *uCosminexus Application Server Management Portal User Guide*.

  Note that the increase in the amount of PRF trace files that are output causes the following events:

  - Overhead occurs due to the writing to the trace files and leads to performance deterioration.
  - The number of trace files is switched due to a large amount of writing, and the J2EE server can no longer obtain the trace that could be obtained.

For normal operations, specify the contents of the user-extended trace based performance analysis configuration file appropriately and adjust the trace collection amount.

- The application classes are rewritten in the user-extended trace based performance analysis. At this time, if the rewritten class size exceeds 64 KB, the rewriting fails. In this case, take actions such as dividing the class to reduce the class size.

8 Trace Collection Points and PRF Trace Collection Levels of the Trace Based Performance Analysis

The trace based performance analysis includes the trace based performance analysis that analyzes the Application Server performance, and the user-extended trace based performance analysis that extends the target of the trace based performance analysis and analyzes the performance of the applications set up on an Application Server machine. This chapter describes the trace collection points and PRF trace collection levels output in the trace based performance analysis and user-extended trace based performance analysis.

For an overview of the trace based performance analysis and for details about how to perform the analysis, see the chapter *7. Performance Analysis by Using Trace Based Performance Analysis*.

8.1 Organization of this chapter

This chapter describes the trace collection points and PRF trace collection levels of the following trace based performance analysis.

- Trace collection points and PRF trace collection levels output for each function layer by the trace based performance analysis

  When a request is sent from a Web client or EJB client, the J2EE server, EJB client, and CTM output trace information at fixed processing points (trace collection points). Furthermore, in the trace based performance analysis, the trace can be output by setting up the PRF trace collection level, such as *Standard* and *Advanced*.

- Trace collection points and PRF trace collection levels output for each method forming the starting point of application processing by the user-extended trace based performance analysis

  In the user-extended trace based performance analysis, by specifying a method that forms the starting point of application processing as the processing point for acquiring the application performance analysis information, you can output the trace at the point when that method is called.

The following table describes the organization of this chapter.

Table 8–1: Organization of this chapter (trace based performance analysis)

Category	Title	Reference location
Explanation	Trace Get Point of trace based performance analysis and the PRF Trace Get Level	*8.2*
	Trace collection points of a CTM	*8.3*
	Trace collection points of a Web container (trace of request processing)	*8.4*
	Trace collection points of a Web container (session trace)	*8.5*
	Trace collection points of a Web container (filter trace)	*8.6*
	Trace collection points of a Web container (trace of the database session failover functionality)	*8.7*
	Trace collection points of an EJB container	*8.8*
	Trace collection points of a JNDI	*8.9*
	Trace collection points of a JTA	*8.10*
	Trace collection points of a DB Connector and JCA container	*8.11*
	Trace collection points of an RMI	*8.12*
	Trace collection points of an OTS	*8.13*
	Trace collection points of standard output, standard error output, and user log	*8.14*
	Trace collection points of a DI	*8.15*
	Trace collection points of the batch application execution functionality	*8.16*
	Trace collection points of the TP1 inbound integrated function	*8.17*
	Trace collection points of Cosminexus JMS Provider	*8.18*
	Trace collection points of JavaMail	*8.19*
	Trace collection points of JSF 2.2	*8.20*
	Trace collection points of CDI	*8.21*
	Trace collection points when a J2EE server is started or terminated	*8.22*

Category	Title	Reference location
	Trace collection points of an application	*8.23*
	Trace collection points of JAX-RS	*8.24*
	Trace collection points of Java batch	*8.25*
	Trace collection points of WebSocket	*8.26*
	Trace collection points of Concurrency Utilities	*8.27*

Note:

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

For an overview of the trace based performance analysis, and details on how to perform the analysis, see *7. Performance Analysis by Using Trace Based Performance Analysis*.

8.2 Trace Get Point of trace based performance analysis and the PRF Trace Get Level

This section describes the trace collection points and PRF trace collection levels.

8.2.1 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 about trace collections during the startup and termination of the J2EE server, see *8.22 Trace collection points when a J2EE server is started or terminated*.

(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 8–2: 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*
0x8202 to 0x8203 0x8206 to 0x8210 0x8214 to 0x8216 0x8219 to 0x8225 0x8234 to 0x8239 0x8302 to 0x8303 0x8306 to 0x8310 0x8314 to 0x8316 0x8319 to 0x8325 0x8334 to 0x8339	Web container	2	*8.4*, *8.5*, *8.6*, *8.7*
0x8401 to 0x840A 0x8425 to 0x8428 0x842D to 0x8434	EJB container	6	*8.8*

Event ID	Function layer	No. in the figures#	Reference location
0x8453 to 0x8454 0x8460 to 0x846D 0x8470 to 0x8477 0x8490 to 0x8491 0x84A0 to 0x84D9 0x8C41			
0x8603 to 0x861C	JNDI	4	*8.9*
0x8435 to 0x843F 0x8811 to 0x8820 0x8C41	JTA	7	*8.10*
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*
0x8E01 to 0x8E06	RMI	3	*8.12*
0x9400 to 0x9413	OTS	9	*8.13*
0x9C00 to 0x9C03	Standard output, standard error output, and user log	--	*8.14*
0x9D00, 0x9D01	DI	10	*8.15*
0xA100, 0xA101	Batch application execution functionality	11	*8.16*
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*
0xA600 to 0xA60F 0xA610 to 0xA619 0xA61E, 0xA61F 0xA620 to 0xA62F 0xA630 to 0xA63F 0xA640 to 0xA64F 0xA650 to 0xA65F 0xA660 to 0xA667	Cosminexus JMS Provider	15	*8.18*

Event ID	Function layer	No. in the figures#	Reference location
0xA61A, 0xA61B 0xA668 to 0xA66F 0xA670 to 0xA67F 0xA686 to 0xA68D 0xA692 to 0xA69B 0xA69E, 0xA69F 0xA6A0, 0xA6A1 0xA6A6 to 0xA6AB			
0xAD00 to 0xAD15 0xAD80 to 0xAD93	JavaMail	16	*8.19*
0xAF20 to 0xAF2D	JSF 2.2	17	*8.20*
0xb002 to 0xb009	CDI	18	*8.21*
0xD000 to 0xD005	JAX-RS	19	*8.24*
0xD020 to 0xD04D	Java Batch	20	*8.25*
0xE100 to 0xE147	WebSocket	21	*8.26*
0xD050 to 0xD057	Concurrency Utilities	22	*8.27*

Legend:

--: Not applicable

#

Corresponds to the numbers in Figure 8-1 through Figure 8-4.

The following figures show the function layers for which the PRF trace is output, and the trace collection points for each system configuration.

Figure 8–1: Function layers and trace collection points (in the case of Web client configuration)

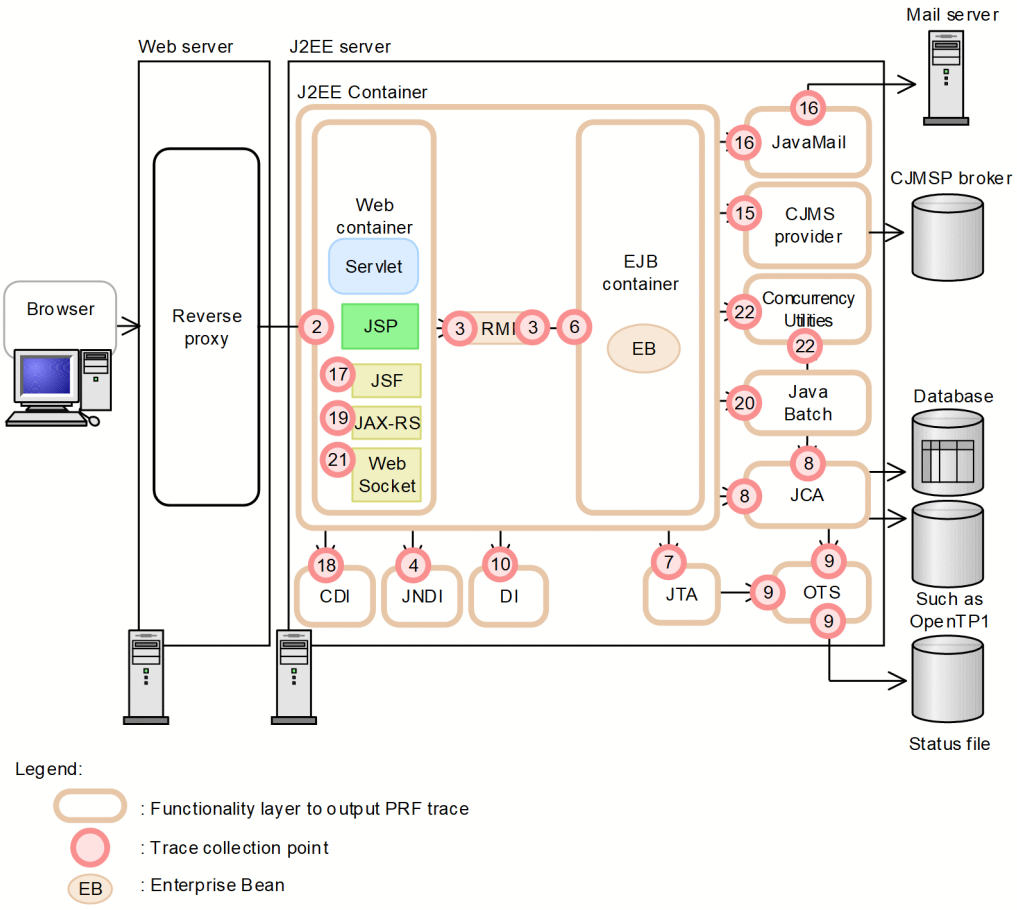


Figure 8–2: Function layers and trace collection points (in the case of a system for executing batch applications)

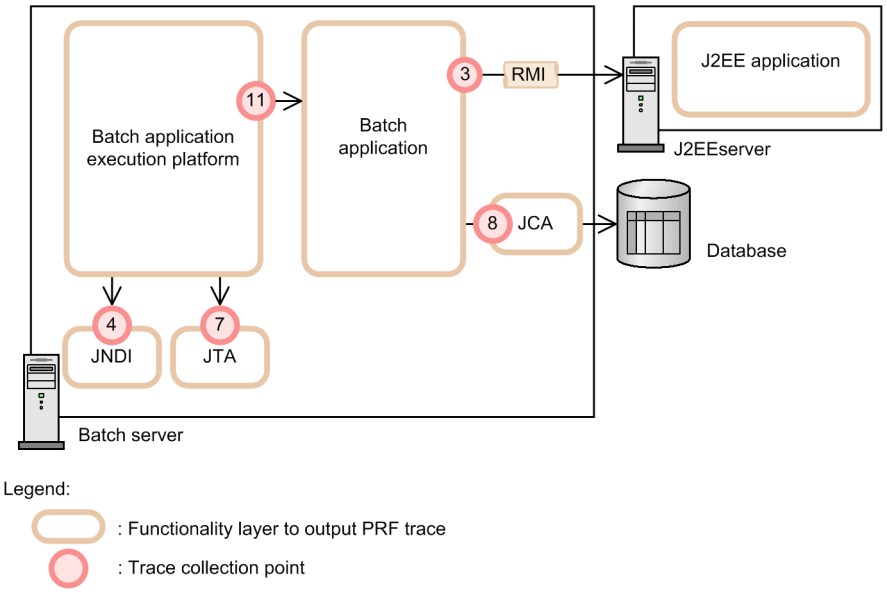


Figure 8–3: Function layers and trace collection points (in the case of EJB client, TPBroker client, or TPBroker OTM client configuration (CTM usage))

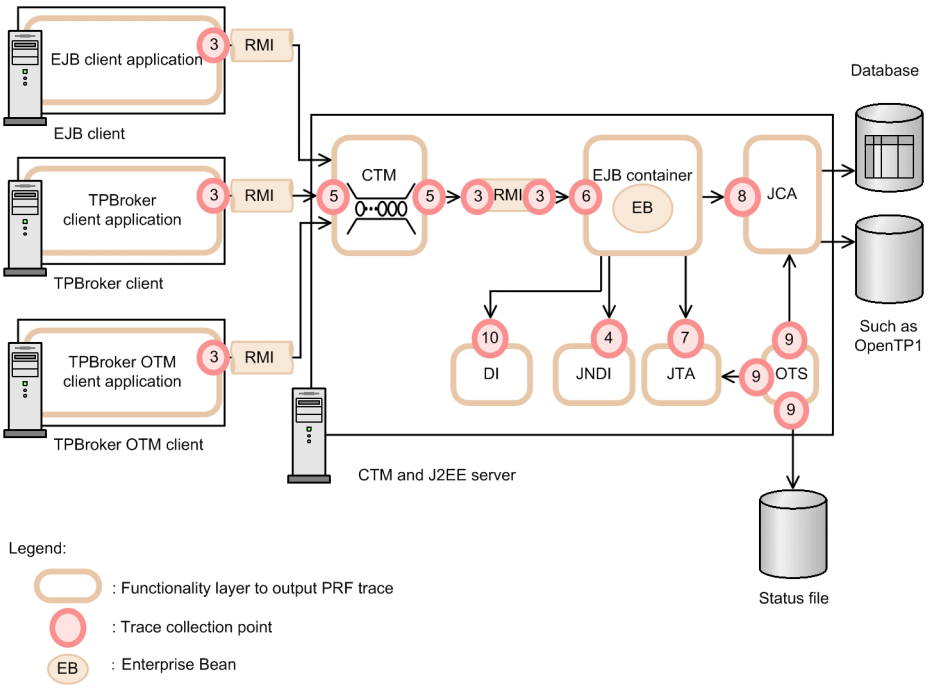
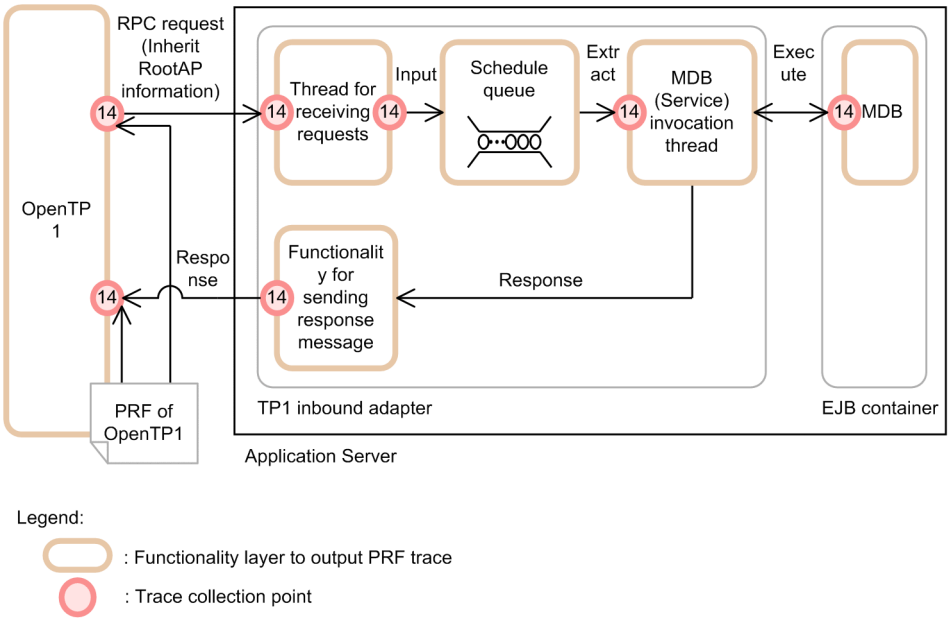


Figure 8–4: Functionality layers and trace collection points (In the TP1 inbound integrated function)



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 8-2*.

Reference note

Apart from the function layers described in *Table 8-2*, 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 8-2* for which the PRF trace can be collected, and their event IDs.

Table 8–3: Correspondence between the event IDs and function layers other than those described in table 8-2 for which the PRF trace can be collected

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)
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 the trace information, see *8.23 Trace collection points of an application*.

(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

8.2.2 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 8-2*.

- 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 in the functional layers of RMI, JSF 2.2, JAX-RS, Java batch, WebSocket, and Concurrency Utilities.

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.

8.3 Trace collection points of a CTM

This section describes the trace collection points in a CTM, and the trace information that can be collected.

8.3.1 Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–4: Details of trace collection points in a CTM

Event ID	No. in the figures#	Trace acquisition points	Level
0x1101	21	Immediately before sending a request from the OTM gateway	A
0x1102	22	Immediately before receiving a response from the OTM gateway	A
0x1301	4	Immediately before sending a request from the EJB regulator	A
0x1302	13	Immediately after receiving a response in the EJB regulator	A
0x1401	1	Entrance of the `Create` method in the CTM	A
0x1402	2	Exit of the `Create` method in the CTM	A
0x1403	5	Immediately after receiving a request from the EJB regulator	A
0x1404	8	Immediately before sending a request from the CTM to the J2EE server or batch server	A
0x1405	9	Immediately after receiving a request from the J2EE server or batch server in the CTM	A
0x1406	12	Immediately before sending a response from the CTM to the EJB regulator	A
0x2002	23	Immediately before receiving a request from the OTM client	A
0x2003	24	Immediately before receiving a response from the OTM client	A
0x2101	3	Immediately after receiving a request from the EJB client or `cjexecjob` command	A
0x2102	14	Immediately before sending a response to the EJB client or `cjexecjob` command	A
0x2103	15	Entrance of the `Remove` method in the CTM	A
0x2104	16	Exit of the `Remove` method in the CTM	A
0x3000	6	Immediately before queue of the request	A
0x3001	7	Immediately after extracting a request from the queue	A
0x3002	17	Immediately before sending a request to another CTM	A
0x3003	18	Immediately after receiving a request from another CTM	A
0x3004	10	Immediately before queue of the request response	B
0x3005	11	Immediately after extracting the request response from the queue	B
0x3006	19	Immediately before sending a request response to another CTM	A
0x3007	20	Immediately after receiving a request response from another CTM	A

Event ID	No. in the figures#	Trace acquisition points	Level
0x3008	--	Collected when the server status changes. (Not collected when a request is being processed)	B

Legend:

A: Standard

B: Advanced

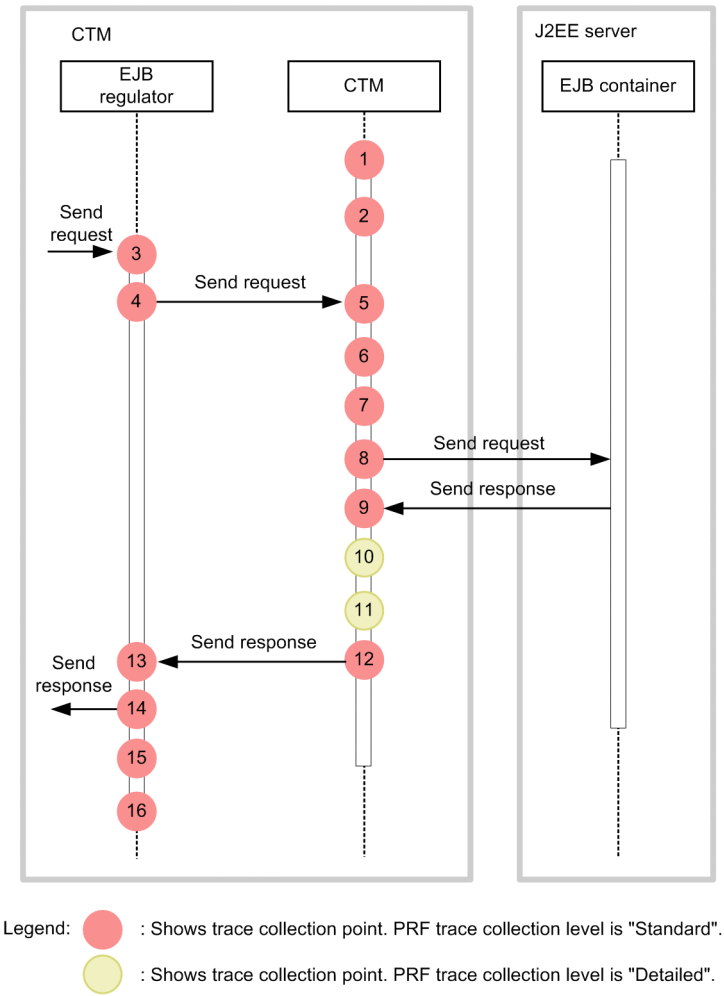
--: Not applicable

#

Corresponds to the numbers in Figure 8-5 through Figure 8-8.

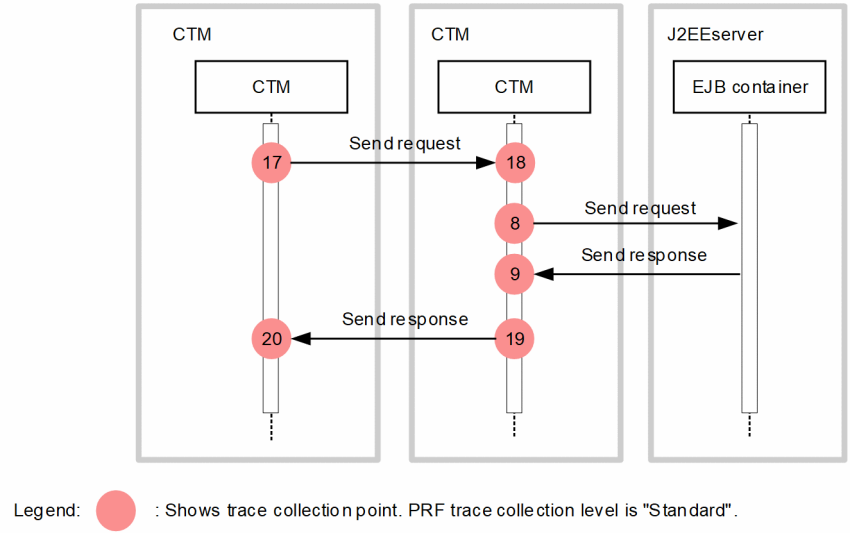
The following figure shows the trace collection points in a CTM.

Figure 8–5: Trace collection points of a CTM



For linkage with another CTM, the trace information is also collected at the locations shown in the following figure. The figure might be referenced in connection with the above figure. Note that this figure displays only the locations that are linked with another CTM.

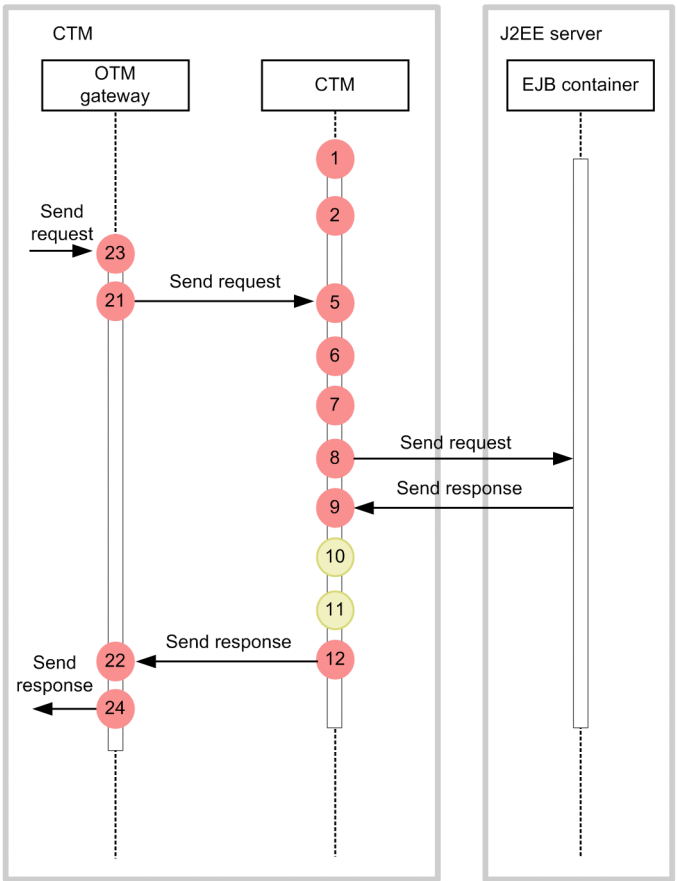
Figure 8–6: Trace collection points of a CTM (For linkage with another CTM)



Legend: : Shows trace collection point. PRF trace collection level is "Standard".

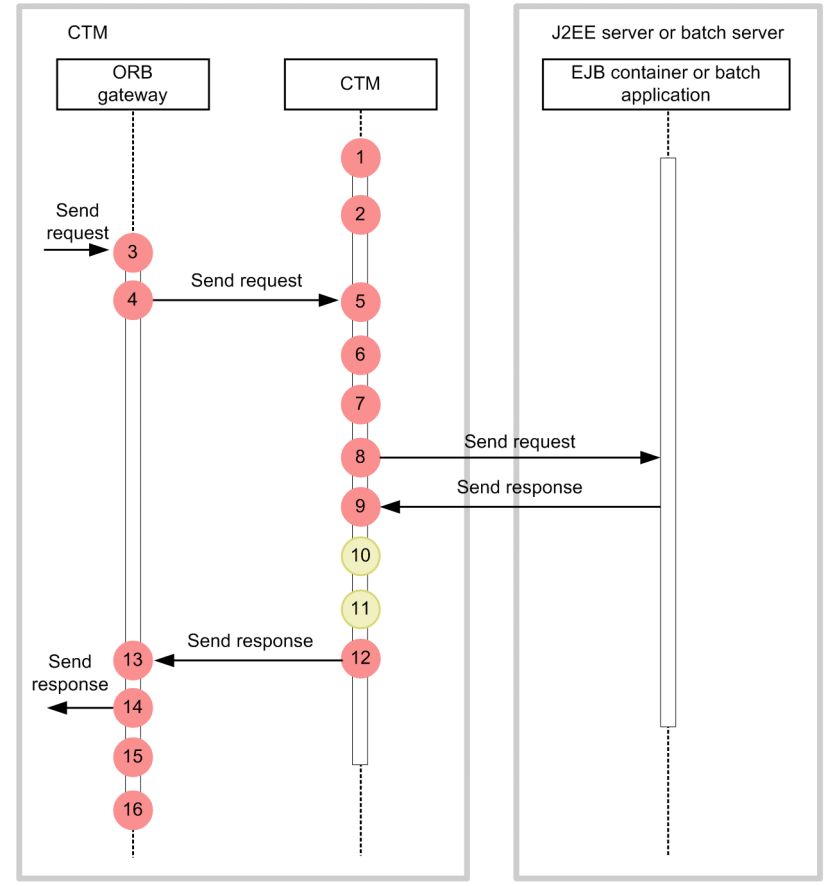
Furthermore, when using the OTM gateway or ORB gateway, the trace information is also collected at the locations shown in the following figure. The figure might be referenced in connection with the above figure. Note that this figure displays only the locations that are linked with another CTM.

Figure 8–7: Trace collection points of a CTM (when using the OTM gateway)



Legend: : Shows trace collection point. PRF trace collection level is "Standard".
: Shows trace collection point. PRF trace collection level is "Detailed".

Figure 8–8: Trace collection points of a CTM (when using the ORB gateway)



8.3.2 Trace information that can be collected

The following table describes the trace information that can be collected in a CTM.

Table 8–5: Trace information that can be collected in a CTM

No. in the figures#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x1401	A	Remote interface name	Method name	--
2	0x1402	A	Remote interface name	Method name	Internal information
3	0x2101	A	Remote interface name	Method name	--
4	0x1301	A	Remote interface name	Method name	--
5	0x1403	A	Remote interface name	Method name	--
6	0x3000	A	--	--	--
7	0x3001	A	--	--	Internal information
8	0x1404	A	Remote interface name	Method name	Internal information

No. in the figures#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
9	0x1405	A	Remote interface name	Method name	Internal information
10	0x3004	B	--	--	--
11	0x3005	B	--	--	Internal information
12	0x1406	A	Remote interface name	Method name	--
13	0x1302	A	Remote interface name	Method name	--
14	0x2102	A	Remote interface name	Method name	--
15	0x2103	A	Remote interface name	Method name	--
16	0x2104	A	Remote interface name	Method name	--
17	0x3002	A	--	--	--
18	0x3003	A	--	--	--
19	0x3006	A	--	--	--
20	0x3007	A	--	--	--
21	0x1101	A	Remote interface name	Method name	--
22	0x1102	A	Remote interface name	Method name	--
23	0x2002	A	Remote interface name	Method name	--
24	0x2003	A	Remote interface name	Method name	--
--	0x3008	B	--	--	Internal information

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-5 through Figure 8-8.

8.4 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.

8.4.1 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 8–6: Details of trace collection points in a Web container (trace of request processing)

Event ID	No. in the figure[#1]	Trace acquisition points	Level
0x8202[#2]	6	Immediately before a servlet or JSP is invoked	A/B
0x8203	5	Immediately before the filter is invoked	B
0x8206	7	Immediately before `forward()` or `include()` of RequestDispatcher is invoked	B
0x8207	6	When the static contents are invoked	A/B
0x8234	4	Immediately after the start of synchronous processing	A
0x8235	14	Immediately after the start of asynchronous processing	A
0x8236	3	Immediately after a request is acquired, or when the request header analysis is complete	A
0x8237	1	Immediately before the start of the reading of data from the reverse proxy	B
0x8238	8	Immediately before the start of the writing of data to the reverse proxy	B
0x8239	16	Immediately after the start of asynchronous servlet processing	A
0x8302	11	Immediately after the processing of the servlet or JSP is complete	A/B
0x8303	12	Immediately after the completion of processing of the filter	B
0x8306	10	Immediately after the completion of the `forward()` or `include()` processing of RequestDispatcher	B
0x8307	11	Immediately after the completion of processing of static contents	A/B
0x8334	13	Immediately before the completion of synchronous processing	A
0x8335	15	Immediately before the completion of asynchronous processing	A
0x8336	18	Immediately after the completion of request processing	A
0x8337	2	Immediately after the completion of the reading of data from the reverse proxy	B
0x8338	9	Immediately after the completion of the writing of data to the reverse proxy	B
0x8339	17	Immediately before the completion of asynchronous servlet processing	A

Legend:

A: Standard

B: Advanced

A/B: Different information is collected for the *Standard* and *Advanced* levels.

#1

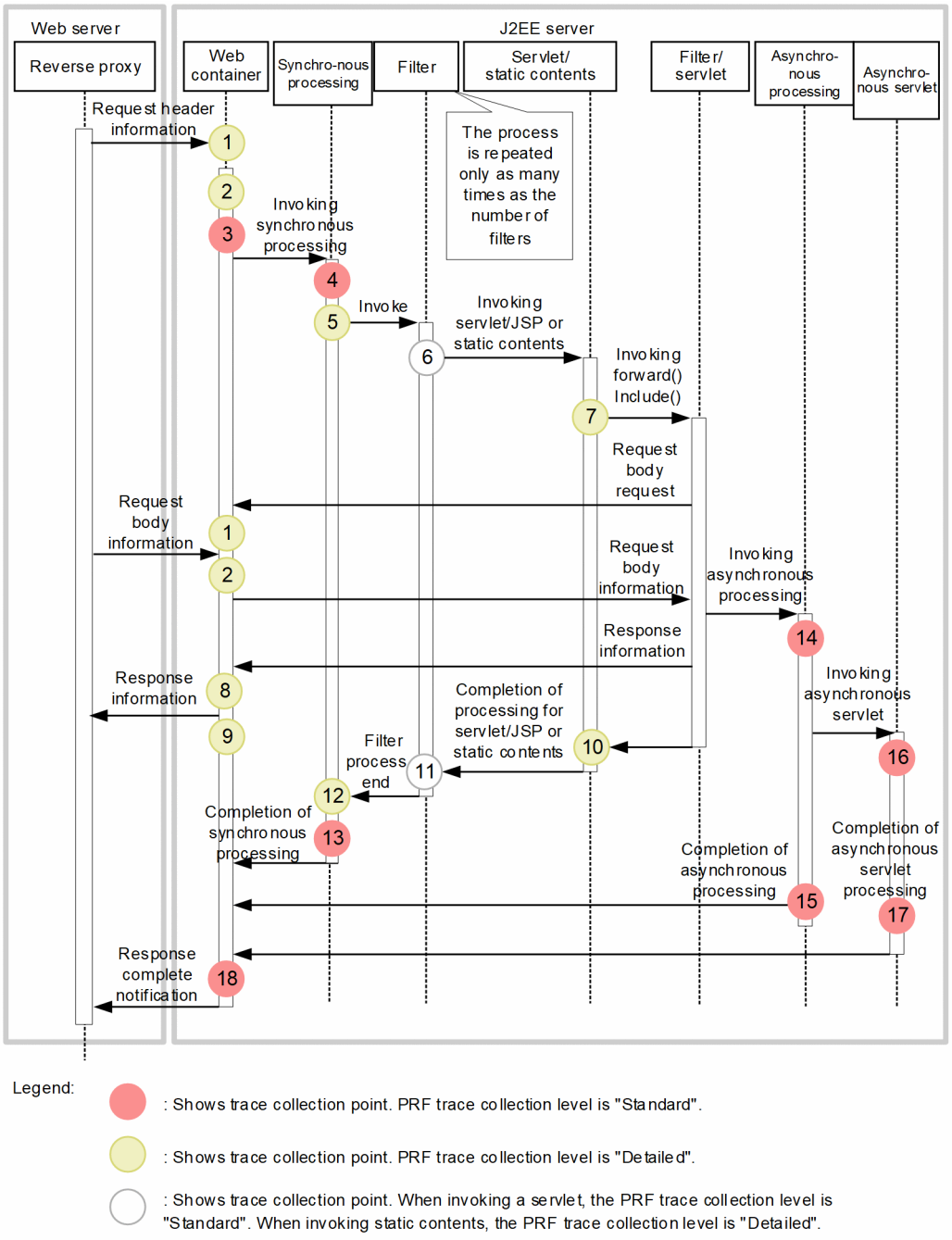
Corresponds to the numbers in Figure 8-9.

#2

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 8–9: Trace collection points of a Web container (trace of request processing)



8.4.2 Trace information that can be collected

The following table describes the trace information that can be collected in a Web container.

Table 8–7: 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	0x8237	B	Requested size	--	--
2	0x8337	B	Size that could be read	--	• When normal: *Entrance-time* • For an exception: *Entrance-time-exception-name*
3	0x8236	A	HTTP method	URI	Value for the request header name specified in the property (none if the header name is not specified)
4	0x8234	A	--	--	--
5	0x8203	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 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*
	0x8207	B	--	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 Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-ID-characters*: *global-session-ID*
8	0x8238	B	Write size	--	--
9	0x8338	B	Size that could be written	--	• When normal: *Entrance-time* • For an exception: *Entrance-time-exception-name*
10	0x8306	B	Class name	Dispatch type 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	0x8302	A	Class name (JSP file name when JSP is invoked)	--	• 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*
12	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*
13	0x8334	A	--	--	*Entrance-time*
14	0x8235	A	Implementation class name for asynchronous processing	--	--
15	0x8335	A	Implementation class name for asynchronous processing	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
16	0x8239	A	--	--	--
17	0x8339	A	--	--	*Entrance-time*
18	0x8336	A	HTTP method	URI	*Entrance-time-status-code*

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-9.

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.

8.5 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.

8.5.1 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`, and `0x8206`.

Table 8–8: Details of trace collection points in a Web container (session trace)

Event ID	No. in the figure[#1]	Trace acquisition points	Level[#2]
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
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
0x8236	1	Immediately after a request is acquired, or when the request header analysis is complete	A
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 (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	12	Immediately after the completion of the processing of the servlet or JSP via RequestDispatcher	B

Event ID	No. in the figure[#1]	Trace acquisition points	Level[#2]
0x8307	10, 13	Immediately after the completion of the processing of the static contents (DefaultServlet)	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
0x8336	16	Immediately after the completion of request processing	A

Legend:

A: Standard

B: Advanced

A/B: Different information is collected for the *Standard* and *Advanced* levels.

#1

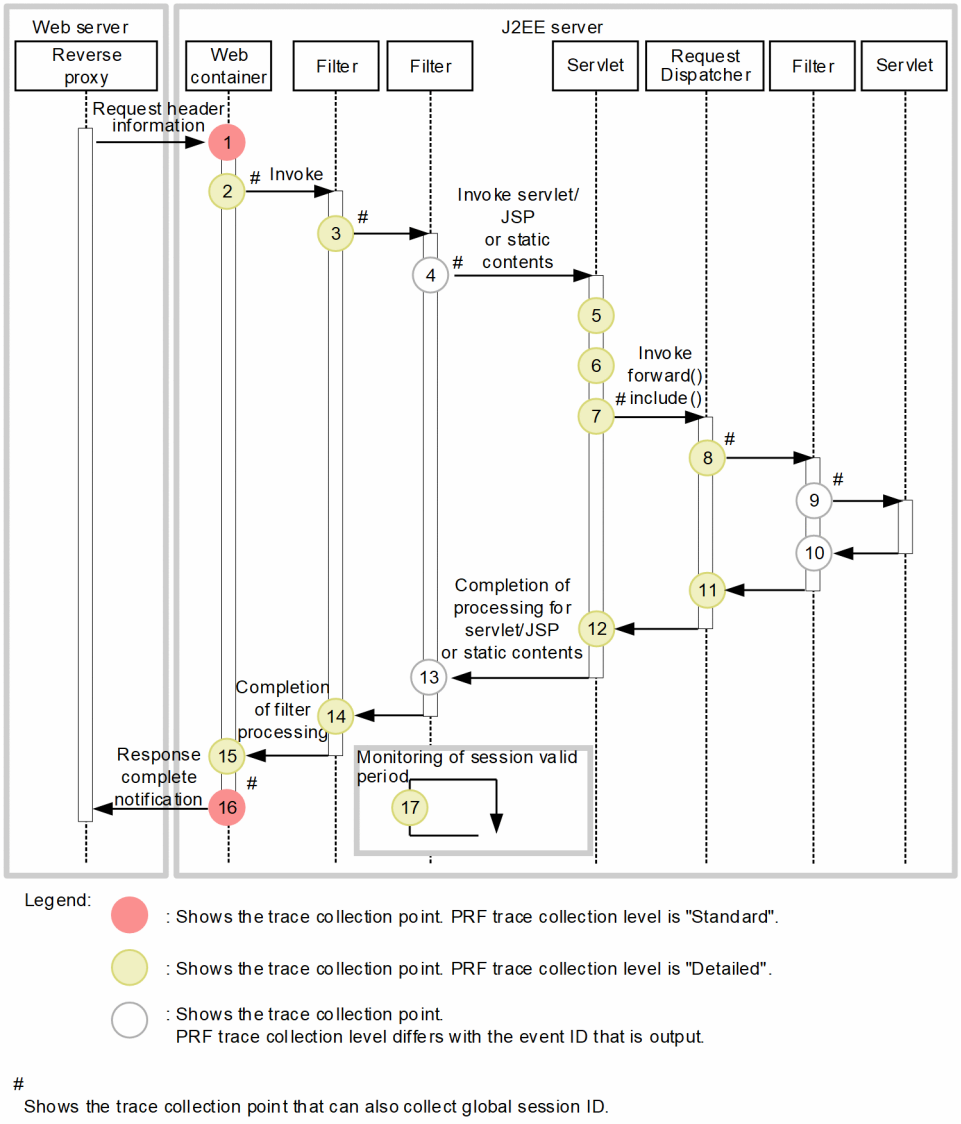
Corresponds to the numbers in Figure 8-10.

#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 8–10: Trace collection points of a Web container (session trace)



The session ID that can be acquired at each point is as follows:

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 17

An invalid session ID can be acquired only when a session that has exceeded the valid period is discarded.

8.5.2 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`, and `0x8215`.

Table 8–9: 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	0x8236	A	HTTP method	URI	Value for the request header name specified in the property (none if the header name is not specified)
4, 9	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*
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 Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-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*

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
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	0x8336	A	HTTP method	URI	*Entrance-time status-code*
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: *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*
11	0x8314	B	Class name	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*
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*
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 8-10.

8.6 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.

8.6.1 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 8–10: Details of trace collection points in a Web container during normal termination (filter trace)

Event ID	No. in the figure#	Trace acquisition points	Level
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
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
0x8236	1	Immediately after a request is acquired, or when the request header analysis is complete	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	B

Event ID	No. in the figure#	Trace acquisition points	Level
		(filter for which `REQUEST` is specified in the `<dispatcher>` tag of the `<filter-mapping>` tag of `web.xml`)	
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
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
0x8336	12	Immediately after the completion of request processing	A

Legend:

A: Standard

B: Advanced

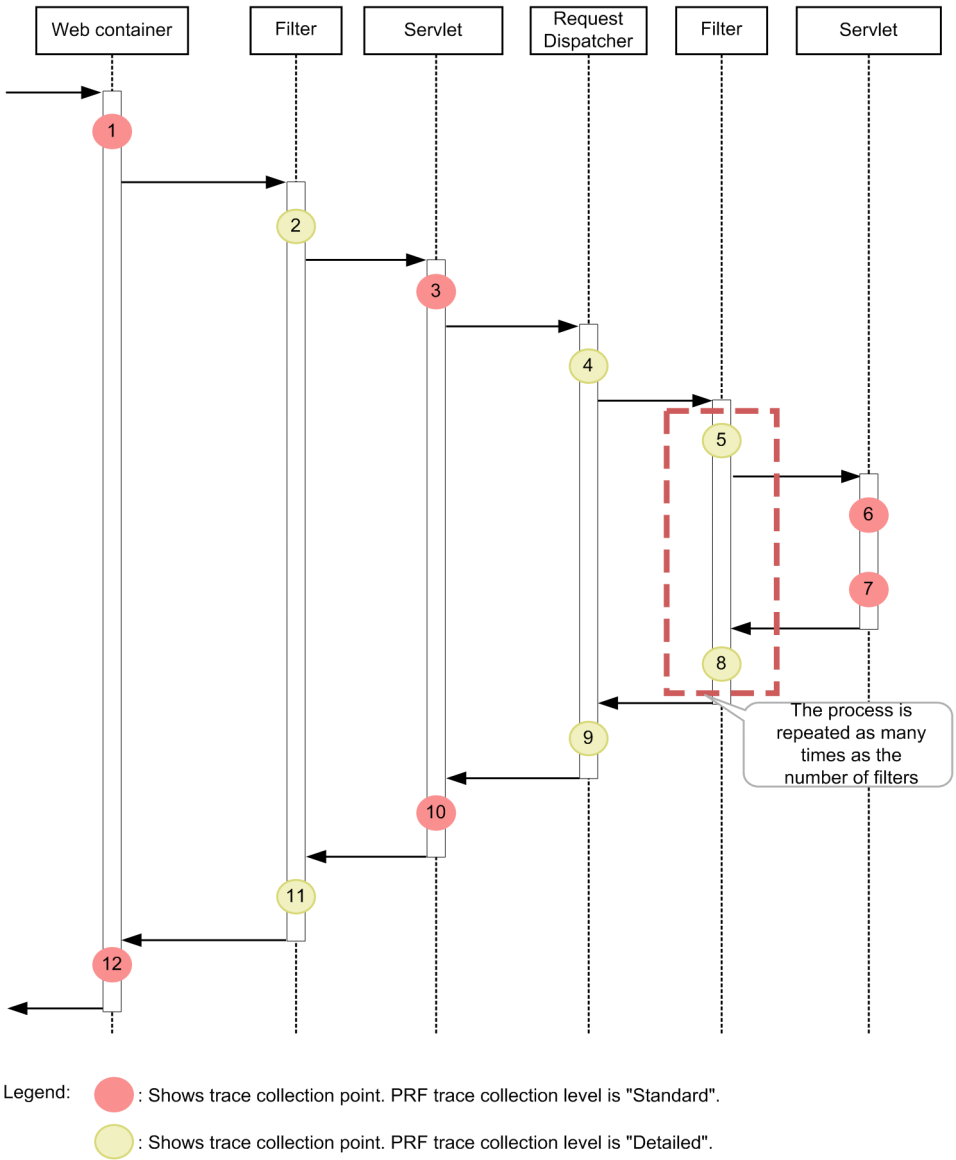
A/B: Different information is collected for the *Standard* and *Advanced* levels.

#

Corresponds to the numbers in Figure 8-11.

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 8–11: Trace collection points in a Web container during normal termination (filter trace)



(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 8–11: 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	0x8236	A	HTTP method	URI	Value for the request header name specified in the property (none if the header name is not specified)
2	0x8203	B	Class name	Context root name	*Number-of-session-ID-characters*: *session-ID*: *number-of-global-session-*

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
					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*
		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*

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*
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: *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	0x8336	A	HTTP method	URI	*Entrance-time status-code*

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-11.

8.6.2 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 8–12: Details of trace collection points in a Web container when an exception occurs (filter trace)

Event ID	No. in the figure#	Trace acquisition points	Level
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 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	6	Immediately before a servlet or JSP is invoked via RequestDispatcher	B
0x8207	8	Immediately before the static contents are invoked (DefaultServlet)	B
0x8216	7	Immediately before the filter that is executed during transfer to the error page is invoked (filter for which ERROR is specified in the <dispatcher> tag of the <filter-mapping> tag of web.xml)	B
0x8236	1	Immediately after a request is acquired, or when the request header analysis is complete	A
0x8302	4, 9	Immediately after completion of the processing of the servlet or JSP	A/B
0x8303	5	Immediately after completion of the 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
0x8306	11	Immediately after completion of the processing of the servlet or JSP via RequestDispatcher	B
0x8307	9	Immediately after completion of the processing of the static contents (DefaultServlet)	B
0x8316	10	Immediately after completion of the processing of the filter executed during transfer to the error page (filter for which ERROR is specified in the <dispatcher> tag of the <filter-mapping> tag of web.xml)	B
0x8336	12	Immediately after the completion of request processing	A

Legend:

A: Standard

B: Advanced

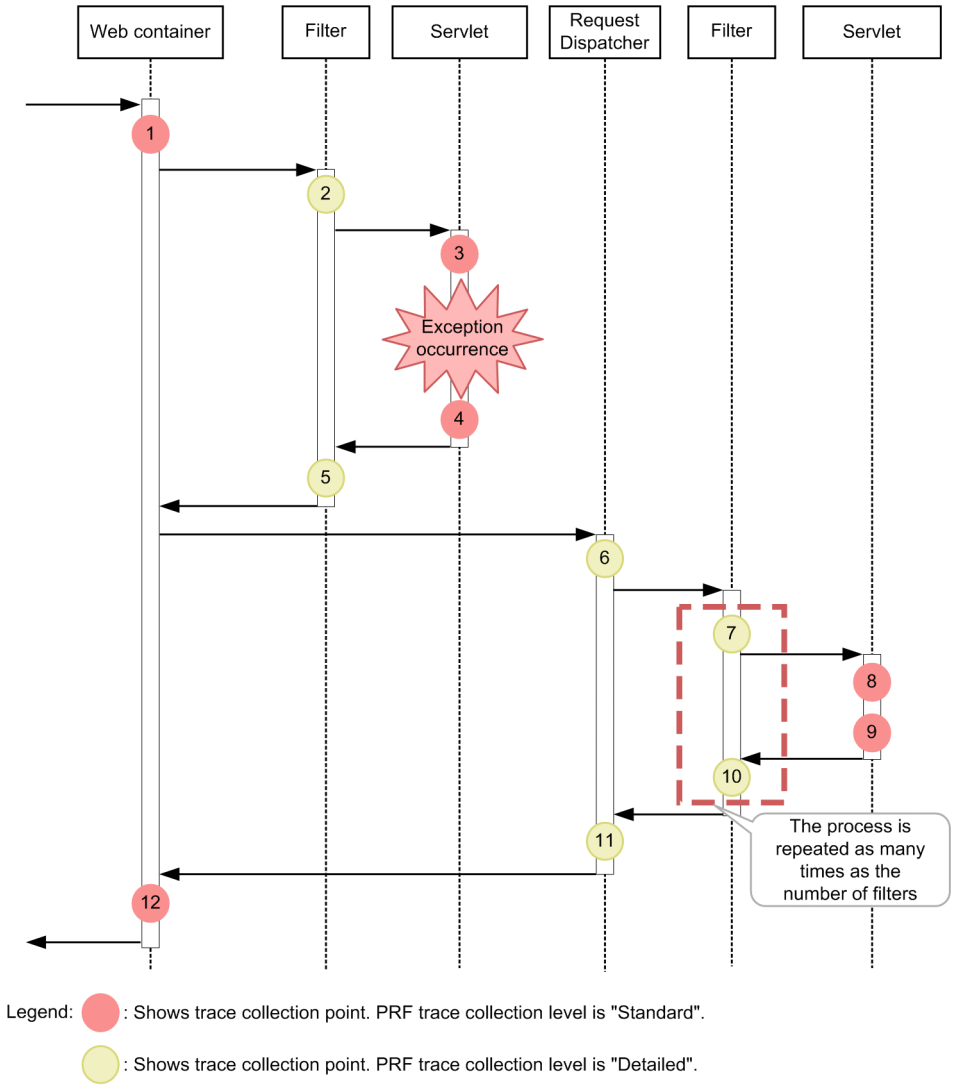
A/B: Different information is collected for the *Standard* and *Advanced* levels.

#

Corresponds to the numbers in Figure 8-12.

The following figure shows the trace collection points in a Web container when an exception occurs.

Figure 8–12: Trace collection points in a Web container when an exception occurs



(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 8–13: 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	0x8236	A	HTTP method	URI	Value for the request header name specified in the property (none if the header name is not specified)
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*
8	0x8202	A	Class name (JSP file name when a JSP is invoked)	--	--
		B		Context root name	*Number-of-session-ID-characters*: *session-*

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
					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	0x8336	A	HTTP method	URI	*Entrance-time status-code*

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-12.

8.7 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.

8.7.1 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 subsection 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 8–14: 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
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
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
0x8236	1	Immediately after a request is acquired, or when the request header analysis is complete	A
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
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

Event ID	No. in the figure#	Trace acquisition points	Level
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
0x8336	10	Immediately after the completion of request processing	A

Legend:

A: Standard

B: Advanced

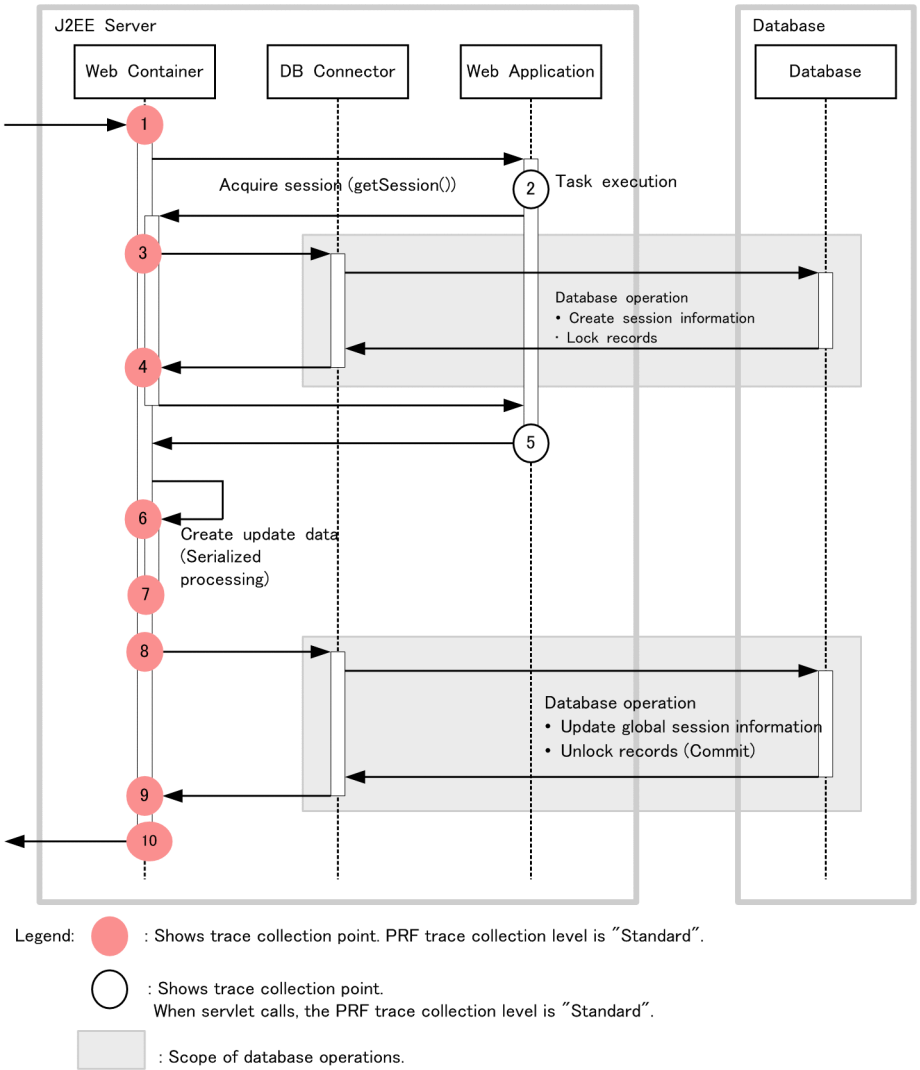
A/B: Different information is collected for the *Standard* and *Advanced* levels.

#

Corresponds to the numbers in Figure 8-13.

The following figure shows the trace collection points.

Figure 8–13: Trace collection points of the request processing for creating an HTTP session (database session failover functionality)



(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 8–15: 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	0x8236	A	HTTP method	URI	Value for the request header name specified in the property (none if the header name is not specified)
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:

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*
	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*
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	0x8336	A	HTTP method	URI	*Entrance-time status-code*

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-13.

8.7.2 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 subsection 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 8–16: 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
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
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
0x8222[#2]	10	Immediately before starting database access after the Web application processing with the database session failover functionality	A
0x8236	1	Immediately after a request is acquired, or when the request header analysis is complete	A
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
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

Event ID	No. in the figure[#1]	Trace acquisition points	Level
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
0x8336	12	Immediately after the completion of request processing	A

Legend:

A: Standard

B: Advanced

A/B: Different information is collected for the *Standard* and *Advanced* levels.

#1

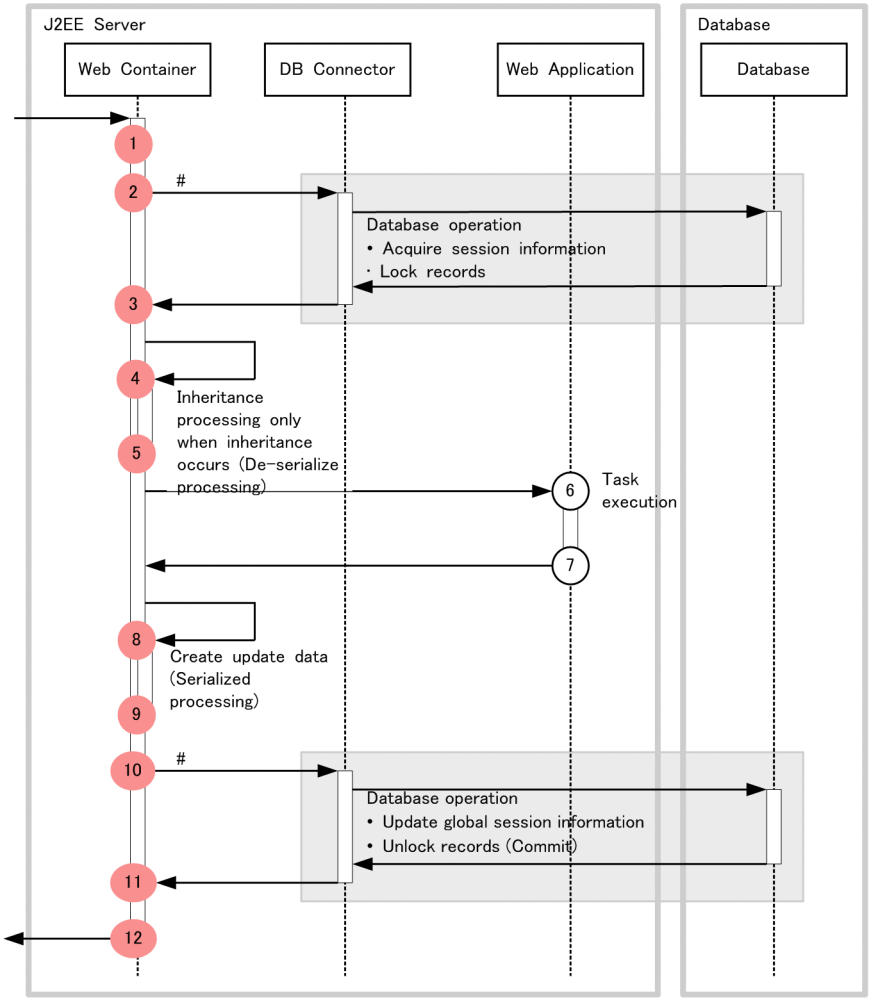
Corresponds to the numbers in Figure 8-14.

#2

Not output for requests meant for referencing the HTTP session.

The following figure shows the trace collection points.

Figure 8–14: Trace collection points of the request processing for updating an HTTP session (database session failover functionality)



Legend:
: Shows trace collection point. PRF trace collection level is "Standard".
: Shows trace collection point.
When servlet calls, the PRF trace collection level is "Standard".
: Scope of database operations.

The actual DB Connector call occurs 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 8–17: 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	0x8236	A	HTTP method	URI	Value for the request header name specified in the property (none if the header name is not specified)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
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: *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:

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*
	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	0x8336	A	HTTP method	URI	*Entrance-time status-code*

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-14.

8.7.3 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 subsection 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 8–18: 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
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
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
0x8236	1	Immediately after a request is acquired, or when the request header analysis is complete	A
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
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
0x8336	10	Immediately after the completion of request processing	A

Legend:

A: Standard

B: Advanced

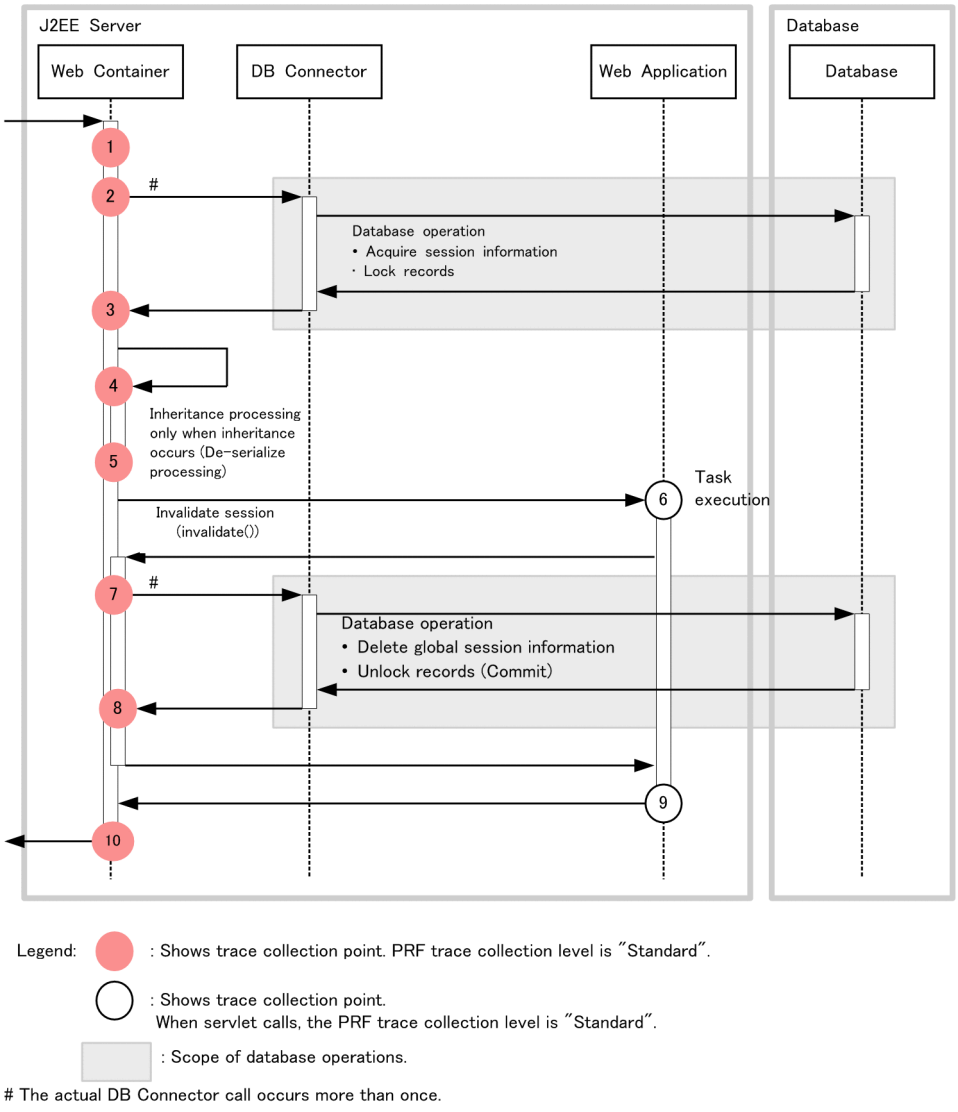
A/B: Different information is collected for the *Standard* and *Advanced* levels.

#

Corresponds to the numbers in Figure 8-15.

The following figure shows the trace collection points.

Figure 8–15: Trace collection points of the request processing for disabling an HTTP session (database session failover functionality)



(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 8–19: 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	0x8236	A	HTTP method	URI	Value for the request header name specified in the property (none if the header name is not specified)
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 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
			When integrity protection mode is disabled -1		
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*
	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	0x8336	A	HTTP method	URI	*Entrance-time status-code*

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-15.

8.7.4 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 subsection 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 8–20: 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

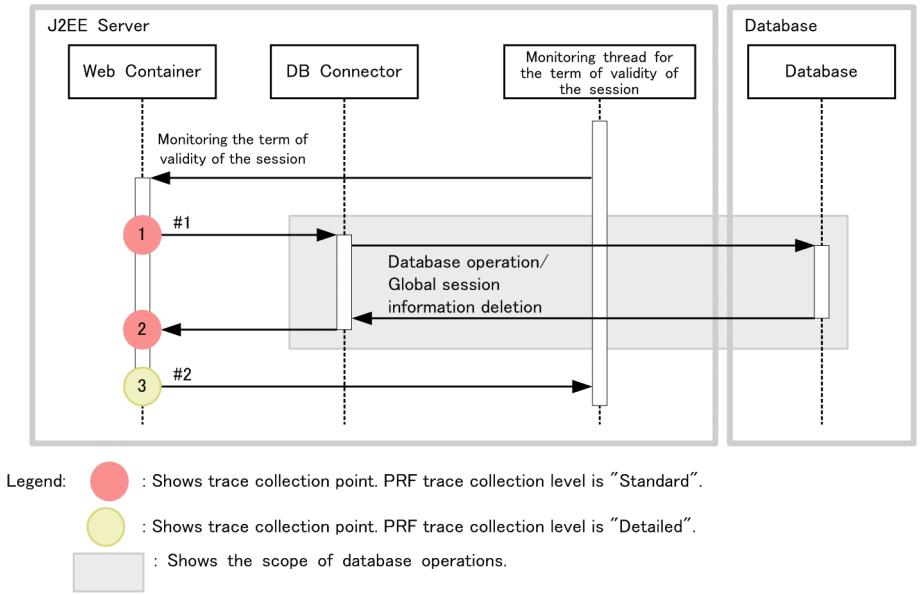
B: Advanced

#

Corresponds to the numbers in Figure 8-16.

The following figure shows the trace collection points.

Figure 8–16: Trace collection points of the request processing for disabling an HTTP session through valid period monitoring (database session failover functionality)



(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 8–21: 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 8-16.

#2

Not output when the integrity protection mode is disabled.

8.8 Trace collection points of an EJB container

This section describes the trace collection points of an EJB container and also the trace information that can be collected, in different cases of a Session Bean or Entity Bean, Message-driven Bean, or during the use of the Timer Service, and in the case of occurrence of method cancellation.

8.8.1 In the case of a Session Bean or Entity Bean

This subsection describes the trace collection points of a Session Bean or Entity Bean, and the trace information that can be collected.

(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 8–22: Details of trace collection points in a Session Bean or Entity Bean

Event ID	No. in the figure#	Trace acquisition points		Level
0x8401	1	In the case of a remote home interface	Immediately after the EJB container receives a request	A
0x8402	4		Immediately before the EJB container sends a response	A
0x8403	1	In the case of a local home interface	Immediately after the EJB container receives a request	A
0x8404	4		Immediately before the EJB container sends a response	A
0x8405	1	In the case of a remote component interface	Immediately after the EJB container receives a request	A
0x8406	4		Immediately before the EJB container sends a response	A
0x8407	1	In the case of a local component interface	Immediately after the EJB container receives a request	A
0x8408	4		Immediately before the EJB container sends a response	A
0x8409	2	Immediately before the EJB container calls back the business method of the EJB		B
0x840A	3	Immediately after returning from the callback of the EJB business method		B
0x8453	2	Immediately before the EJB container calls back the home method of the EJB		B
0x8454	3	Immediately after returning from the callback of the EJB home method		B
0x8470	1	In the case of a remote business interface	Immediately after the EJB container receives a request	A
0x8471	4		Immediately before the EJB container sends a response	A
0x8472	1	In the case of a local business interface	Immediately after the EJB container receives a request	A

Event ID	No. in the figure#	Trace acquisition points		Level
0x8473	4		Immediately before the EJB container sends a response	A
0x8474	1	In the case of a remote business interface	Immediately after the EJB container receives a request for creating an instance of the Stateful Session Bean via remote business interface	A
0x8475	1		Immediately before the EJB container sends a response for the request to create an instance of the Stateful Session Bean via remote business interface	A
0x8476	1	In the case of a local business interface	Immediately after the EJB container receives a request for creating an instance of the Stateful Session Bean via local business interface	A
0x8477	1		Immediately before the EJB container sends a response for the request to create an instance of the Stateful Session Bean via local business interface	A

Legend:

A: Standard

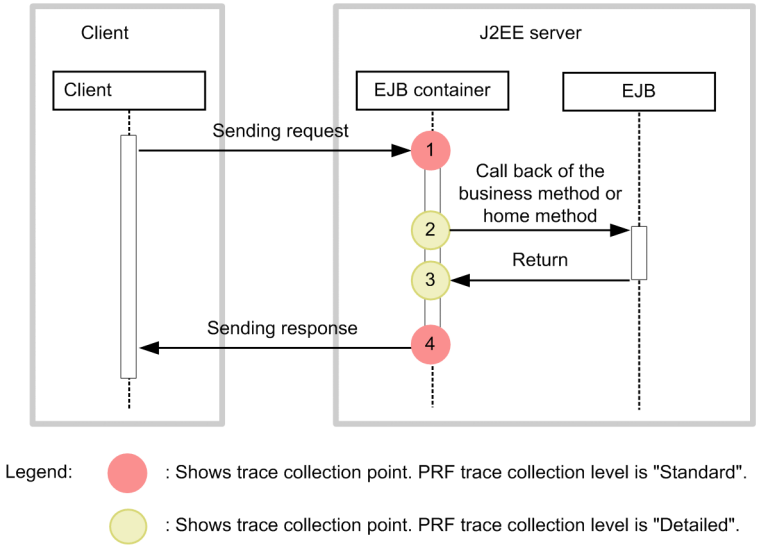
B: Advanced

#

Corresponds to the numbers in Figure 8-17.

The following figure shows the trace collection points in a Session Bean or Entity Bean.

Figure 8–17: Trace collection points of a Session Bean or Entity Bean



(2) Trace information that can be collected

The following table describes the trace information that can be collected in a Session Bean or Entity Bean.

Table 8–23: Trace information that can be collected in a Session Bean or Entity Bean

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8401	A	Bean name	Method name, number of arguments	--
	0x8403	A	Bean name	Method name, number of arguments	--
	0x8405	A	Bean name	Method name, number of arguments	--
	0x8407	A	Bean name	Method name, number of arguments	--
	0x8470	A	Bean name	Method name, number of arguments	--
	0x8472	A	Bean name	Method name, number of arguments	--
	0x8474	A	Bean name	--	--
	0x8475	A	Bean name	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
	0x8476	A	Bean name	--	--
	0x8477	A	Bean name	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
2	0x8409	B	Bean name	Method name, number of arguments	--
	0x8453	B	Bean name	Method name, number of arguments	--
3	0x840A	B	Bean name	Method name, number of arguments	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
	0x8454	B	Bean name	Method name, number of arguments	
4	0x8402	A	Bean name	Method name, number of arguments	
	0x8404	A	Bean name	Method name, number of arguments	
	0x8406	A	Bean name	Method name, number of arguments	
	0x8408	A	Bean name	Method name, number of arguments	
	0x8471	A	Bean name	Method name, number of arguments	
	0x8473	A	Bean name	Method name, number of arguments	

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-17.

8.8.2 In the Case of Message-driven Bean (EJB2.0)

This subsection describes the trace collection points of a Message-driven Bean (EJB2.0), and also the trace information that can be collected.

(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 8–24: Details of trace collection points in a Message-driven Bean (EJB2.0)

Event ID	No. in the figure#	Trace acquisition points	Level
0x8425	1	When the message processing in the EJB container starts	A
0x8426	4	When the message processing in the EJB container ends	A
0x8427	2	Immediately before the EJB container call backs the `onMessage` method of the Message-driven Bean	B
0x8428	3	Immediately after returning from the callback of the `onMessage` method of the Message-driven Bean	B

Legend:

A: Standard

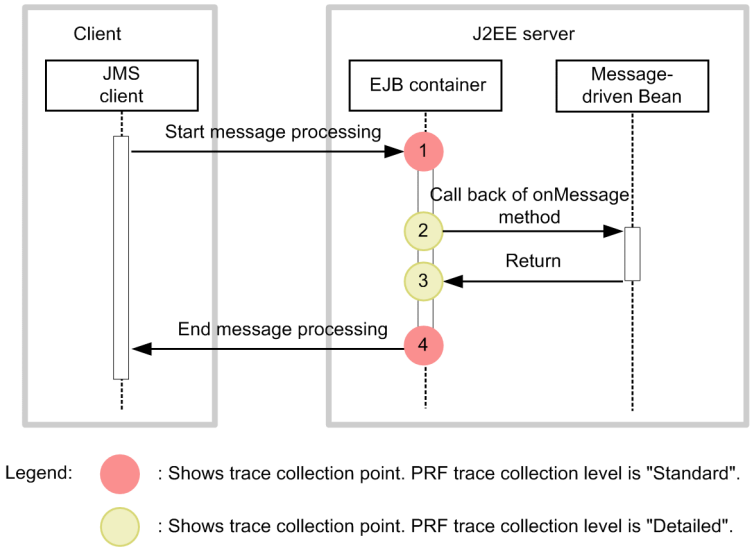
B: Advanced

#

Corresponds to the numbers in Figure 8-18.

The following figure shows the trace collection points in a Message-driven Bean (EJB2.0).

Figure 8–18: Trace collection points of a Message-driven Bean (EJB2.0)



(2) Trace information that can be collected

The following table describes the trace information that can be collected in a Message-driven Bean (EJB2.0).

Table 8–25: Trace information that can be collected in a Message-driven Bean (EJB2.0)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8425	A	Bean name	--	--
2	0x8427	B	--	--	--
3	0x8428	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
4	0x8426	A	Bean name	--	

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-18.

8.8.3 In the case of a Message-driven Bean (EJB2.1 and later)

This subsection describes the trace collection points of a Message-driven Bean (EJB2.1 and later), and also the trace information that can be collected.

(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 8–26: Details of trace collection points in a Message-driven (EJB2.1 and later)

Event ID	No. in the figures#	Trace acquisition points	Level
0x842D	1	Immediately after invoking the `beforeDelivery()` method of the Message-driven Bean	A
0x842E	2	Immediately before the return the `beforeDelivery()` method of the Message-driven Bean	A
0x842F	3	Immediately after invoking the method of message listener of the Message-driven Bean from the resource adapter	A
0x8431	4	Immediately before the EJB container calls back the method of message listener of the Message-driven Bean	A
0x8432	5	Immediately after returning from the callback of the method of message listener of the Message-driven Bean	A
0x8430	6	Immediately before the return of the method of message listener of the Message-driven Bean that is invoked from the resource adapter	A
0x8433	7	Immediately after invoking the `beforeDelivery()` method of the Message-driven Bean	A

Event ID	No. in the figures#	Trace acquisition points	Level
0x8434	8	Immediately before the return the `beforeDelivery()` method of the Message-driven Bean	A

Legend:

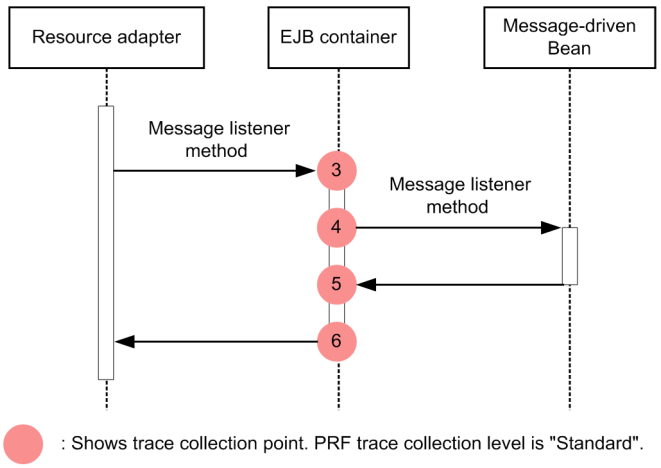
A: Standard

#

Corresponds to the numbers in Figure 8-19 and Figure 8-20.

The following figure shows the trace collection points in a Message-driven Bean (EJB2.1 and later).

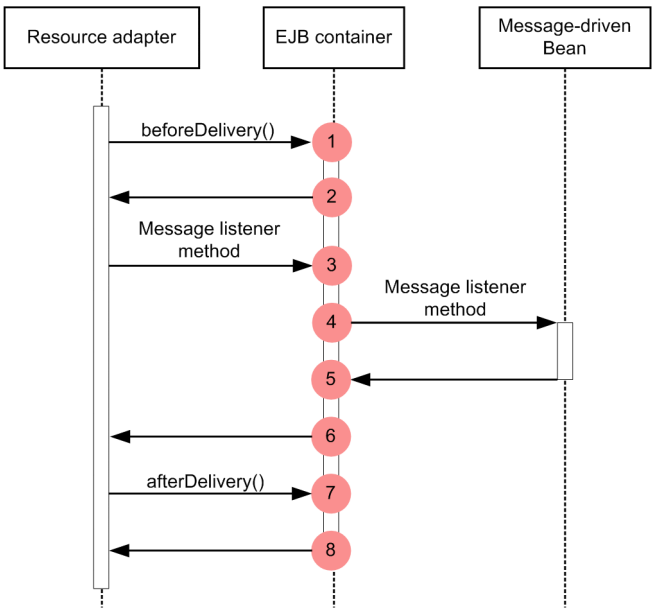
Figure 8–19: Trace collection points of a Message-driven Bean (EJB2.1 and later) (in the case of optionA#)



Legend: : Shows trace collection point. PRF trace collection level is "Standard".

# Indicates the message delivery option described in Connector 1.5 specifications.

Figure 8–20: Trace collection points of a Message-driven Bean (EJB2.1 and later) (in the case of optionB#)



Legend: : Shows trace collection point. PRF trace collection level is "Standard".

# Indicates the message delivery option described in Connector 1.5 specifications.

(2) Trace information that can be collected

The following table describes the trace information that can be collected in a Message-driven Bean (EJB2.1 and later).

Table 8–27: Trace information that can be collected in a Message-driven Bean (EJB2.1 and later)

No. in the figures#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x842D	A	Bean name	--	--
2	0x842E	A	Bean name	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
3	0x842F	A	Bean name	Method name	--
4	0x8431	A	Bean name	Method name	--
5	0x8432	A	Bean name	Method name	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
6	0x8430	A	Bean name	Method name	
7	0x8433	A	Bean name	--	--
8	0x8434	A	Bean name	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-19 and Figure 8-20.

8.8.4 For Timer Service

This subsection describes the trace collection points of the Timer Service, and the trace information that can be collected.

(1) For createTimer

(a) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–28: Details of trace collection points in Timer Service (for createTimer)

Event ID	No. in the figure#	Trace collection points	Level
0x8460	1	When the processing of `TimerService.createTimer(Date initialExpiration, long intervalDuration, Serializable info)` starts	A
0x8462	1	When the processing of `TimerService.createTimer(Date expiration, Serializable info)` starts	A
0x8464	1	When the processing of `TimerService.createTimer(long initialDuration, long intervalDuration, Serializable info)` starts	A
0x8466	1	When the processing of `'TimerService.createTimer(long duration, Serializable info)` starts	A
0x8461	2	When the processing of `TimerService.createTimer(Date initialExpiration, long intervalDuration, Serializable info)` ends	A
0x8463	2	When the processing of `TimerService.createTimer(Date expiration, Serializable info)` ends	A
0x8465	2	When the processing of `'TimerService.createTimer(long initialDuration, long intervalDuration, Serializable info)` ends	A
0x8467	2	When the processing of `TimerService.createTimer(long duration, Serializable info)` ends	A

Legend:

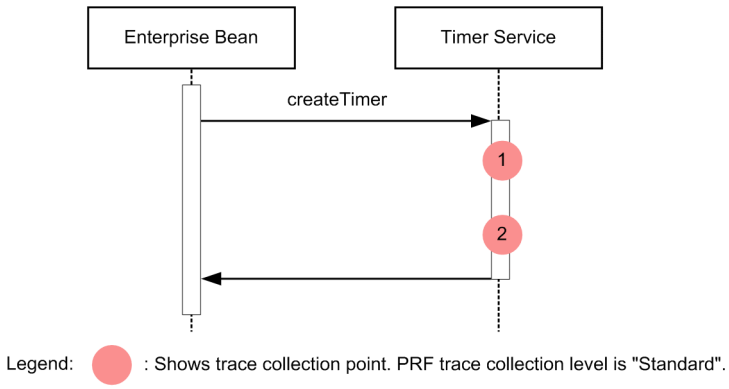
A: Standard

#

Corresponds to the numbers in Figure 8-21.

The following figure shows the trace collection points of Timer Service for createTimer.

Figure 8–21: Trace collection points of Timer Service (for createTimer)



(b) Trace information that can be collected

The following table describes the trace information that can be collected in Timer Service for createTimer.

Table 8–29: Trace information that can be collected in Timer Service (for createTimer)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8460	A	Class name of the Bean to be called back	--	Argument information
1	0x8462	A	Class name of the Bean to be called back	--	Argument information
1	0x8464	A	Class name of the Bean to be called back	--	Argument information
1	0x8466	A	Class name of the Bean to be called back	--	Argument information
2	0x8461	A	Class name of the Bean to be called back	--	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*
2	0x8463	A	Class name of the Bean to be called back	--	
2	0x8465	A	Class name of the Bean to be called back	--	
2	0x8467	A	Class name of the Bean to be called back	--	

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-21.

(2) For cancel

(a) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–30: Details of the trace collection points in Timer Service (for cancel)

Event ID	No. in the figure#	Trace collection points	Level
0x8468	1	When the processing of `javax.ejb.Timer.cancel()` starts	A
0x8469	2	When the processing of `javax.ejb.Timer.cancel()` ends	A

Legend:

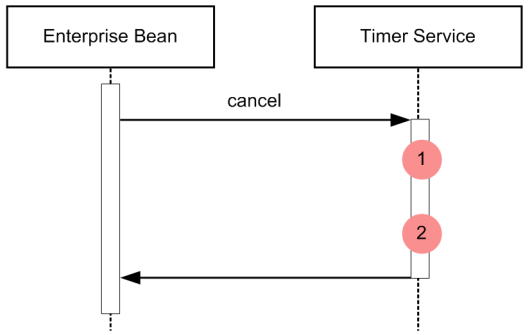
A: Standard

#

Corresponds to the numbers in Figure 8-22.

The following figure shows the trace collection points of Timer Service for `cancel`.

Figure 8–22: Trace collection points of Timer Service (for cancel)



Legend: : Shows trace collection point. PRF trace collection level is "Standard".

(b) Trace information that can be collected

The following table describes the trace information that can be collected in Timer Service for `cancel`.

Table 8–31: Trace information that can be collected in Timer Service (for cancel)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8468	A	Class name of the Bean to be called back	--	--
2	0x8469	A	Class name of the Bean to be called back	--	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-22.

(3) In the case of a callback

(a) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–32: Details of the trace collection points in Timer Service (for a callback)

Event ID	No. in the figure#	Trace collection points	Level
0x846C	1	When the thread starts the callback processing	B
0x846A	2	When the callback of the timeout callback method of the Enterprise Bean starts	A
0x846B	3	When the callback of the timeout callback method of the Enterprise Bean ends	A
0x846D	4	When the thread terminates the callback processing	B

Legend:

A: Standard

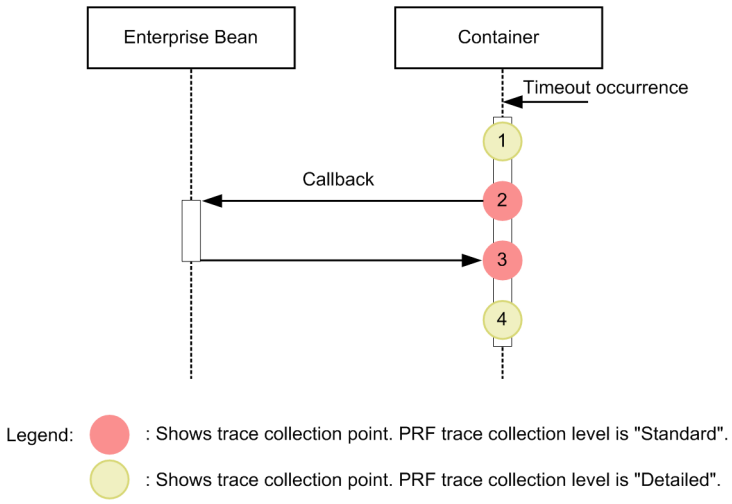
B: Advanced

#

Corresponds to the numbers in Figure 8-23.

The following figure shows the trace collection points of Timer Service for a callback.

Figure 8–23: Trace collection points of Timer Service (for a callback)



(b) Trace information that can be collected

The following table describes the trace information that can be collected in Timer Service for a callback.

The information collected during callback of the timeout method is output to the root application information. Furthermore, `0` is output to the client application information.

Table 8–33: Trace information that can be collected in Timer Service (for a callback)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x846C	B	Class name of the Bean to be called back	--	--
2	0x846A	A	Class name of the Bean to be called back	--	--
3	0x846B	A	Class name of the Bean to be called back	--	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*
4	0x846D	B	Class name of the Bean to be called back	--	

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-23.

(4) In the case of createSingleActionTimer

(a) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–34: Details of the trace collections points in Timer Service (for createSingleActionTimer)

Event ID	No. in the figure#	trace collection point	Level
0x84A0	1	When the processing of `TimerService.createSingleActionTimer(long duration, TimerConfig timerConfig)` starts	A
0x84A1	2	When the processing of `TimerService.createSingleActionTimer(long duration, TimerConfig timerConfig)` ends	A
0x84A2	1	When the processing of `TimerService.createSingleActionTimer(Date expiration, TimerConfig timerConfig)` starts	A
0x84A3	2	When the processing of `TimerService.createSingleActionTimer(Date expiration, TimerConfig timerConfig)` ends	A

Legend:

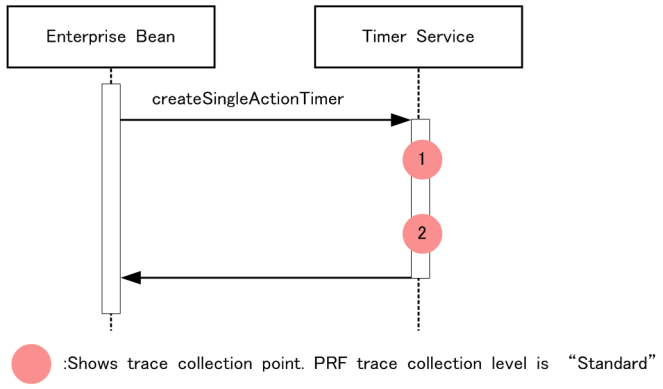
A: Standard

#

Corresponds to the numbers in Figure 8-24.

The following figure shows the trace collection points in `createSingleActionTimer`.

Figure 8–24: Trace collection points of Timer Service (for createSingleActionTimer)



(b) trace information that can be collected

The following table describes the trace information that can be collected in `createSingleActionTimer`.

Table 8–35: Trace information that can be collected in Timer Service (for createSingleActionTimer)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x84A0	A	Class name of the Bean to be called back	--	Argument information

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
2	0x84A1	A	Class name of the Bean to be called back	--	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*
1	0x84A2	A	Class name of the Bean to be called back	--	Argument information
2	0x84A3	A	Class name of the Bean to be called back	--	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-24.

(5) In the case of createIntervalTimer

(a) Trace collection points and PRF trace

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–36: Details of the trace collections points in Timer Service (for createIntervalTimer)

Event ID	No. in the figure#	Trace collection points	Level
0x84A4	1	When the processing of `TimerService.createIntervalTimer(long initialDuration, long intervalDuration, TimerConfig timerConfig)` starts	A
0x84A5	2	When the processing of `TimerService.createIntervalTimer(long initialDuration, long intervalDuration, TimerConfig timerConfig)` ends	A
0x84A6	1	When the processing of `TimerService.createIntervalTimer(Date initialExpiration, long intervalDuration, TimerConfig timerConfig)` starts	A
0x84A7	2	When the processing of `TimerService.createIntervalTimer(Date initialExpiration, long intervalDuration, TimerConfig timerConfig)` ends	A

Legend:

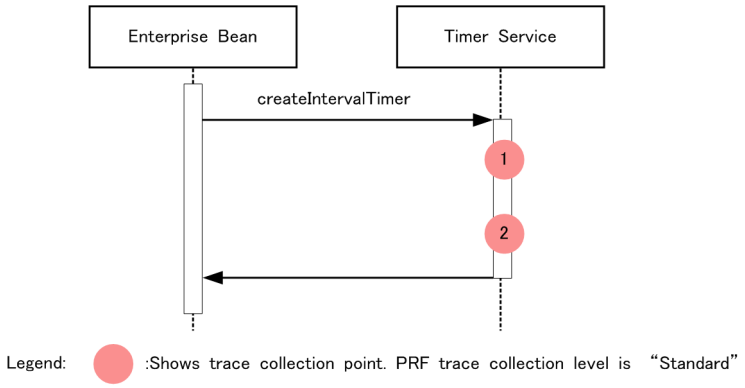
A: Standard

#

Corresponds to the numbers in Figure 8-25.

The following figure shows the trace collection points of `createIntervalTimer`.

Figure 8–25: Trace collection points of Timer Service (for createIntervalTimer)



(b) Trace information that can be collected

The following table describes the trace information that can be collected in `createIntervalTimer`.

Table 8–37: Trace information that can be collected in Timer Service (for createIntervalTimer)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x84A4	A	Class name of the Bean to be called back	--	Argument information
2	0x84A5	A	Class name of the Bean to be called back	--	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*
1	0x84A6	A	Class name of the Bean to be called back	--	Argument information
2	0x84A7	A	Class name of the Bean to be called back	--	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-25.

(6) In the case of createCalendarTimer

(a) Trace collection points and PRF trace

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–38: Details of the trace collections points in Timer Service (for createCalendarTimer)

Event ID	No. in the figure#	Trace collection points	Level
0x84A8	1	When the processing of `TimerService.createCalendarTimer(ScheduleExpression schedule)` starts	A
0x84A9	2	When the processing of `TimerService.createCalendarTimer(ScheduleExpression schedule)` ends	A
0x84AA	1	When the processing of `TimerService.createCalendarTimer(ScheduleExpression schedule, TimerConfig config)` starts	A
0x84AB	2	When the processing of `TimerService.createCalendarTimer(ScheduleExpression schedule, TimerConfig config)` ends	A

Legend:

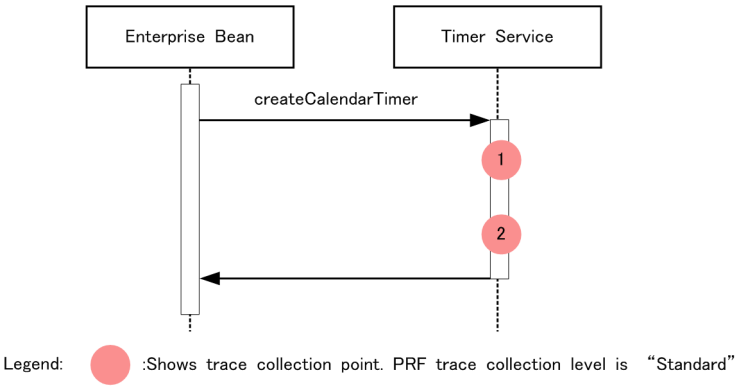
A: Standard

#

Corresponds to the numbers in Figure 8-26.

The following figure shows the trace collection points of `createCalendarTimer`.

Figure 8–26: Trace collection points of Timer Service (for createCalendarTimer)



(b) trace information that can be collected

The following table describes the trace information that can be collected in `createCalendarTimer`.

Table 8–39: Trace information that can be collected in Timer Service (for createCalendarTimer)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x84A8	A	Class name of the Bean to be called back	--	Argument information
2	0x84A9	A	Class name of the Bean to be called back	--	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x84AA	A	Class name of the Bean to be called back	--	Argument information
2	0x84AB	A	Class name of the Bean to be called back	--	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-26.

8.8.5 When the Session Bean is invoked asynchronously

This subsection describes the trace collection points when the Session Bean is invoked asynchronously, and the trace information that can be collected.

In an asynchronous invocation, the output order of the event IDs output by the same thread is guaranteed, but the output order of the event IDs output by different threads is not guaranteed. Therefore, the following events might occur:

- The output order of the event IDs might differ from the order described in this subsection.
- The event IDs of asynchronous invocation might be output between the event IDs of other functionality.

(1) When a Session Bean is invoked asynchronously from the local client

(a) Trace collection points and PRF trace

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–40: Details of the trace collection points when the Session Bean is invoked asynchronously from the local client

Event ID	No. in the figure#	Trace collection points	Level
0x8472	1	Immediately after the EJB container receives a request (for a local invocation)	A
0x8473	2	Immediately before the EJB container sends a response (for a local invocation)	A
0x84C0	3	Immediately before the EJB container calls back the asynchronous business method of EJB	A
0x84D6	4	Immediately before the EJB container starts the asynchronous business method of EJB (for a local invocation)	A
0x8409	5	Immediately before the EJB container calls back the EJB business method	B
0x840A	6	Immediately after returning from a callback of the EJB business method	B
0x84D7	7	Immediately after the asynchronous business method of EJB is terminated (for a local invocation)	A

Event ID	No. in the figure#	Trace collection points	Level
0x84C1	8	Immediately after returning from a callback of the asynchronous business method of EJB	A

Legend:

A: Standard

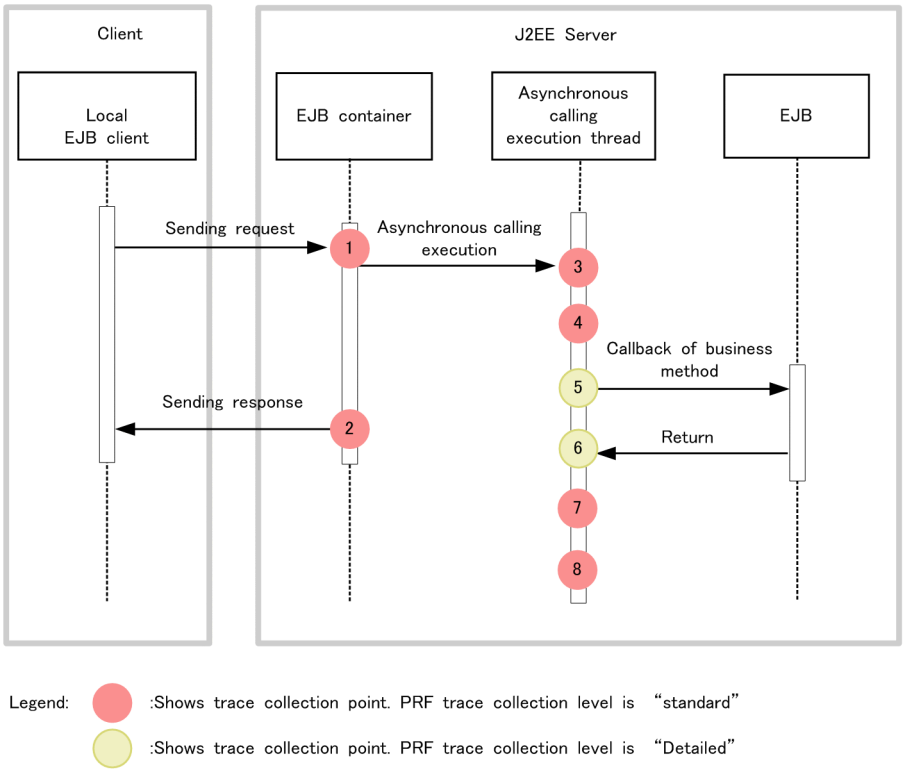
B: Advanced

#

Corresponds to the numbers in Figure 8-27.

The following figure shows the trace collection points when the Session Bean is invoked asynchronously from the local client.

Figure 8–27: Trace collection points when the Session Bean is invoked asynchronously from the local client



(b) Trace information that can be collected

The following table describes the trace information that can be collected when the Session Bean is invoked asynchronously from the local client.

Table 8–41: Trace information that can be collected when the Session Bean is invoked asynchronously from the local client

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Option
1	0x8472	A	Bean name	Method name, number of arguments	--
2	0x8473	A	Bean name	Method name, number of arguments	• When normal: *Entrance-time*

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Option
					• For an exception: *Entrance-timeexception-name*
3	0x84C0	A	Class name of the Bean to be called back	Method name, number of arguments	Root application information at the source of an asynchronous method invocation
4	0x84D6	A	Class name of the Bean to be called back	Method name, number of arguments	--
5	0x8409	B	Bean name	Method name, number of arguments	--
6	0x840A	B	Bean name	Method name, number of arguments	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*
7	0x84D7	A	Class name of the Bean to be called back	Method name, number of arguments	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*
8	0x84C1	A	Class name of the Bean to be called back	Method name, number of arguments	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-27.

(2) When the Session Bean is invoked asynchronously from the remote client

(a) Trace collection points and PRF trace

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–42: Details of the trace collection points when the Session Bean is invoked asynchronously from the remote client

Event ID	No. in the figure#	Trace collection points	Level
0x8470	1	Immediately after the EJB container receives a request (for a remote invocation)	A

Event ID	No. in the figure#	Trace collection points	Level
0x8471	2	Immediately before the EJB container sends a response (for a remote invocation)	A
0x84C0	3	Immediately before the EJB container calls back the asynchronous business method of EJB	A
0x84D8	4	Immediately before the EJB container starts the asynchronous business method of EJB (for a remote invocation)	A
0x8409	5	Immediately before the EJB container calls back the EJB business method	B
0x840A	6	Immediately after returning from a callback of the EJB business method	B
0x84D9	7	Immediately after the asynchronous business method of EJB is terminated (for a remote invocation)	A
0x84C1	8	Immediately after returning from a callback of the asynchronous business method of EJB	A

Legend:

A: Standard

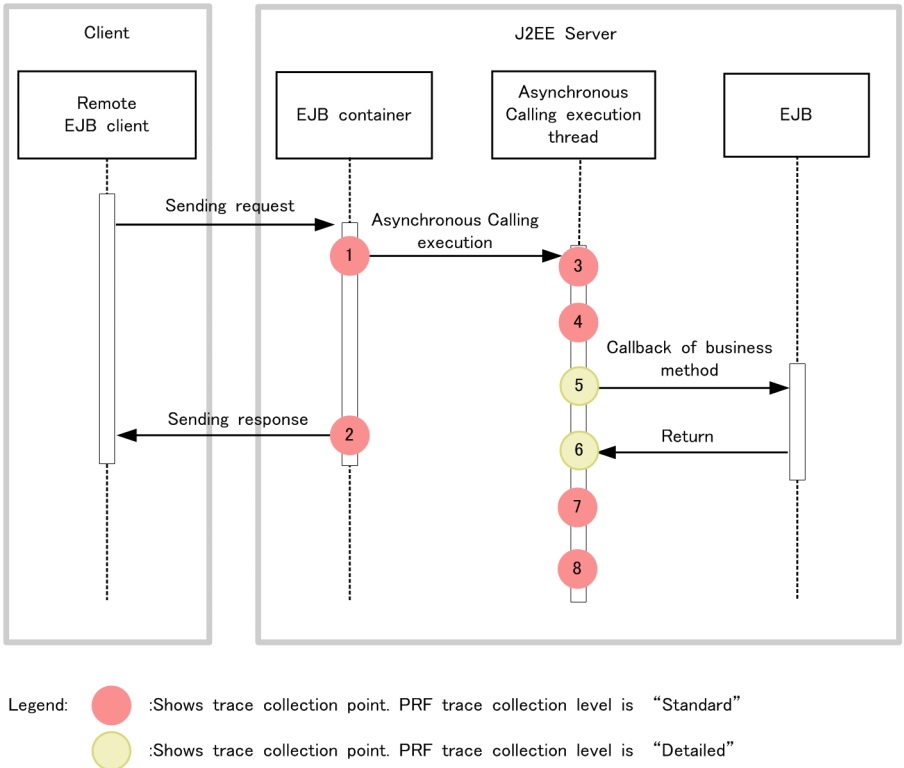
B: Advanced

#

Corresponds to the numbers in Figure 8-28.

The following figure shows the trace collection points when the Session Bean is invoked asynchronously from the remote client.

Figure 8–28: Trace collection points when the Session Bean is invoked asynchronously from the remote client



(b) Trace information that can be collected

The following table describes the trace information that can be collected when the Session Bean is invoked asynchronously from the remote client.

Table 8–43: Trace information that can be collected when the Session Bean is invoked asynchronously from the remote client

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Option
1	0x8470	A	Bean name	Method name, number of arguments	--
2	0x8471	A	Bean name	Method name, number of arguments	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*
3	0x84C0	A	Class name of the Bean to be called back	Method name, number of arguments	Root application information at the source of an asynchronous method invocation
4	0x84D8	A	Class name of the Bean to be called back	Method name, number of arguments	--
5	0x8409	B	Bean name	Method name, number of arguments	--
6	0x840A	B	Bean name	Method name, number of arguments	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*
7	0x84D9	A	Class name of the Bean to be called back	Method name, number of arguments	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*
8	0x84C1	A	Class name of the Bean to be called back	Method name, number of arguments	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-28.

(3) When get is invoked from the local client

(a) Trace collection points and PRF trace

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–44: Details of the trace collection points when get is invoked from the local client

Event ID	No. in the figure#	Trace collection points	Level
0x84C2	1	When the processing of `Future.get()` starts (for a local invocation)	B
0x84C3	2	When the processing of `Future.get()` ends (for a local invocation)	B
0x84C4	1	When the processing of `Future.get(long timeout, TimeUnit unit)` starts (for a local invocation)	B
0x84C5	2	When the processing of `Future.get(long timeout, TimeUnit unit)` ends (for a local invocation)	B

Legend:

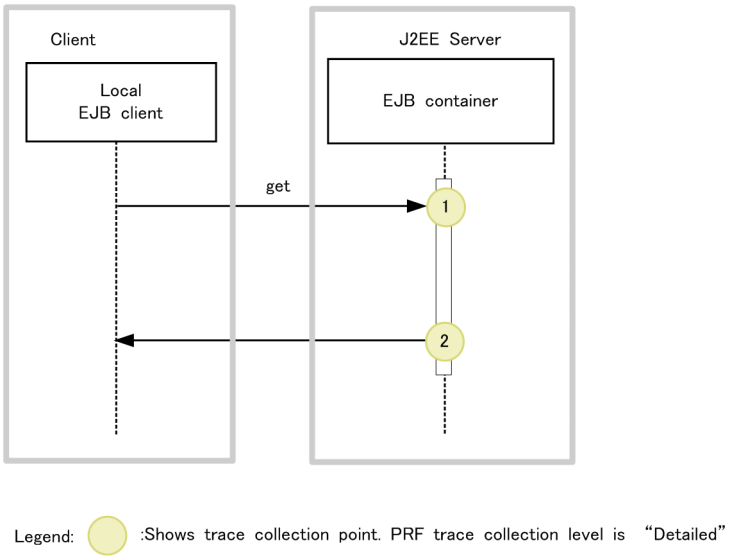
B: Advanced

#

Corresponds to the numbers in Figure 8-29.

The following figure shows the trace collection points when `get` is invoked from the local client.

Figure 8–29: Trace collection points when get is invoked from the local client



(b) Trace information that can be collected

The following table describes the trace information that can be collected when `get` is invoked from the local client.

Table 8–45: Trace information that can be collected when get is invoked from the local client

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Option
1	0x84C2	B	--	--	--
2	0x84C3	B	--	--	• When normal:

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Option
					Entrance-time • For an exception: *Entrance-timeexception-name*
1	0x84C4	B	--	--	Method name, argument information
2	0x84C5	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*

Legend:

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-29.

(4) When get is invoked from the remote client

(a) Trace collection points and PRF trace

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–46: Details of the trace collection points when get is invoked from the remote client

Event ID	No. in the figure#	Trace collection points	Level
0x84C6	1	When the processing of `Future.get()` starts (for a remote invocation)	B
0x84C2	2	When the processing of `Future.get()` starts (for a local invocation)	B
0x84C3	3	When the processing of `Future.get()` ends (for a local invocation)	B
0x84C7	4	When the processing of `Future.get()` ends (for a remote invocation)	B
0x84C8	1	When the processing of `Future.get(long timeout, TimeUnit unit)` starts (for a remote invocation)	B
0x84C4	2	When the processing of `Future.get(long timeout, TimeUnit unit)` starts (for a local invocation)	B
0x84C5	3	When the processing of `Future.get(long timeout, TimeUnit unit)` ends (for a local invocation)	B
0x84C9	4	When the processing of `Future.get(long timeout, TimeUnit unit)` ends (for a remote invocation)	B

Legend:

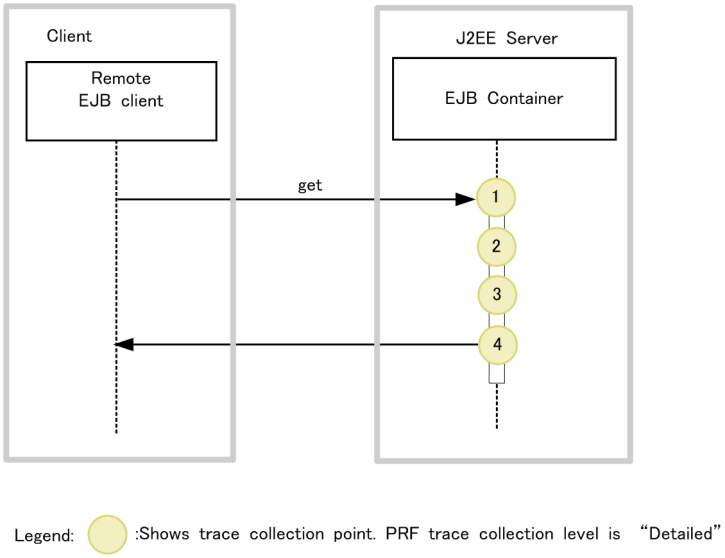
B: Advanced

#

Corresponds to the numbers in Figure 8-30.

The following figure shows the trace collection points when `get` is invoked from the remote client.

Figure 8–30: Trace collection points when get is invoked from the remote client



(b) Trace information that can be collected

The following table describes the trace information that can be collected when `get` is invoked from the remote client.

Table 8–47: Trace information that can be collected when get is invoked from the remote client

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x84C6	B	--	--	--
2	0x84C2	B	--	--	--
3	0x84C3	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*
4	0x84C7	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*
1	0x84C8	B	--	--	Method name, argument information
2	0x84C4	B	--	--	Method name, argument information
3	0x84C5	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-timeexception-name*
4	0x84C9	B	--	--	• 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-timeexception-name

Legend:

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-30.

(5) When isDone is invoked from the local client

(a) Trace collection points and PRF trace

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–48: Details of the trace collection points when isDone is invoked from the local client

Event ID	No. in the figure#	Trace collection points	Level
0x84CA	1	When the processing of `Future.isDone()` starts (for a local invocation)	B
0x84CB	2	When the processing of `Future.isDone()` ends (for a local invocation)	B

Legend:

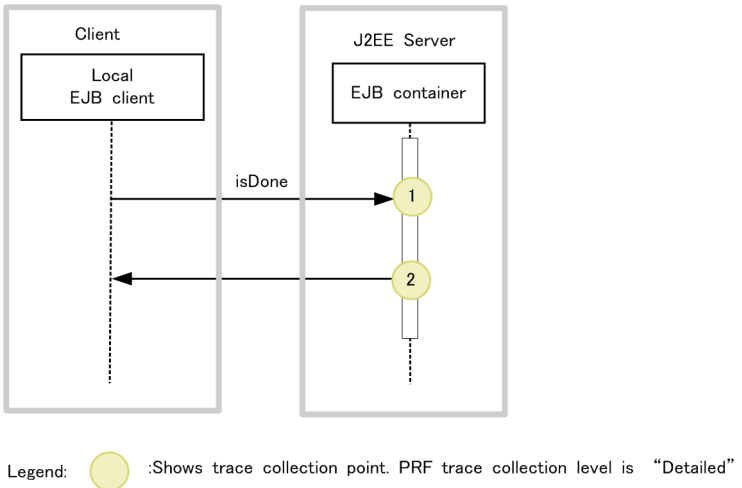
B: Advanced

#

Corresponds to the numbers in Figure 8-31.

The following figure shows the trace collection points when `isDone` is invoked from the local client.

Figure 8–31: Trace collection points when isDone is invoked from the local client



(b) Trace information that can be collected

The following table describes the trace information that can be collected when `isDone` is invoked from the local client.

Table 8–49: Trace information that can be collected when isDone is invoked from the local client

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Option
1	0x84CA	B	--	--	--
2	0x84CB	B	--	--	Return value of `isDone()`

Legend:

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-31.

Note

No exceptions are thrown in the execution of this method. Therefore, the logs showing abnormal status are not generated.

(6) When isDone is invoked from the remote client

(a) Trace collection points and PRF trace

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–50: Details of the trace collection points when isDone is invoked from the remote client

Event ID	No. in the figure#	Trace collection points	Level
0x84CE	1	When the processing of `Future.isDone()` starts (for a remote invocation)	B
0x84CA	2	When the processing of `Future.isDone()` starts (for a local invocation)	B
0x84CB	3	When the processing of `Future.isDone()` ends (for a local invocation)	B
0x84CF	4	When the processing of `Future.isDone()` ends (for a remote invocation)	B

Legend:

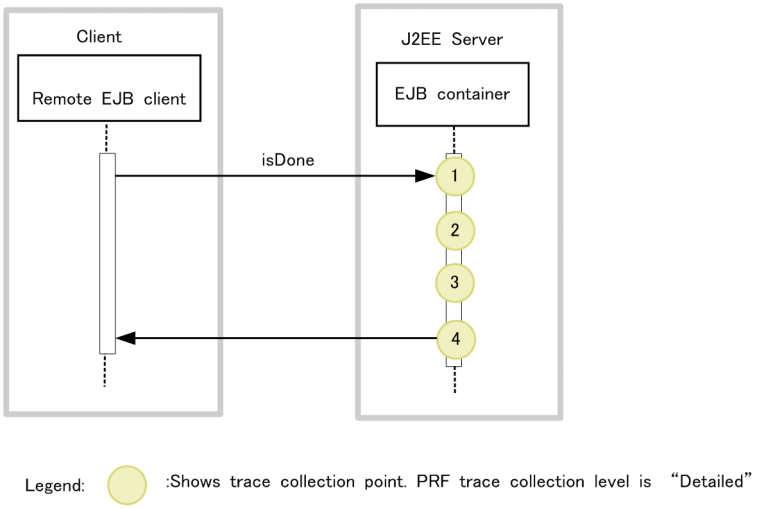
B: Advanced

#

Corresponds to the numbers in Figure 8-32.

The following figure shows the trace collection points when `isDone` is invoked from the remote client.

Figure 8–32: Trace collection points when isDone is invoked from the remote client



(b) Trace information that can be collected

The following table describes the trace information that can be collected when `isDone` is invoked from the remote client.

Table 8–51: Trace information that can be collected when isDone is invoked from the remote client

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Option
1	0x84CE	B	--	--	--
2	0x84CA	B	--	--	--
3	0x84CB	B	--	--	Return value of `isDone()`
4	0x84CF	B	--	--	• When normal: *Entrance-time* • For an exception: *exception-name*

Legend:

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-32.

(7) When isCancelled is invoked from the local client

(a) Trace collection points and PRF trace

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–52: Details of the trace collection points when isCancelled is invoked from the local client

Event ID	No. in the figure#	Trace collection points	Level
0x84CC	1	When the processing of `Future.isCancelled()` starts (for a local invocation)	B

Event ID	No. in the figure#	Trace collection points	Level
0x84CD	2	When the processing of `Future.isCancelled()` ends (for a local invocation)	B

Legend:

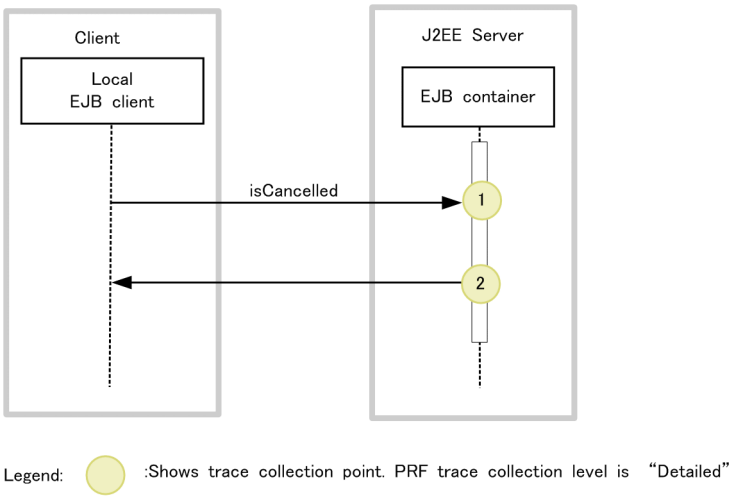
B: Advanced

#

Corresponds to the numbers in Figure 8-33.

The following figure shows the trace collection points when `isCancelled` is invoked from the local client.

Figure 8–33: Trace collection points when isCancelled is invoked from the local client



(b) Trace information that can be collected

The following table describes the trace information that can be collected when `isCancelled` is invoked from the local client.

Table 8–53: Trace information that can be collected when isCancelled is invoked from the local client

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Option
1	0x84CC	B	--	--	--
2	0x84CD	B	--	--	Return value of `isCancelled()`

Legend:

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-33.

Note

No exceptions are thrown in the execution of this method. Therefore, the logs showing abnormal status are not generated.

(8) When isCancelled is invoked from the remote client

(a) Trace collection points and PRF trace

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–54: Details of the trace collection points when isCancelled is invoked from the remote client

Event ID	No. in the figure#	Trace collection points	Level
0x84D0	1	When the processing of `Future.isCancelled()` starts (for a remote invocation)	B
0x84CC	2	When the processing of `Future.isCancelled()` starts (for a local invocation)	B
0x84CD	3	When the processing of `Future.isCancelled()` ends (for a local invocation)	B
0x84D1	4	When the processing of `Future.isCancelled()` ends (for a remote invocation)	B

Legend:

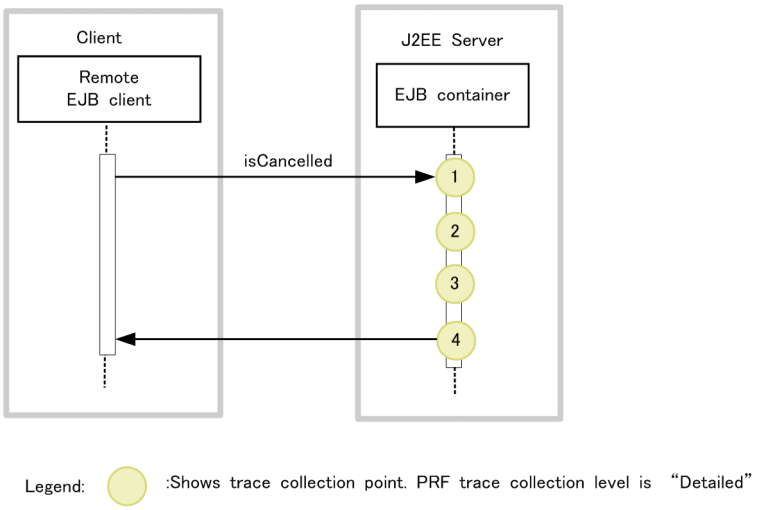
B: Advanced

#

Corresponds to the numbers in Figure 8-34.

The following figure shows the trace collection points when `isCancelled` is invoked from the remote client.

Figure 8–34: Trace collection points when isCancelled is invoked from the remote client



(b) Trace information that can be collected

The following table describes the trace information that can be collected when `isCancelled` is invoked from the remote client.

Table 8–55: Trace information that can be collected when isCancelled is invoked from the remote client

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Option
1	0x84D0	B	--	--	--
2	0x84CC	B	--	--	--
3	0x84CD	B	--	--	Return value of `isCancelled()`
4	0x84D1	B	--	--	• When normal: Return value of `isCancelled()` • For an exception: *exception-name*

Legend:

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-34.

(9) When cancel is invoked from the local client

(a) Trace collection points and PRF trace

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–56: Details of the trace collection points when cancel is invoked from the local client

Event ID	No. in the figure#	Trace collection points	Level
0x84D2	1	When the processing of `Future.cancel()` starts (for a local invocation)	A
0x84D3	2	When the processing of `Future.cancel()` ends (for a local invocation)	A

Legend:

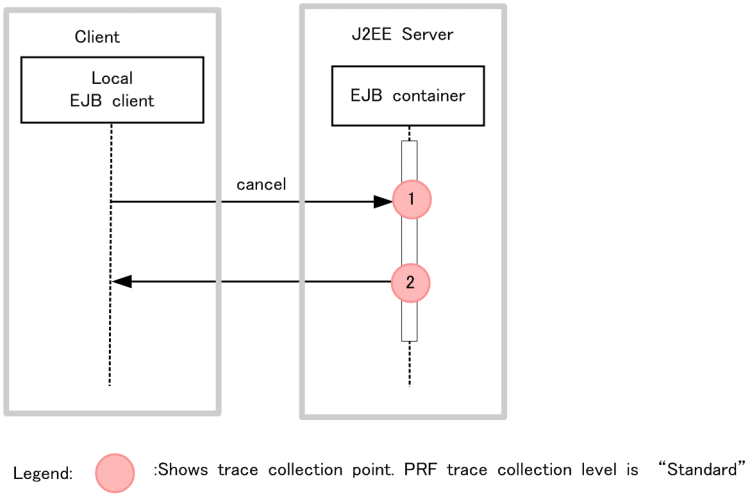
A: Standard

#

Corresponds to the numbers in Figure 8-35.

The following figure shows the trace collection points when `cancel` is invoked from the local client.

Figure 8–35: Trace collection points when cancel is invoked from the local client



(b) Trace information that can be collected

The following table describes the trace information that can be collected when `cancel` is invoked from the local client.

Table 8–57: Trace information that can be collected when cancel is invoked from the local client

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Option
1	0x84D2	A	--	--	Method name, argument information
2	0x84D3	A	--	--	Return value of `cancel()`

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-35.

Note

No exceptions are thrown in the execution of this method. Therefore, the logs showing abnormal status are not generated.

(10) When cancel is invoked from the remote client

(a) Trace collection points and PRF trace

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–58: Details of the trace collection points when cancel is invoked from the remote client

Event ID	No. in the figure#	Trace collection points	Level
0x84D4	1	When the processing of `Future.cancel()` starts (for a remote invocation)	A
0x84D2	2	When the processing of `Future.cancel()` starts (for a local invocation)	A
0x84D3	3	When the processing of `Future.cancel()` ends (for a local invocation)	A

Event ID	No. in the figure#	Trace collection points	Level
0x84D5	4	When the processing of `Future.cancel()` ends (for a remote invocation)	A

Legend:

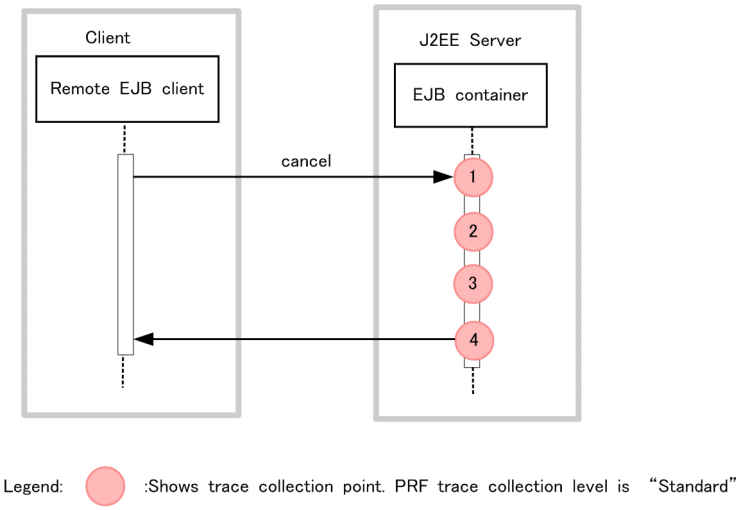
A: Standard

#

Corresponds to the numbers in Figure 8-36.

The following figure shows the trace collection points when `cancel` is invoked from the remote client.

Figure 8–36: Trace collection points when cancel is invoked from the remote client



(b) Trace information that can be collected

The following table describes the trace information that can be collected when `cancel` is invoked from the remote client.

Table 8–59: Trace information that can be collected when cancel is invoked from the remote client

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Option
1	0x84D4	A	--	--	Method name, argument information
2	0x84D2	A	--	--	Method name, argument information
3	0x84D3	A	--	--	Return value of `cancel()`
4	0x84D5	A	--	--	• When normal: Return value of `cancel()` • For an exception: *exception-name*

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-36.

8.8.6 When method cancellation occurs

This subsection describes the trace collection points when method cancellation occurs, and also describes the trace information that can be collected.

(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 8–60: Details of the trace collection points when method cancellation occurs

Event ID	No. in the figure#	Trace collection points	Level
0x8490	1	When the method cancellation processing starts	A
0x8C41	2	When an SQL statement is output to check errors	A
0x8491	3	When the method cancellation processing is completed	A

Legend:

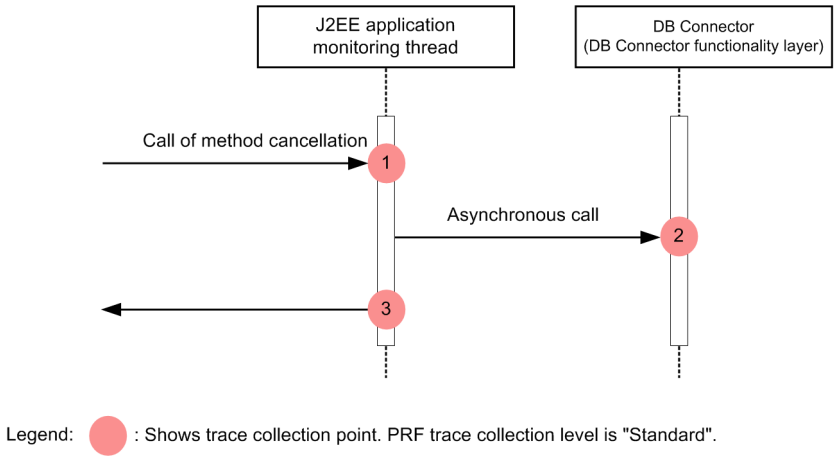
A: Standard

#

Corresponds to the numbers in Figure 8-37.

The following figure shows the trace collection points when method cancellation occurs.

Figure 8–37: Trace collection points when method cancellation occurs



(2) Trace information that can be collected

The following table describes the trace information that can be collected when method cancellation occurs.

Table 8–61: Trace information that can be collected when method cancellation occurs

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8490	A	Root application information of the application for which method cancellation is to be executed	--	--
2	0x8C41	A	Root application information of the connection for which a transaction timeout,	--	SQL statement

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
			forced termination of the J2EE application, or method cancellation was executed		
3	0x8491	A	--	--	*Entrance-time*

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-37.

8.9 Trace collection points of a JNDI

This section describes the trace collection points of a JNDI, and the trace information that can be collected.

8.9.1 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 8–62: Details of trace collection points in a JNDI

Event ID	No. in the figure#	Trace acquisition points			Level
0x8603	1	When searching the name space in JNDI, or the CORBA Naming Service	`javax.naming.Context.lookup`	Immediately after invocation	A
0x8604	2			Immediately before return	A
0x8605	1		`javax.naming.Context.list`	Immediately after invocation	B
0x8606	2			Immediately before return	B
0x8607	1		`javax.naming.Context.listBindings`	Immediately after invocation	B
0x8608	2			Immediately before return	B
0x8609	1	When searching the name space in which `javax.ejb.EJBLocalHome` is saved	`javax.naming.Context.lookup`	Immediately after invocation	A
0x860A	2			Immediately before return	A
0x860B	1		`javax.naming.Context.list`	Immediately after invocation	B
0x860C	2			Immediately before return	B
0x860D	1		`javax.naming.Context.listBindings`	Immediately after invocation	B
0x860E	2			Immediately before return	B
0x860F	1	When searching the Java name space: `java`	`javax.naming.Context.lookup`	Immediately after invocation	A
0x8610	2			Immediately before return	A
0x8611	1		`javax.naming.Context.list`	Immediately after invocation	B
0x8612	2			Immediately before return	B
0x8613	1		`javax.naming.Context.listBindings`	Immediately after invocation	B

Event ID	No. in the figure#	Trace acquisition points			Level
0x8614	2			Immediately before return	B
0x8615	1	When using the round-robin search functionality	`javax.naming.Context.lookup`	Immediately after invocation	A
0x8616	2			Immediately before return	A
0x8617	1	When using the CORBA Naming Service switch functionality	`javax.naming.Context.lookup`	Immediately after invocation	A
0x8618	2			Immediately before return	A
0x8619	1		`javax.naming.Context.list`	Immediately after invocation	B
0x861A	2			Immediately before return	B
0x861B	1		`javax.naming.Context.listBindings`	Immediately after invocation	B
0x861C	2			Immediately before return	B

Legend:

A: Standard

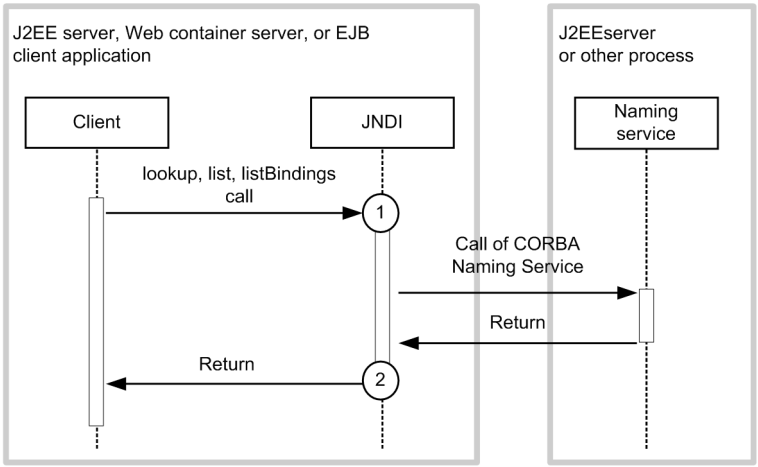
B: Advanced

#

Corresponds to the numbers in Figure 8-38.

The following figure shows the trace collection points in a JNDI.

Figure 8–38: Trace collection points of a JNDI



Legend: ◯ : Shows performance analysis trace collection point.
For the javax.naming.Context.lookup method, the collection level is "Standard".
For the javax.naming.Context.list method or the javax.naming.Context.list.Bindings method, the collection level is "Detailed".

8.9.2 Trace information that can be collected

The following table describes the trace information that can be collected in a JNDI.

Table 8–63: Trace information that can be collected in a JNDI

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8603	A	--	Specified name	--
	0x8609	A			
	0x860F	A			
	0x8615	A			
	0x8617	A			
	0x8605	B			
	0x860B	B			
	0x8611	B			
	0x8619	B			
	0x8607	B			
	0x860D	B			
	0x8613	B			
	0x861B	B			
2	0x8604	A	--		• When normal: *Entrance-time* • For an exception: *Entrance-time: exception name*
	0x860A	A			
	0x8610	A			
	0x8616	A			
	0x8618	A			
	0x8606	B			
	0x860C	B			
	0x8612	B			
	0x861A	B			
	0x8608	B			
	0x860E	B			
	0x8614	B			
	0x861C	B			

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-38.

Reference note

- In the trace information of a JNDI, `0` is displayed in the *root application information* and *client application information* that constitutes the key information, in the following cases:
  - When the `lookup` method is invoked from the client
  - When the server is either starting or stopping
- 0x8609 and 0x860A are output twice when you are using business interface.

8.10 Trace collection points of a JTA

This section describes the trace collection points of the JTA and the trace information that can be collected.

8.10.1 When a CMT and TransactionManager are used

This subsection describes the trace collection points when a CMT and `javax.transaction.TransactionManager` are used, and also the trace information that can be collected.

(1) Trace collection points and the PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–64: Details of trace collection points when a CMT and TransactionManager are used

Event ID	No. in the figure#	Trace acquisition points	Level
0x8811	1	Immediately before the transaction starts	A
0x8812	2	Immediately after the transaction starts	A
0x8815	3	Immediately before the transaction is committed	A
0x8816	4	Immediately after the transaction is committed	A
0x8817	3	Immediately before the transaction rolls back	A
0x8818	4	Immediately after the transaction rolls back	A

Legend:

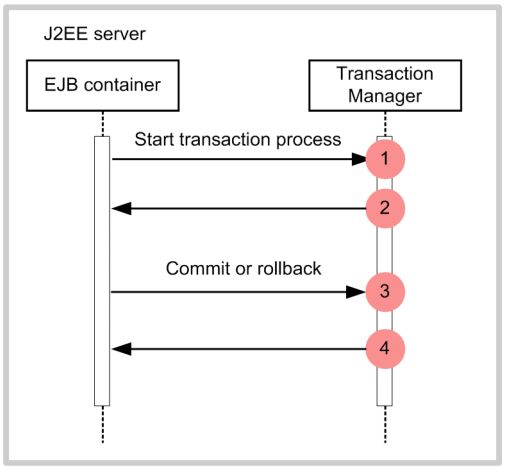
A: Standard

#

Corresponds to the numbers in Figure 8-39 and Figure 8-40.

The following figure shows the trace collection points when a CMT is used.

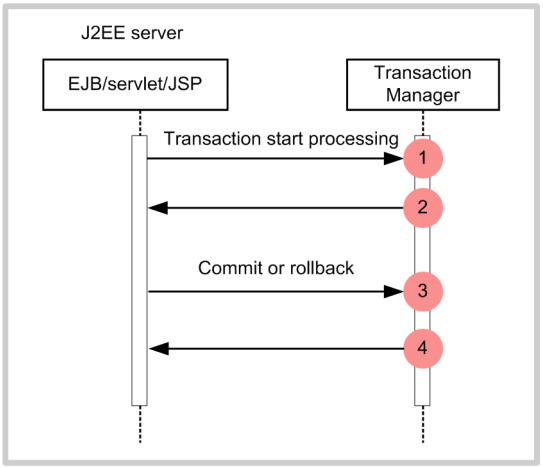
Figure 8–39: Trace collection points when a CMT is used



Legend: : Shows trace collection point. PRF trace collection level is "Standard".

The following figure shows the trace collection points when TransactionManager is used.

Figure 8–40: Trace collection points when TransactionManager is used



Legend: : Shows trace collection point. PRF trace collection level is "Standard".

(2) Trace information that can be collected

The following table describes the trace information that can be collected when a CMT and TransactionManager are used.

Table 8–65: Trace information that can be collected when a CMT and TransactionManager are used

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8811	A	--	--	--
2	0x8812	A	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
3	0x8815	A	--	--	--
	0x8817	A	--	--	--
4	0x8816	A	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
	0x8818	A	--	--	

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-39 and Figure 8-40.

8.10.2 When UserTransaction is used

This subsection describes the trace collection points when UserTransaction is used, and also the trace information that can be collected.

(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 8–66: Details of trace collection points when UserTransaction is used

Event ID	No. in the figure#	Trace acquisition points	Level
0x8813	1	Immediately before the transaction starts	A
0x8814	2	Immediately after the transaction starts	A
0x8815	3	Immediately before the transaction is committed	A
0x8816	4	Immediately after the transaction is committed	A
0x8817	3	Immediately before the transaction rolls back	A
0x8818	4	Immediately after the transaction rolls back	A

Legend:

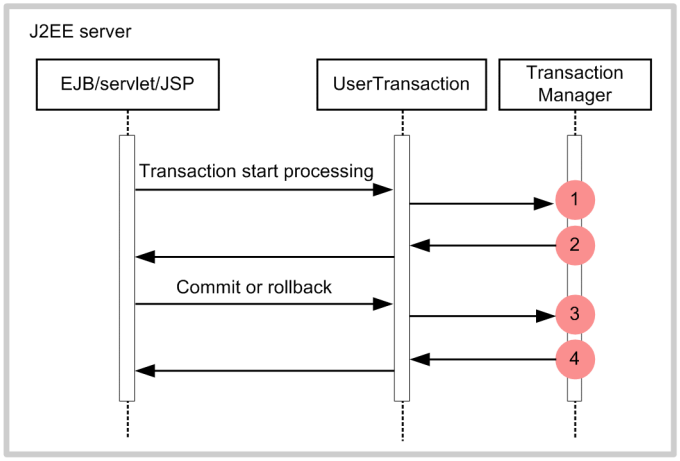
A: Standard

#

Corresponds to the numbers in Figure 8-41.

The following figure shows the trace collection points when UserTransaction is used.

Figure 8–41: Trace collection points when UserTransaction is used



Legend: : Shows trace collection point. PRF trace collection level is "Standard".

(2) Trace information that can be collected

The following table describes the trace information that can be collected when UserTransaction is used.

Table 8–67: Trace information that can be collected when UserTransaction is used

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8813	A	--	--	--
2	0x8814	A	--	--	• 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
3	0x8815	A	--	--	--
	0x8817	A	--	--	--
4	0x8816	A	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
	0x8818	A	--	--	

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-41.

8.10.3 In the case of a transaction timeout

This subsection describes the trace collection points in the case of a transaction timeout, and also describes the trace information that can be collected.

(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 8–68: Details of trace collection points in the case of a transaction timeout

Event ID	No. in the figure#	Trace acquisition points	Level
0x8819	1	Immediately before the transaction timeout processing	A
0x8C41	2	SQL output for failure checking	A
0x8820	3	Immediately after the transaction timeout processing	A

Legend:

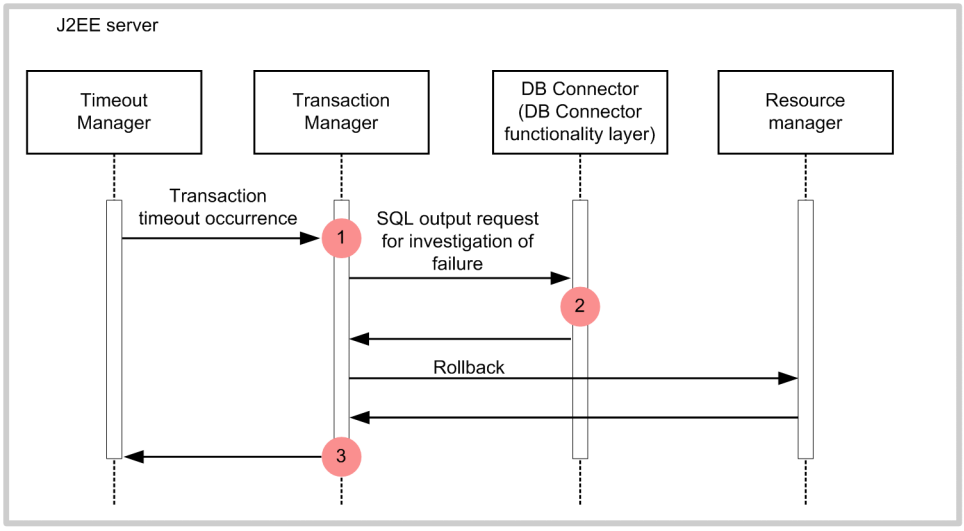
A: Standard

#

Corresponds to the numbers in Figure 8-42.

The following figure shows the trace collection points in the case of a transaction timeout.

Figure 8–42: Trace collection points in the case of a transaction timeout



Legend: : Shows trace collection point. 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 a transaction timeout.

Table 8–69: Trace information that can be collected in the case of a transaction timeout

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8819	A	Root application information of the transaction that has timed out	--	--
2	0x8C41	A	Root application information of the connection for which a transaction timeout, forced termination of the J2EE application, or method cancellation was executed	--	SQL statement
3	0x8820	A	--	--	*Entrance-time*

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-42.

8.10.4 When using the asynchronous concurrent processing for threads

This subsection describes the trace collection points and the trace information that can be acquired, when you use the asynchronous concurrent processing for threads.

(1) Trace collection points of TimerManager

- **Trace collection points and PRF trace collection levels**

The following table describes the event IDs, trace collection points, and PRF trace collection levels:

Table 8–70: Details of the trace collection points of TimerManager

Event ID	Number in the figure#	Trace collection point	Level
0x8435	1	Immediately before TimerManager starts	A
0x8436	2	Just after TimerManager starts	A
0x8439	5	Immediately before TimerManager stops	A
0x843A	6	Just after TimerManager stops	A
0x843D	3	Immediately before executing the listener for TimerManager	A
0x843E	4	Just after executing the listener for TimerManager	A

Legend:

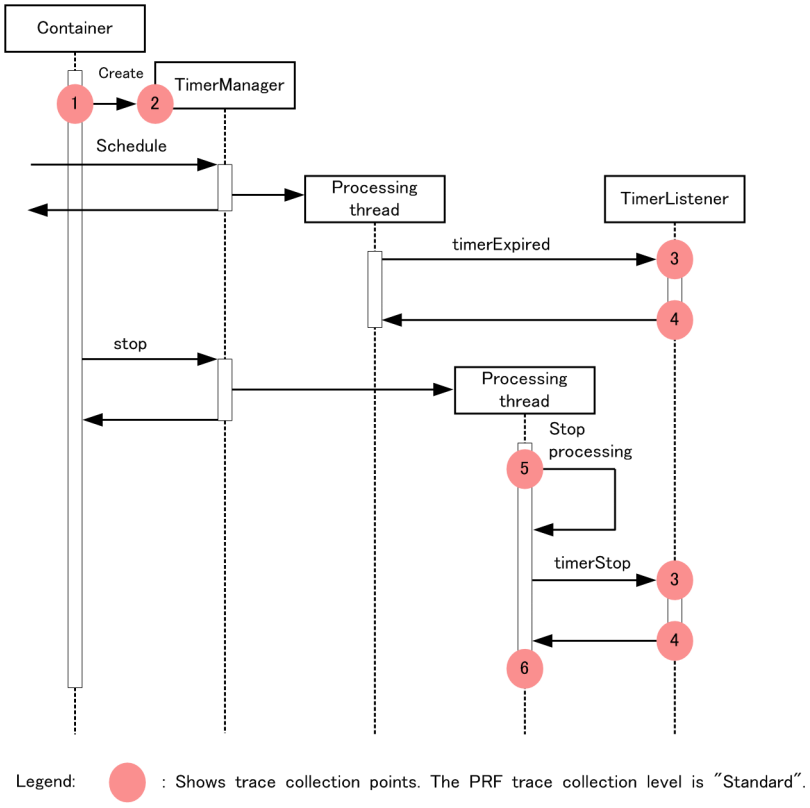
A: Standard

#

Corresponds to the numbers in Figure 8-43.

The following figure shows the trace collection points of TimerManager.

Figure 8–43: Trace collection points of TimerManager



- **Trace information that can be collected**

  The following table describes the trace information that can be collected for TimerManager.

Table 8–71: Trace information that can be collected for TimerManager

Number in the figure[#1]	Event ID	Level	Information that can be collected		
			Interface name	Operation name	Option
1	0x8435	A	--	--	--
2	0x8436	A	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
3	0x843D	A	Method name[#2]	Unique number for each schedule	--
4	0x843E	A	Method name[#2]	Unique number for each schedule	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
5	0x8439	A	--	--	--
6	0x843A	A	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*

Legend:

A: Standard

--: Not applicable

#1

Corresponds to the numbers in Figure 8-43.

#2

`TimerManager.timerExpired`, `StopTimerListener.timerStop`, or `CancelTimerListener.timerCancel` is output.

(2) Trace collection points of WorkManager

- **Trace collection points and PRF trace collection levels**

  The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–72: Details of the trace collection points of WorkManager

Event ID	Number in the figure[#]	Trace collection point	Level
0x8437	1	Immediately before WorkManager starts	A
0x8438	2	Just after WorkManager starts	A
0x8440	4	Just after executing the listener or task processing for WorkManager	A

Event ID	Number in the figure#	Trace collection point	Level
0x843B	5	Immediately before WorkManager stops	A
0x843C	6	Just after WorkManager stops	A
0x843F	3	Immediately before executing the listener or task processing for WorkManager	A

Legend:

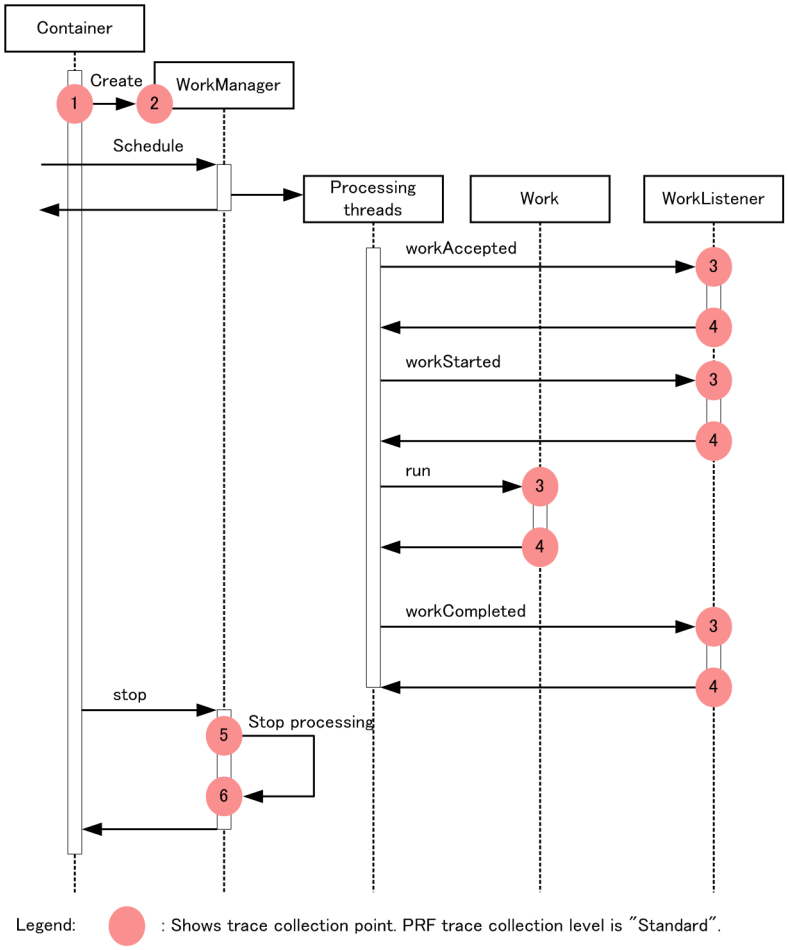
A: Standard

#

Corresponds to the numbers in Figure 8-44.

The following figure shows the trace collection points of WorkManager.

Figure 8–44: Trace collection points of WorkManager



- **Trace information that can be collected**

  The following table describes the trace information that can be collected for WorkManager.

  Table 8–73: Trace information that can be collected for WorkManager

Number in the figure# 1	Event ID	Level	Information that can be collected		
			Interface name	Operation name	Option
1	0x8437	A	--	--	--

Number in the figure[#1]	Event ID	Level	Information that can be collected		
			Interface name	Operation name	Option
2	0x8438	A	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
3	0x843F	A	Method name[#2]	Unique number for each schedule	--
4	0x8440	A	Method name[#2]	Unique number for each schedule	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
5	0x843B	A	--	--	--
6	0x843C	A	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*

Legend:

A: Standard

--: Not applicable

#1

Corresponds to the numbers in Figure 8-44.

#2

`Work.run`, `WorkListener.workAccepted`, `WorkListener.workRejected`, `WorkListener.workStarted`, or `WorkListener.workCompleted` is output.

8.11 Trace collection points of a DB Connector and JCA container

This section describes the trace collection points of a DB Connector and JCA container, and also the trace information that can be collected.

Of the trace collection points related to a connection, some can be collected only when a local transaction is used. This section separately describes the trace collection points that can be collected irrespective of the transaction support level, and the trace collection points that can be collected only when a local transaction is used. Here, the trace collection points that can be collected irrespective of the transaction support level are called *connection-related trace collection points*.

8.11.1 Connection-related trace collection points and trace information that can be collected

This subsection describes the connection-related trace collection points and trace information that can be collected.

(1) Trace get point and the PRF trace get level

The following four tables describe the event IDs, trace collection points, and PRF trace collection levels with reference to each event ID:

- In the case of 0x8B00, 0x8B01, 0x8B80-0x8B83, 0x8C00-0x8C03, 0x8C10-0xC13, 0x8C20-0x8C29, 0x8C2A-0x8C3F (when connection-related processing is executed)
  Reference: *Table 8-74*
- In the case of 0x8C80-0x8C93 (when a method of the java.sql.Statement interface is executed)
  Reference: *Table 8-75*
- In the case of 0x8CC0-0x8CD9 (when a method of the java.sql.PreparedStatement interface is executed)
  Reference: *Table 8-76*
- In the case of 0x8D00-0x8D19 (when a method of the java.sql.CallableStatement interface is executed)
  Reference: *Table 8-77*

Tip

For details about the event ID 0x8C41, see the sections *8.8 Trace collection points of an EJB container* and *8.10 Trace collection points of a JTA*.

Table 8–74: Details of trace collection points in a DB Connector and JCA container (when connection-related processing is executed) 1

Event ID	No. in the figure#	Trace acquisition points	Level
0x8B00	2	Immediately after invoking the connection acquisition request from the resource adapter	B
0x8B01	5	Immediately before the return of the connection acquisition request from the resource adapter	B
0x8B80	3	Immediately before the invocation of physical connection creation	B
0x8B81	4	Immediately after the return of physical connection creation	B
0x8B82	14	Immediately before the invocation of physical connection discard	B

Event ID	No. in the figure#	Trace acquisition points		Level
0x8B83	15	Immediately after the return of physical connection discard		B
0x8C00	1	When a database connection is established with `javax.sql.DataSource.getConnection()`	When the processing starts	A
0x8C01	6		When the processing ends	A
0x8C02	1	When a database connection is established with `javax.sql.DataSource.getConnection(String username, String password)`	When the processing starts	A
0x8C03	6		When the processing ends	A
0x8C10	1	When a database connection is established with `javax.sql.DataSource.getConnection()` during the use of the connection pool clustering functionality (compatibility functionality)	When the processing starts	A
0x8C11	6		When the processing ends	A
0x8C12	1	When a database connection is established with `javax.sql.DataSource.getConnection(String username, String password)` during the use of the connection pool clustering functionality (compatibility functionality)	When the processing starts	A
0x8C13	6		When the processing ends	A
0x8C20	13	When the database and JDBC resources of the Connection object are released with `java.sql.Connection.close()`	When the processing starts	A
0x8C21	16		When the processing ends	A
0x8C22	11	`java.sql.Connection.commit()`	When the processing starts	B
0x8C23	12		When the processing ends	B
0x8C24	11	`java.sql.Connection.rollback()`	When the processing starts	B
0x8C25	12		When the processing ends	B
0x8C26	11	`java.sql.Connection.rollback(Savepoint savepoint)`	When the processing starts	B
0x8C27	12		When the processing ends	B
0x8C28	7	`java.sql.Connection.createStatement()`	When the processing starts	B
0x8C29	8		When the processing ends	B
0x8C2A	7	`java.sql.Connection.createStatement(int resultSetType, int resultSetConcurrency)`	When the processing starts	B
0x8C2B	8		When the processing ends	B
0x8C2C	7	`java.sql.Connection.createStatement(int resultSetType, int resultSetConcurrency, int resultSetHoldability)`	When the processing starts	B
0x8C2D	8		When the processing ends	B

Event ID	No. in the figure#	Trace acquisition points		Level
0x8C2E	7	`java.sql.Connection.prepareCall(String sql)`	When the processing starts	B
0x8C2F	8		When the processing ends	B
0x8C30	7	`java.sql.Connection.prepareCall(String sql, int resultSetType, int resultSetConcurrency)`	When the processing starts	B
0x8C31	8		When the processing ends	B
0x8C32	7	`java.sql.Connection.prepareCall(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)`	When the processing starts	B
0x8C33	8		When the processing ends	B
0x8C34	7	`java.sql.Connection.prepareStatement(String sql)`	When the processing starts	B
0x8C35	8		When the processing ends	B
0x8C36	7	`java.sql.Connection.prepareStatement(String sql, int autoGeneratedKeys)`	When the processing starts	B
0x8C37	8		When the processing ends	B
0x8C38	7	`java.sql.Connection.prepareStatement(String sql, int[] columnIndexes)`	When the processing starts	B
0x8C39	8		When the processing ends	B
0x8C3A	7	`java.sql.Connection.prepareStatement(String sql, int resultSetType, int resultSetConcurrency)`	When the processing starts	B
0x8C3B	8		When the processing ends	B
0x8C3C	7	`java.sql.Connection.prepareStatement(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)`	When the processing starts	B
0x8C3D	8		When the processing ends	B
0x8C3E	7	`java.sql.Connection.prepareStatement(String sql, String[] columnNames)`	When the processing starts	B
0x8C3F	8		When the processing ends	B

Legend:

A: Standard

B: Advanced

Note: Event ID of DB Connector is not output, if you are using SQL Server 2005.

#

Corresponds to the numbers in Figure 8-45.

Table 8–75: Details of trace collection points in a DB Connector and JCA container (when a method of the java.sql.Statement interface is executed) 2

Event ID	No. in the figure#	Trace acquisition points		Level
0x8C80	9	`execute(String sql)`	When the processing starts	B
0x8C81	10		When the processing ends	B
0x8C82	9	`execute(String sql, int autoGeneratedKeys)`	When the processing starts	B
0x8C83	10		When the processing ends	B
0x8C84	9	`execute(String sql, int[] columnIndexes)`	When the processing starts	B
0x8C85	10		When the processing ends	B
0x8C86	9	`execute(String sql, String[] columnNames)`	When the processing starts	B
0x8C87	10		When the processing ends	B
0x8C88	9	`executeBatch()`	When the processing starts	B
0x8C89	10		When the processing ends	B
0x8C8A	9	`executeQuery(String sql)`	When the processing starts	B
0x8C8B	10		When the processing ends	B
0x8C8C	9	`executeUpdate(String sql)`	When the processing starts	B
0x8C8D	10		When the processing ends	B
0x8C8E	9	`executeUpdate(String sql, int autoGeneratedKeys)`	When the processing starts	B
0x8C8F	10		When the processing ends	B
0x8C90	9	`executeUpdate(String sql, int[] columnIndexes)`	When the processing starts	B
0x8C91	10		When the processing ends	B
0x8C92	9	`executeUpdate(String sql, String[] columnNames)`	When the processing starts	B
0x8C93	10		When the processing ends	B

Legend:

B: Advanced

#

Corresponds to the numbers in Figure 8-45.

Table 8–76: Details of trace collection points in a DB Connector and JCA container (when a method of the java.sql.PreparedStatement interface is executed) 3

Event ID	No. in the figure#	Trace acquisition points		Level
0x8CC0	9	`execute()`	When the processing starts	B
0x8CC1	10		When the processing ends	B
0x8CC2	9	`execute(String sql)`	When the processing starts	B
0x8CC3	10		When the processing ends	B
0x8CC4	9	`execute(String sql, int autoGeneratedKeys)`	When the processing starts	B
0x8CC5	10		When the processing ends	B
0x8CC6	9	`execute(String sql, int[] columnIndexes)`	When the processing starts	B
0x8CC7	10		When the processing ends	B
0x8CC8	9	`execute(String sql, String[] columnNames)`	When the processing starts	B
0x8CC9	10		When the processing ends	B
0x8CCA	9	`executeBatch()`	When the processing starts	B
0x8CCB	10		When the processing ends	B
0x8CCC	9	`executeQuery()`	When the processing starts	B
0x8CCD	10		When the processing ends	B
0x8CCE	9	`executeQuery(String sql)`	When the processing starts	B
0x8CCF	10		When the processing ends	B
0x8CD0	9	`executeUpdate()`	When the processing starts	B
0x8CD1	10		When the processing ends	B
0x8CD2	9	`executeUpdate(String sql)`	When the processing starts	B
0x8CD3	10		When the processing ends	B

Event ID	No. in the figure#	Trace acquisition points		Level
0x8CD4	9	`executeUpdate(String sql, int autoGeneratedKeys)`	When the processing starts	B
0x8CD5	10		When the processing ends	B
0x8CD6	9	`executeUpdate(String sql, int[] columnIndexes)`	When the processing starts	B
0x8CD7	10		When the processing ends	B
0x8CD8	9	`executeUpdate(String sql, String[] columnNames)`	When the processing starts	B
0x8CD9	10		When the processing ends	B

Legend:

B: Advanced

#

Corresponds to the numbers in Figure 8-45.

Table 8–77: Details of trace collection points in a DB Connector and JCA container (when a method of the java.sql.CallableStatement interface is executed) 4

Event ID	No. in the figure#	Trace acquisition points		Level
0x8D00	9	`execute()`	When the processing starts	B
0x8D01	10		When the processing ends	B
0x8D02	9	`execute(String sql)`	When the processing starts	B
0x8D03	10		When the processing ends	B
0x8D04	9	`execute(String sql, int autoGeneratedKeys)`	When the processing starts	B
0x8D05	10		When the processing ends	B
0x8D06	9	`execute(String sql, int[] columnIndexes)`	When the processing starts	B
0x8D07	10		When the processing ends	B
0x8D08	9	`execute(String sql, String[] columnNames)`	When the processing starts	B
0x8D09	10		When the processing ends	B
0x8D0A	9	`executeBatch()`	When the processing starts	B
0x8D0B	10		When the processing ends	B

Event ID	No. in the figure#	Trace acquisition points		Level
0x8D0C	9	`executeQuery()`	When the processing starts	B
0x8D0D	10		When the processing ends	B
0x8D0E	9	`executeQuery(String sql)`	When the processing starts	B
0x8D0F	10		When the processing ends	B
0x8D10	9	`executeUpdate()`	When the processing starts	B
0x8D11	10		When the processing ends	B
0x8D12	9	`executeUpdate(String sql)`	When the processing starts	B
0x8D13	10		When the processing ends	B
0x8D14	9	`executeUpdate(String sql, int autoGeneratedKeys)`	When the processing starts	B
0x8D15	10		When the processing ends	B
0x8D16	9	`executeUpdate(String sql, int[] columnIndexes)`	When the processing starts	B
0x8D17	10		When the processing ends	B
0x8D18	9	`executeUpdate(String sql, String[] columnNames)`	When the processing starts	B
0x8D19	10		When the processing ends	B

Legend:

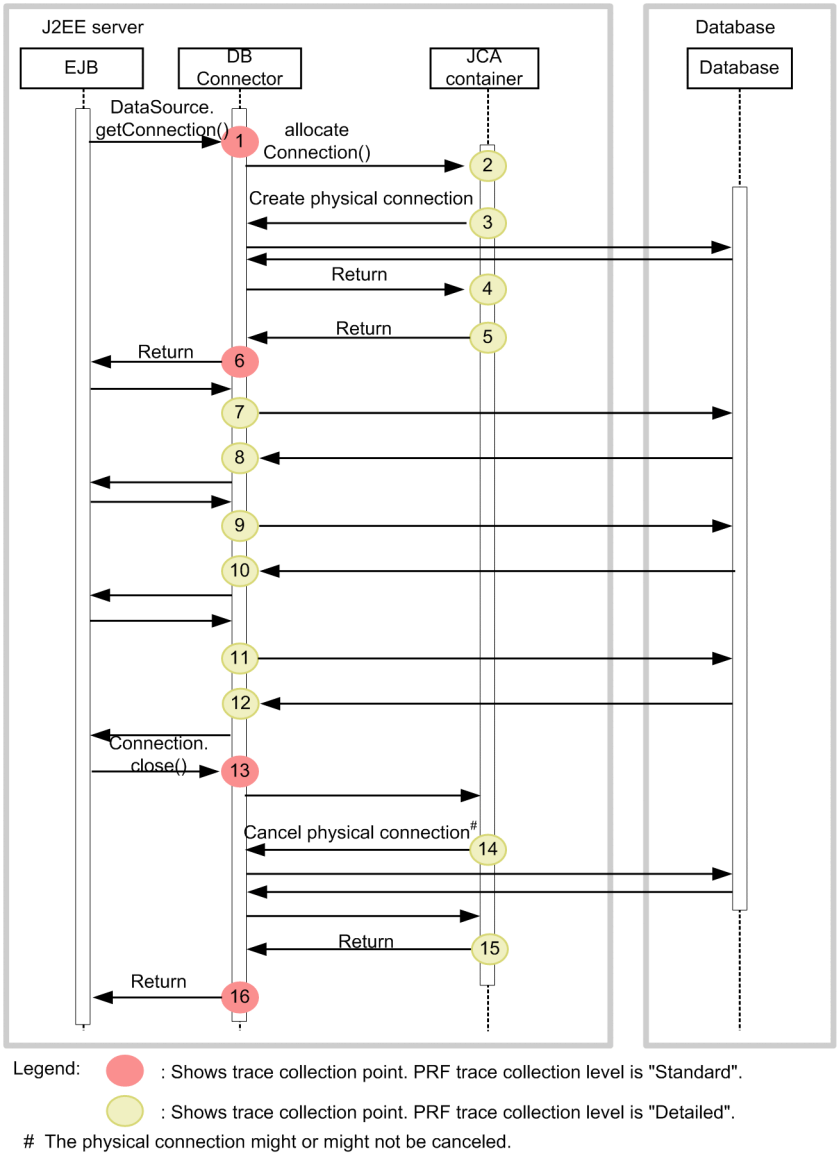
B: Advanced

#

Corresponds to the numbers in Figure 8-45.

The following figure shows the trace collection points in a DB Connector and JCA container.

Figure 8–45: Trace collection points in a DB Connector and JCA container (connection-related)



(2) Trace information that can be collected

The trace information that can be collected in a DB Connector or JCA container is described below. The details are explained in correlation to the numbers of Figure 8-45.

- Trace information corresponding to numbers 1-6 and 11-16

  The following table describes the event IDs and trace information corresponding to numbers 1-6 and 11-16.

  Table 8–78: Trace information that can be collected in a DB Connector and JCA container (when connection-related processing is executed)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8C00	A	--	--	--
	0x8C02	A	--	--	--
	0x8C10	A	--	--	--

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
	0x8C12	A	--	--	--
2	0x8B00	B	--	--	--
3	0x8B80	B	--	--	--
4	0x8B81	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
5	0x8B01	B	--	--	
6	0x8C01	A	Connection ID	--	
	0x8C03	A	Connection ID	--	
	0x8C11	A	Connection ID	--	
	0x8C13	A	Connection ID	--	
11	0x8C22	B	--	--	--
	0x8C24	B	--	--	--
	0x8C26	B	--	--	--
12	0x8C23	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
	0x8C25	B	--	--	
	0x8C27	B	--	--	
13	0x8C20	A	Connection ID	--	--
14	0x8B82	B	--	--	--
15	0x8B83	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
16	0x8C21	A	--	--	

Legend:
A: Standard
B: Advanced
--: Not applicable

#
Corresponds to the numbers in Figure 8-45.

- Trace information corresponding to number 7
  The event IDs corresponding to number 7 are as follows:
  0x8C28, 0x8C2A, 0x8C2C, 0x8C2E, 0x8C30, 0x8C32, 0x8C34,
  0x8C36, 0x8C38, 0x8C3A, 0x8C3C, 0x8C3E
  The trace information that can be collected at these event IDs is as follows:
  - PRF trace collection level
    All *Advanced*.
  - Interface name and operation name
    Not output at these event IDs.

- Optional
  If an sql exists in the method argument, the SQL statement is displayed:

- Trace information corresponding to number 8
  The event IDs corresponding to number 8 are as follows:
  0x8C29, 0x8C2B, 0x8C2D, 0x8C2F, 0x8C31, 0x8C33, 0x8C35,
  0x8C37, 0x8C39, 0x8C3B, 0x8C3D, 0x8C3F
  The trace information that can be collected at these event IDs is as follows:
  - PRF trace collection level
    All *Advanced*.
  - Interface name and operation name
    Not output at these event IDs.
  - Optional
    At these event IDs, the entrance time is displayed when the processing is performed normally. When an exception occurs, the entrance time and exception are displayed.
- Trace information corresponding to number 9
  The event IDs corresponding to number 9 are as follows:
  0x8C80, 0x8C82, 0x8C84, 0x8C86, 0x8C88, 0x8C8A, 0x8C8C, 0x8C8E, 0x8C90, 0x8C92, 0x8CC0, 0x8CC2, 0x8CC4, 0x8CC6, 0x8CC8, 0x8CCA, 0x8CCC, 0x8CCE, 0x8CD0, 0x8CD2, 0x8CD4, 0x8CD6, 0x8CD8, 0x8D00, 0x8D02, 0x8D04, 0x8D06, 0x8D08, 0x8D0A, 0x8D0C, 0x8D0E, 0x8D10, 0x8D12, 0x8D14, 0x8D16, 0x8D18
  The trace information that can be collected at these event IDs is as follows:
  - PRF trace collection level
    All *Advanced*.
  - Interface name and operation name
    Not output at these event IDs.
  - Optional
    If an sql exists in the method argument, the SQL statement is displayed:
- Trace information corresponding to number 10
  The event IDs corresponding to number 10 are as follows:
  0x8C81, 0x8C83, 0x8C85, 0x8C87, 0x8C89, 0x8C8B, 0x8C8D, 0x8C8F, 0x8C91, 0x8C93, 0x8CC1, 0x8CC3, 0x8CC5, 0x8CC7, 0x8CC9, 0x8CCB, 0x8CCD, 0x8CCF, 0x8CD1, 0x8CD3, 0x8CD5, 0x8CD7, 0x8CD9, 0x8D01, 0x8D03, 0x8D05, 0x8D07, 0x8D09, 0x8D0B, 0x8D0D, 0x8D0F, 0x8D11, 0x8D13, 0x8D15, 0x8D17, 0x8D19
  The trace information that can be collected at these event IDs is as follows:
  - PRF trace collection level
    All *Advanced*.
  - Interface name and operation name
    Not output at these event IDs.
  - Optional
    At these event IDs, the entrance time is displayed when the processing is performed normally. When an exception occurs, the entrance time and exception are displayed.

8.11.2 Trace collection points and trace information that can be collected when a local transaction is used

This subsection describes the trace collection points and the trace information that can be collected when a local transaction is used.

(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 8–79: Details of trace collection points in a DB Connector and JCA container (when the processing of a local transaction is executed)

Event ID	No. in the figure#	Trace acquisition points		Level
0x8C60	1	When a local transaction starts	When the processing starts	B
0x8C61	2		When the processing ends	B
0x8C62	3	When the local transaction is committed	When the processing starts	B
0x8C63	4		When the processing ends	B
0x8C64	3	When the local transaction rolls back	When the processing starts	B
0x8C65	4		When the processing ends	B

Legend:

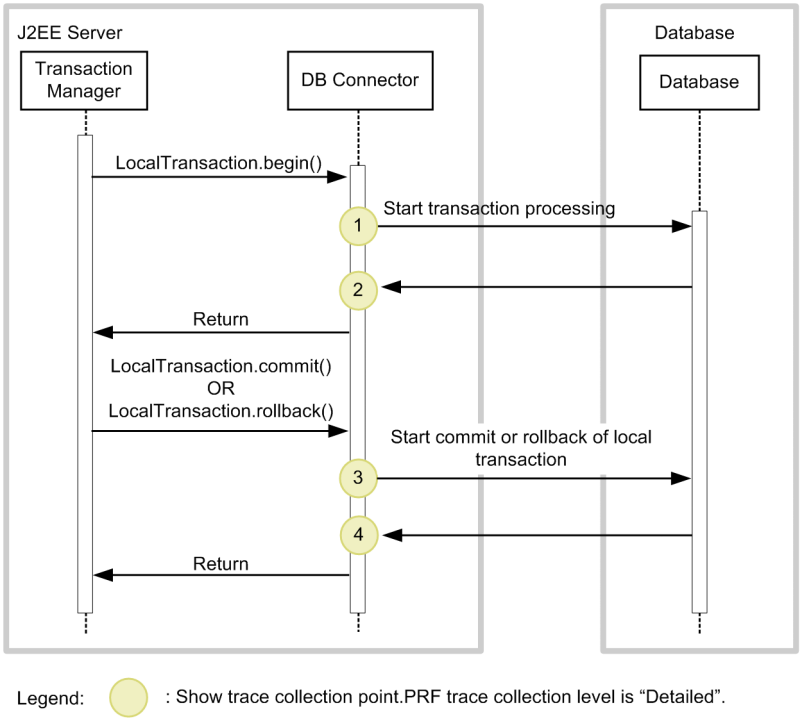
B: Advanced

#

Corresponds to the numbers in Figure 8-46.

The following figure shows the trace collection points.

Figure 8–46: Details of trace collection points in a DB Connector and JCA container (local transaction-related)



Legend: : Show trace collection point.PRF trace collection level is “Detailed”.

(2) Trace information that can be collected

The following table describes the trace information that can be collected.

Table 8–80: Details of trace collection points in a DB Connector and JCA container (when the processing of a local transaction is executed)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8C60	B	--	--	--
2	0x8C61	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
3	0x8C62	B	--	--	--
	0x8C64	B	--	--	--
4	0x8C63	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
	0x8C65	B	--	--	

Legend:

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-46.

8.11.3 Trace collection points and trace information that can be collected when a connection association is used

This subsection describes the trace collection points and the trace information that can be collected when a connection association is used.

(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 8–81: Details of trace collection points in a DB Connector and JCA container (when the processing of a connection association is executed)

Event ID	No. in the figure#	Trace acquisition points	Level
0x8C40	1	When a logical connection is replaced with a physical connection with the connection association functionality	A

Legend:

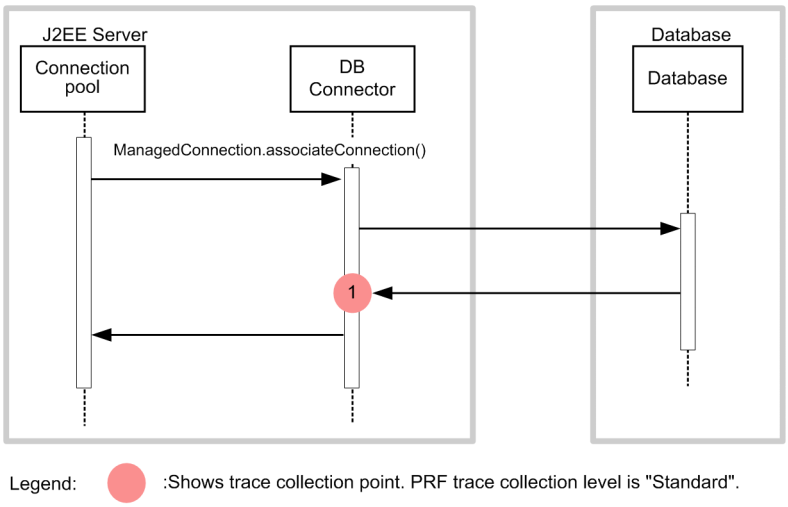
A: Standard

#

Corresponds to the numbers in Figure 8-47.

The following figure shows the trace collection points.

Figure 8–47: Details of trace collection points in a DB Connector and JCA container (connection association-related)



(2) Trace information that can be collected

The following table describes the trace information that can be collected.

Table 8–82: Details of trace collection points in a DB Connector and JCA container (when the connection association processing is executed)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8C40	A	Connection ID (connection ID of the physical connection that replaces another connection with the connection association functionality)	--	--

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-47.

8.11.4 Trace collection points and trace information that can be collected when the automatic connection close functionality is used

This subsection describes the trace collection points and the trace information that can be collected when the connection closes automatically.

(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 8–83: Details of trace collection points in a DB Connector and JCA container (when the connection closes automatically)

Event ID	No. in the figure#	Trace acquisition points	Level
0x8B84	1	Immediately after the invocation of the close processing by the automatic connection close functionality	B
0x8B82	2	Immediately before the invocation of physical connection discard	B
0x8B83	3	Immediately after the return of physical connection discard	B
0x8B85	4	Immediately before the return of the close processing by the automatic connection close functionality	B

Legend:

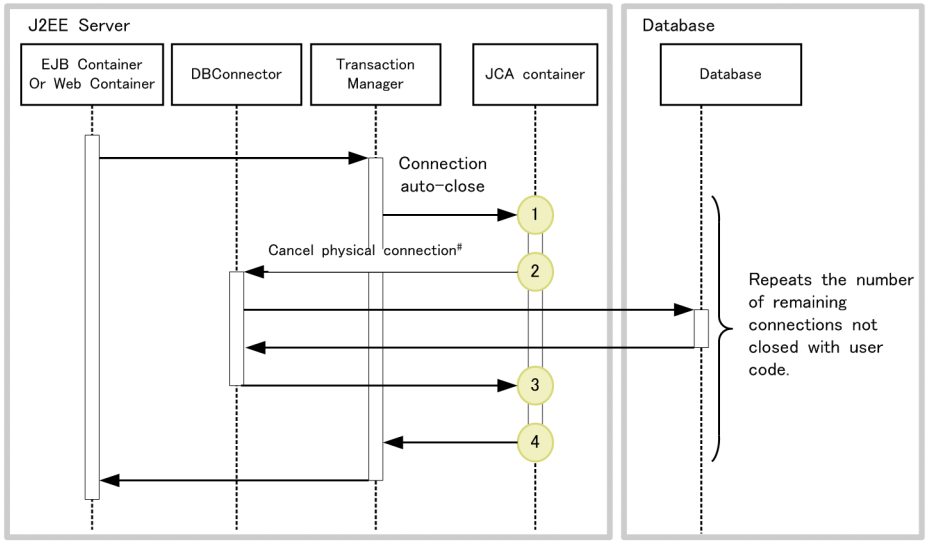
B: Advanced

#

Corresponds to the numbers in Figure 8-48.

The following figure shows the trace collection points.

Figure 8–48: Details of trace collection points in a DB Connector and JCA container (when the connection closes automatically)



Legend: :Shows trace collection point. PRF trace collection level is "Detailed".

The physical connection is not always lost every time the connection is closed.

(2) Trace information that can be collected

The following table describes the trace information that can be collected.

Table 8–84: Details of trace collection points in a DB Connector and JCA container (when the connection closes automatically)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8B84	B	Connection ID	--	--
2	0x8B82	B	--	--	--
3	0x8B83	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
4	0x8B85	B	--	--	*Entrance-time*

Legend:

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-48.

8.11.5 Trace collection points and trace information that can be collected in the case of linkage with the DB Connector for Cosminexus RM

This subsection describes the trace collection points and the trace information that can be collected in the case of linkage with the DB Connector for Cosminexus RM.

In the case of the DB Connector for Cosminexus RM, a connection is established to the database by using the DB Connector. Therefore, when the DB Connector for Cosminexus RM is used, the trace points of the DB Connector are also collected. The items generated during the JDBC connection (such as java.sql.Statement) have the same trace collection points as 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 8–85: Details of trace collection points in a DB Connector and JCA container (during linkage with the DB Connector for Cosminexus RM)

Event ID	No. in the figure#	Trace acquisition points		Level
0x8D60	1	When a database connection is established with `javax.sql.DataSource.getConnection()` during the use of the DB Connector for Cosminexus RM	When the processing starts	A
0x8D61	2		When the processing ends	A
0x8D62	1	When a database connection is established with `javax.sql.DataSource.getConnection(String username, String password)` during the use of the DB Connector for Cosminexus RM	When the processing starts	A
0x8D63	2		When the processing ends	A
0x8D80	5	When the database and JDBC resources of the Connection object are released with `java.sql.Connection.close()` during the use of the DB Connector for Cosminexus RM	When the processing starts	A
0x8D81	6		When the processing ends	A
0x8D82	3	`Connection.createStatement()` during the use of the DB Connector for Cosminexus RM	When the processing starts	B
0x8D83	4		When the processing ends	B
0x8D84	3	`Connection.createStatement(int resultSetType, int resultSetConcurrency)` during the use of the DB Connector for Cosminexus RM	When the processing starts	B
0x8D85	4		When the processing ends	B
0x8D86	3	`Connection.createStatement(int resultSetType, int resultSetConcurrency, int resultSetHoldability)` during the use of the DB Connector for Cosminexus RM	When the processing starts	B
0x8D87	4		When the processing ends	B
0x8D88	3	`Connection.prepareCall(String sql)` during the use of the DB Connector for Cosminexus RM	When the processing starts	B
0x8D89	4		When the processing ends	B

Event ID	No. in the figure#	Trace acquisition points		Level
0x8D8A	3	`(String sql, int resultSetType, int resultSetConcurrency)` during the use of the DB Connector for Cosminexus RM	When the processing starts	B
0x8D8B	4		When the processing ends	B
0x8D8C	3	`Connection.prepareCall(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)` during the use of the DB Connector for Cosminexus RM	When the processing starts	B
0x8D8D	4		When the processing ends	B
0x8D8E	3	`Connection.prepareStatement(String sql)` during the use of the DB Connector for Cosminexus RM	When the processing starts	B
0x8D8F	4		When the processing ends	B
0x8D90	3	`Connection.prepareStatement(String sql, int autoGeneratedKeys)` during the use of the DB Connector for Cosminexus RM	When the processing starts	B
0x8D91	4		When the processing ends	B
0x8D92	3	`Connection.prepareStatement(String sql, int[] columnIndexes)` during the use of the DB Connector for Cosminexus RM	When the processing starts	B
0x8D93	4		When the processing ends	B
0x8D94	3	`Connection.prepareStatement(String sql, int resultSetType, int resultSetConcurrency)` during the use of the DB Connector for Cosminexus RM	When the processing starts	B
0x8D95	4		When the processing ends	B
0x8D96	3	`Connection.prepareStatement(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)` during the use of the DB Connector for Cosminexus RM	When the processing starts	B
0x8D97	4		When the processing ends	B
0x8D98	3	`Connection.prepareStatement(String sql, String[] columnNames)` during the use of the DB Connector for Cosminexus RM	When the processing starts	B
0x8D99	4		When the processing ends	B

Legend:

A: Standard

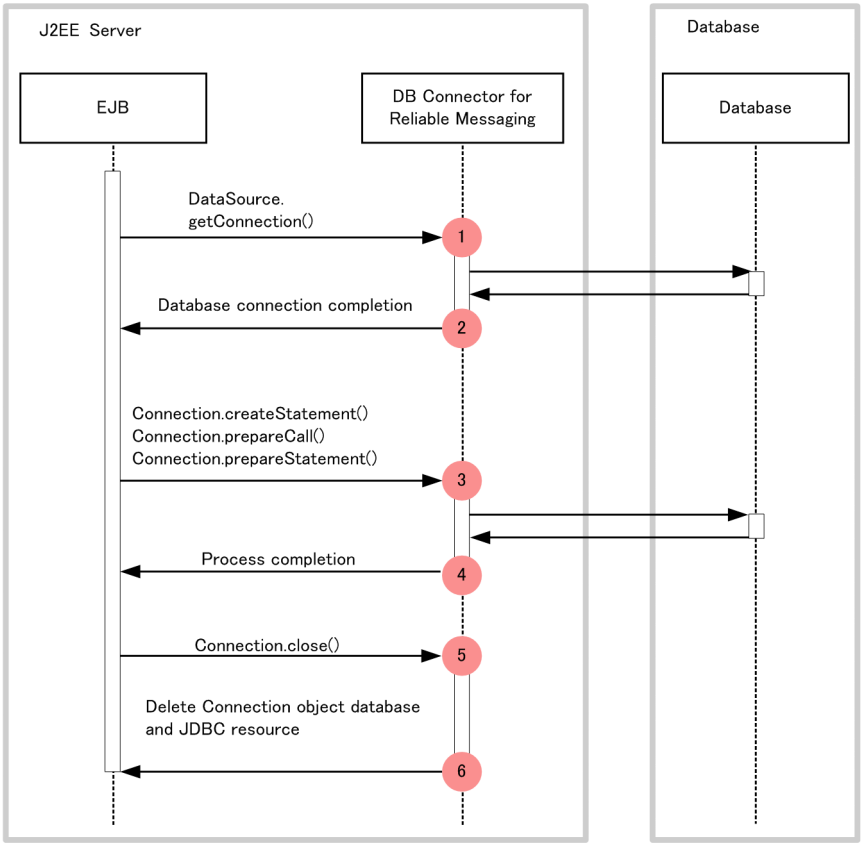
B: Advanced

#

Corresponds to the numbers in Figure 8-49.

The following figure shows the trace collection points.

Figure 8–49: Details of trace collection points in a DB Connector and JCA container (during linkage with the DB Connector for Cosminexus RM)



Legend: : Shows trace collection point. PRF trace collection level is "Standard".

(2) Trace information that can be collected

The following table describes the trace information that can be collected.

Table 8–86: Details of trace collection points in a DB Connector and JCA container (during linkage with the DB Connector for Cosminexus RM)

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8D60	A	--	--	--
2	0x8D61	A	Connection ID	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
1	0x8D62	A	--	--	--
2	0x8D63	A	Connection ID	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
5	0x8D80	A	Connection ID	--	--

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
6	0x8D81	A	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
3	0x8D82	B	--	--	--
4	0x8D83	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
3	0x8D84	B	--	--	--
4	0x8D85	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
3	0x8D86	B	--	--	--
4	0x8D87	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
3	0x8D88	B	--	--	SQL statement
4	0x8D89	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
3	0x8D8A	B	--	--	SQL statement
4	0x8D8B	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
3	0x8D8C	B	--	--	SQL statement
4	0x8D8D	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
3	0x8D8E	B	--	--	SQL statement
4	0x8D8F	B	--	--	• 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*
3	0x8D90	B	--	--	SQL statement
4	0x8D91	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time* *exception-name*
3	0x8D92	B	--	--	SQL statement
4	0x8D93	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time* *exception-name*
3	0x8D94	B	--	--	SQL statement
4	0x8D95	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time* *exception-name*
3	0x8D96	B	--	--	SQL statement
4	0x8D97	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time* *exception-name*
3	0x8D98	B	--	--	SQL statement
4	0x8D99	B	--	--	• When normal: *Entrance-time* • For an exception: *Entrance-time* *exception-name*

Legend:

A: Standard

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-49.

8.11.6 Trace collection points and trace information that can be collected when work management is used

This subsection describes the trace collection points and the trace information that can be collected when work management is used in a resource adapter conforming to Connector1.5.

Note that if you are using a resource adapter other than the product resource adapter, new numbers will be allotted to the root AP information when 0x8B86 is output. If the `<resourceadapter>-<resourceadapter-class>` value of DD (`ra.xml`) starts with "com.hitachi" or "com.cosminexus", the resource adapter will be considered as a product resource adapter.

The root AP information set at the entrance of MDB container is treated as the root AP information of Work, if you invoke Message-driven Bean from Work. Individual Message-driven Bean processing is distinguished by the thread ID.

(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 8–87: Details of trace collection points in a DB Connector and JCA container (when work management is used)

Event ID	No. in the figure#	Trace acquisition points	Level
0x8B86	1	Immediately after invoking a method of `javax.resource.spi.work.WorkManager`	A
0x8B87	6, 9, 14	Immediately before the return of the method of `javax.resource.spi.work.WorkManager`	A
0x8B88	2, 4, 7, 12	Immediately before invoking a method of `javax.resource.spi.work.WorkListener`	B
0x8B89	3, 5, 8, 13	Immediately after the return of the method of `javax.resource.spi.work.WorkListener`	B
0x8B8A	10	Immediately before invoking a method of `javax.resource.spi.work.Work`	A
0x8B8B	11	Immediately after the return of the method of `javax.resource.spi.work.Work`	A

Legend:

A: Standard

B: Advanced

#

Corresponds to the numbers in Figure 8-50 through Figure 8-52.

The following figure shows the trace collection points.

Figure 8–50: Details of trace collection points in a DB Connector and JCA container (when scheduleWork() is invoked)

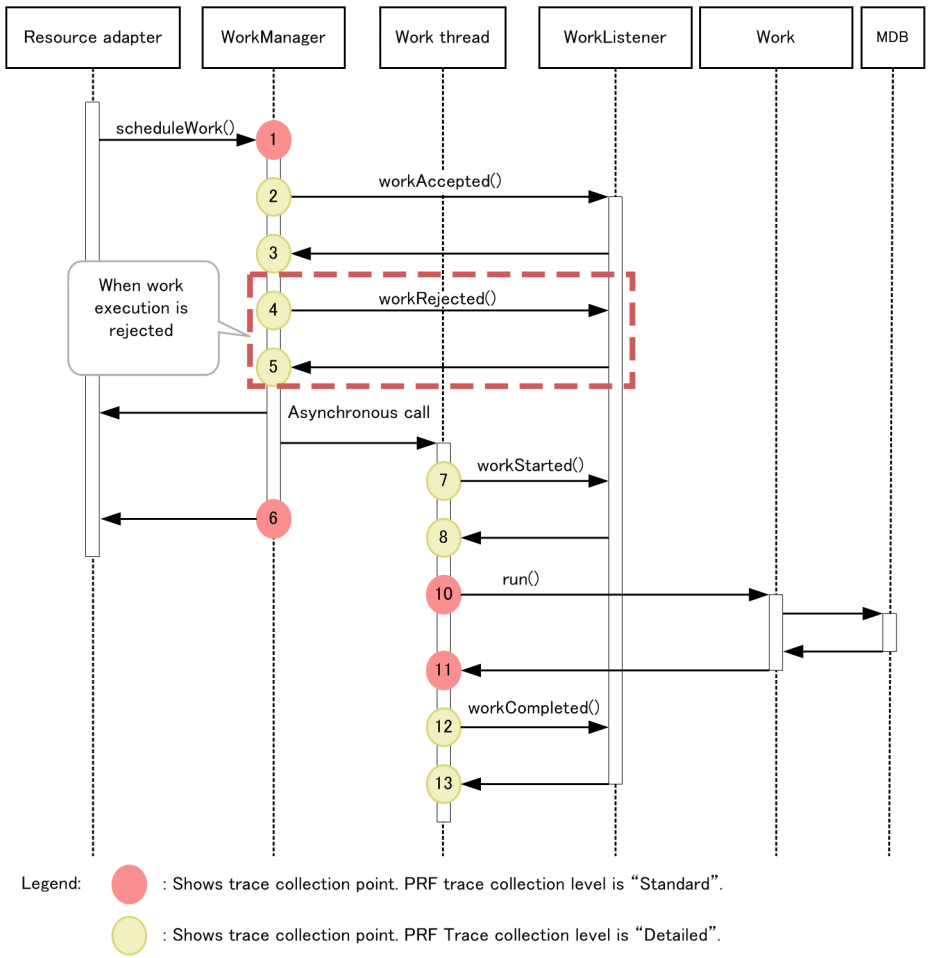


Figure 8–51: Details of trace collection points in a DB Connector and JCA container (when startWork() is invoked)

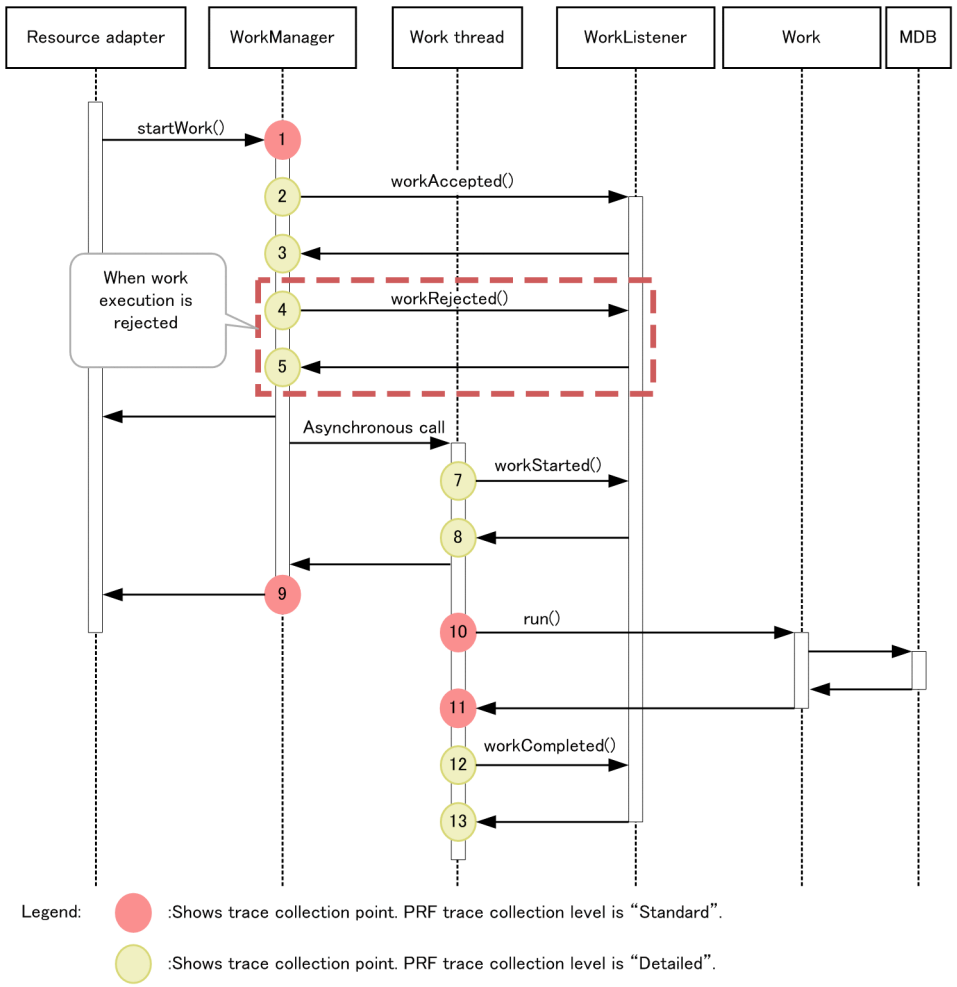
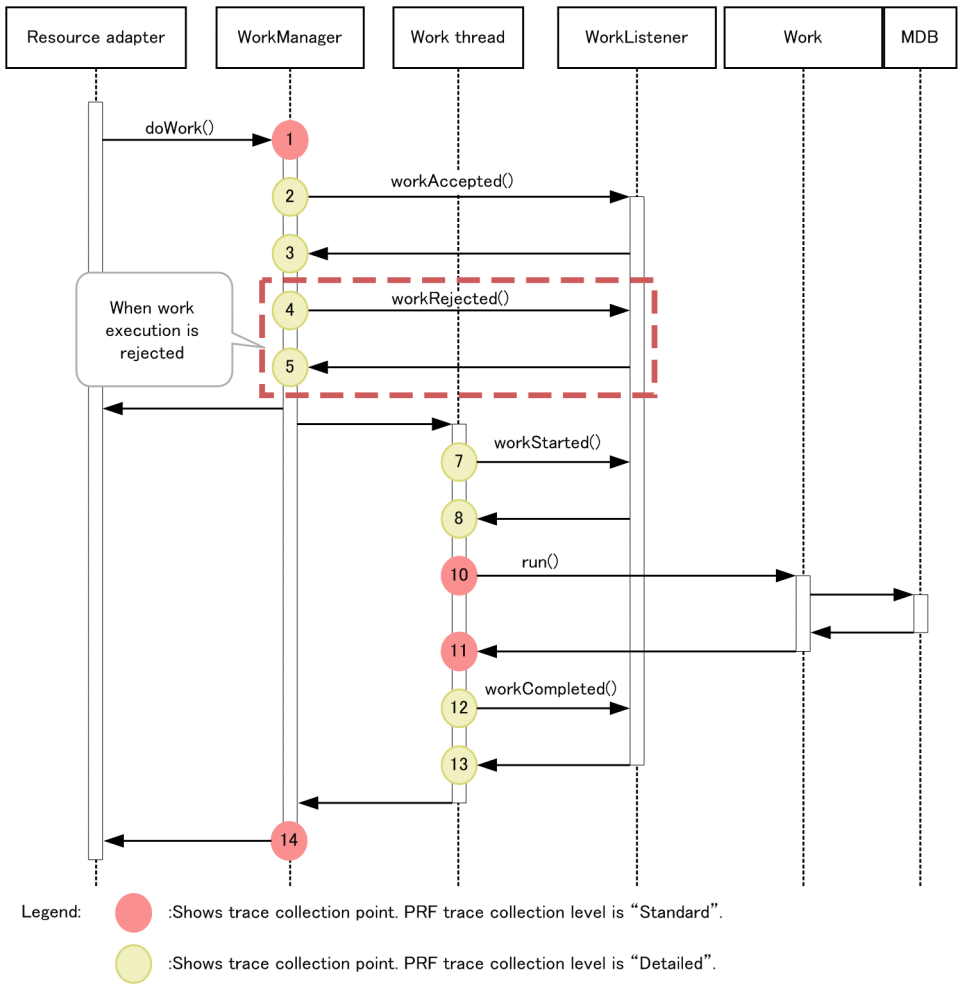


Figure 8–52: Details of trace collection points in a DB Connector and JCA container (when doWork() is invoked)



(2) Trace information that can be collected

The following table describes the trace information that can be collected.

Table 8–88: Details of trace collection points in a DB Connector and JCA container (when work management is used)

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8B86	A	Work class name	Method name	Display name of the resource adapter
2	0x8B88	B	Work class name	Method name	Display name of the resource adapter[#2]
3	0x8B89	B	Work class name	Method name	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
4	0x8B88	B	Work class name	Method name	Display name of the resource adapter[#2]
5	0x8B89	B	Work class name	Method name	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
6	0x8B87	A	Work class name	Method name	
7	0x8B88	B	Work class name	Method name	Display name of the resource adapter[#2]
8	0x8B89	B	Work class name	Method name	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
9	0x8B87	A	Work class name	Method name	
10	0x8B8A	A	Work class name	Method name	Display name of the resource adapter[#2]
11	0x8B8B	A	Work class name	Method name	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
12	0x8B88	B	Work class name	Method name	Display name of the resource adapter[#2]
13	0x8B89	B	Work class name	Method name	• When normal: *Entrance-time* • For an exception: *Entrance-time exception-name*
14	0x8B87	A	Work class name	Method name	

Legend:

A: Standard

B: Advanced

#1

Corresponds to the numbers from Figure 8-50 to Figure 8-52.

#2

In the case of a resource adapter included in the application, *application-name*: *display-name-of-resource-adapter* is displayed.

8.12 Trace collection points of an RMI

This section describes the trace collection points of an RMI, and the trace information that can be collected.

8.12.1 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 8–89: Details of trace collection points in an RMI

Event ID	No. in the figure#	Trace acquisition points	Level
0x8E01	1	When the process for sending a request with the stub starts	A
0x8E02	6	When the process for receiving a response with the stub is complete	A
0x8E03	2	During the processing for sending the request at the client side	A
0x8E04	5	During the processing for receiving the response at the client side	A
0x8E05	3	During the processing for receiving the request at the server side	A
0x8E06	4	During the processing for sending the response at the server side	A

Legend:

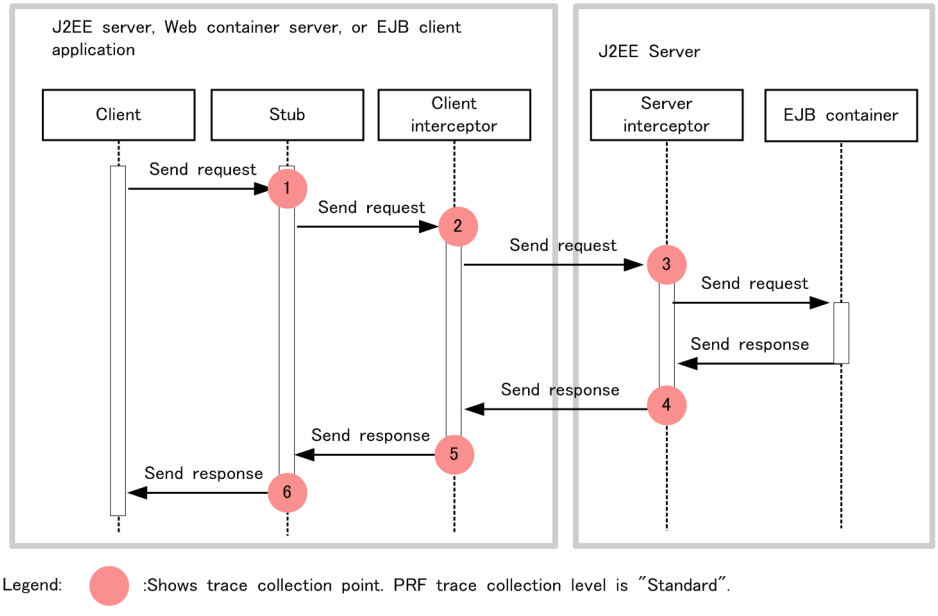
A: Standard

#

Corresponds to the numbers in Figure 8-53.

The following figure shows the trace collection points in an RMI.

Figure 8–53: Trace collection points of an RMI



8.12.2 Trace information that can be collected

The following table describes the trace information that can be collected in an RMI.

Table 8–90: Trace information that can be collected in an RMI

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8E01	A	Interface name	Operation name	--
2	0x8E03	A	--	--	--
3	0x8E05	A	--	--	--
4	0x8E06	A	--	--	--
5	0x8E04	A	--	--	--
6	0x8E02	A	Interface name[#2]	Operation name[#2]	--[#2]

Legend:

A: Standard

--: Not applicable

#1

Corresponds to the numbers in Figure 8-53.

#2

When an exception occurs, the interface name and operation name are not displayed. Furthermore, the exception that has occurred is displayed in the option.

8.13 Trace collection points of an OTS

This section describes the trace collection points of an OTS, and also the trace information that can be collected.

8.13.1 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 8–91: Details of trace collection points in an OTS

Event ID	No. in the figure#	Trace acquisition points	Level
0x9400	1	Immediately after a transaction is generated	A
0x9401	2	Immediately after the transaction status transits to `MarkedRollback`	A
0x9402	3	Immediately after the transaction status transits to `Rollingback`	A
0x9403	12	Immediately after the transaction concludes	A
0x9404	4	Immediately before the conclusion processing to `javax.transaction.xa.XAResource`	B
0x9405	5	Immediately after the conclusion processing to `javax.transaction.xa.XAResource`	B
0x9406	6	Immediately before the conclusion processing to a subordinate transaction	B
0x9407	9	Immediately after the conclusion processing to a subordinate transaction	B
0x9408	7	Immediately after receiving a request for conclusion processing from a superior transaction	B
0x9409	8	Immediately before replying to the request for conclusion processing from the superior transaction	B
0x9410	13	Immediately before acquiring the `javax.transaction.xa.XAResource` object	B
0x9411	14	Immediately after acquiring the `javax.transaction.xa.XAResource` object	B
0x9412	10	Immediately before writing into or reading from the status file	B
0x9413	11	Immediately after writing into or reading from the status file	B

Legend:

A: Standard

B: Advanced

#

Corresponds to the numbers from Figure 8-54 to Figure 8-59.

The following figure shows the trace collection points in an OTS.

Figure 8–54: Trace collection points from the generation of a transaction until its conclusion

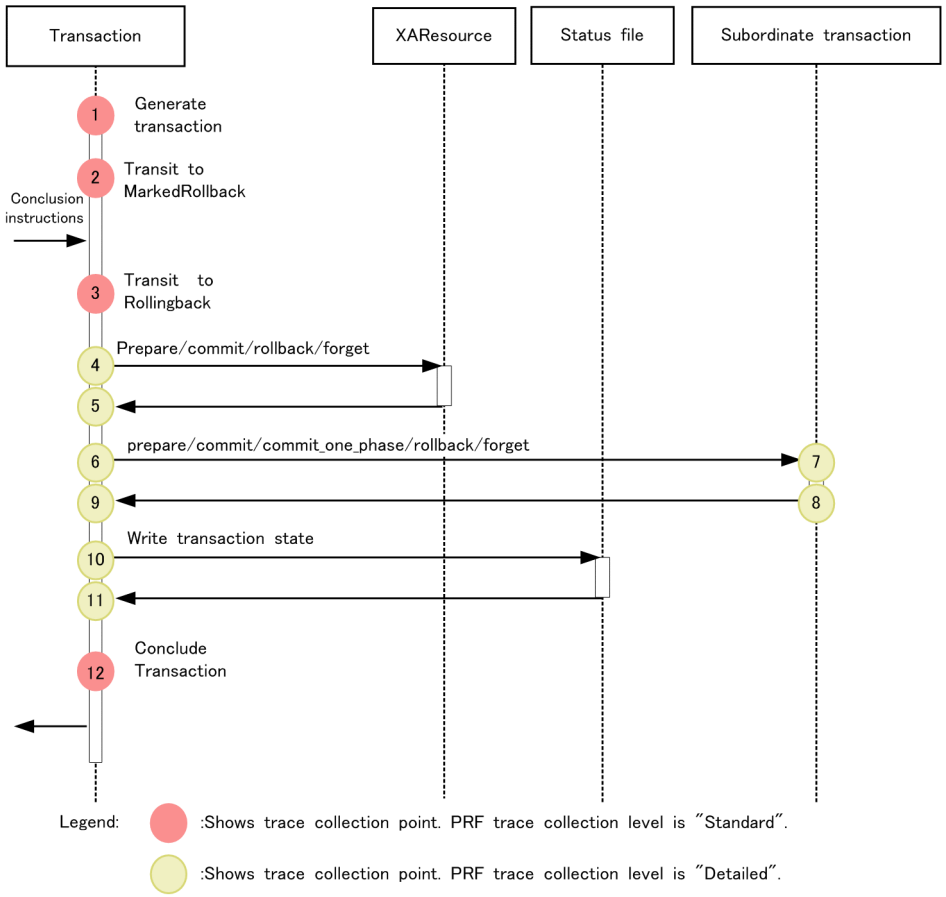


Figure 8–55: Trace collection points related to reading or writing into the status file during the in-process OTS initialization

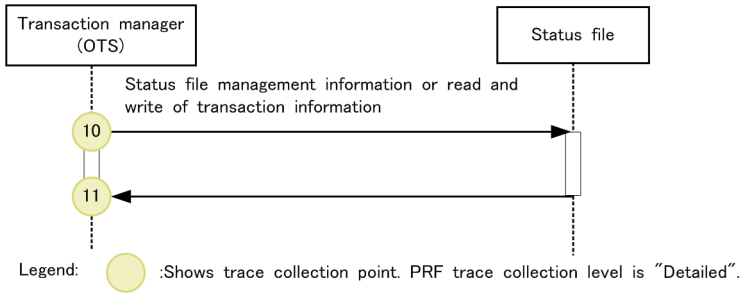


Figure 8–56: Trace collection points during the recovery process of javax.transaction.xa.Xid

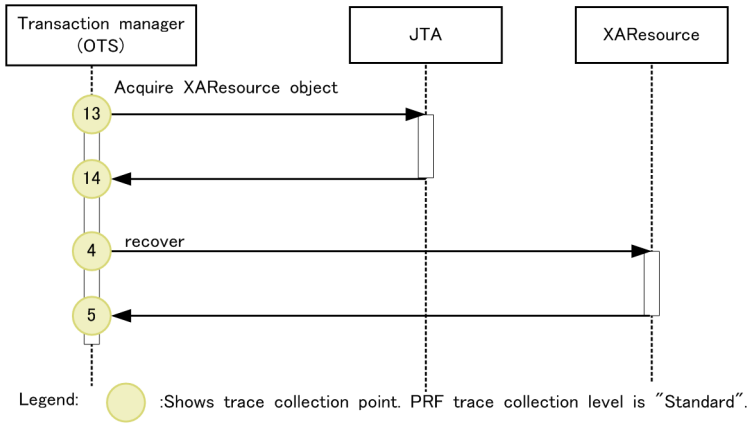


Figure 8–57: Trace collection points in a typical one-phase commit model

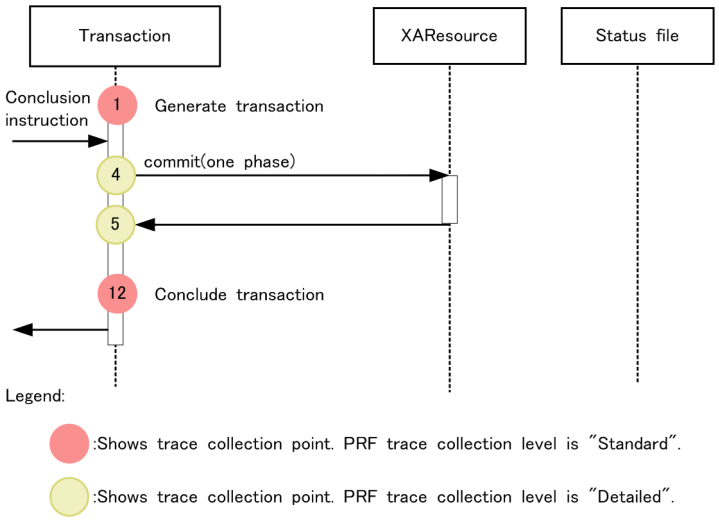


Figure 8–58: Trace collection points in a typical two-phase commit model

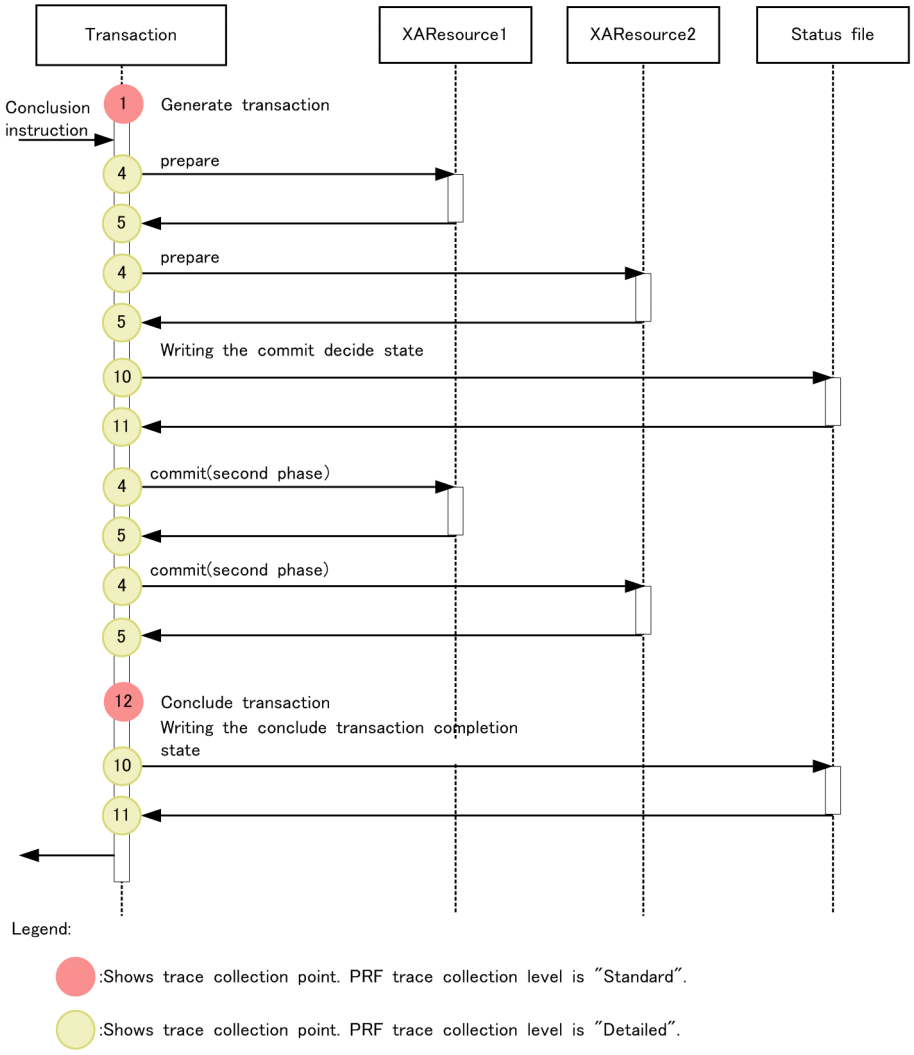
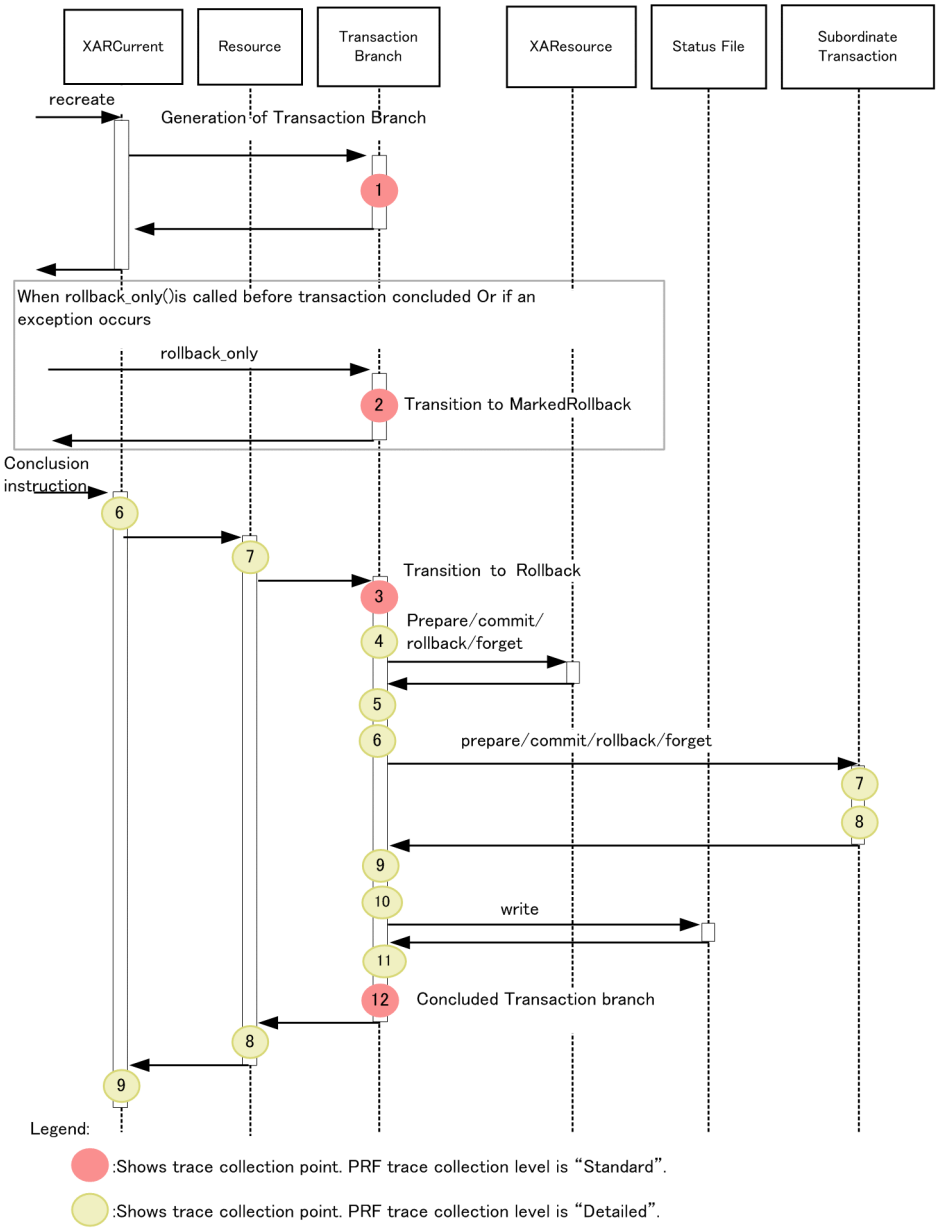


Figure 8–59: PRF acquisition information during transaction linkage with OpenTP1



8.13.2 Trace information that can be collected

The following table describes the trace information that can be collected in an OTS. Note that when more than one instance of information is described for a single item as different points, it implies that any one of those is output.

Table 8–92: Trace information that can be collected in an OTS

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x9400	A	global transaction	• `| created` Generated upon receiving an instruction for starting a transaction. • `| recreated`	*Global-transaction-ID*

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
				Generated upon receiving an instruction from another node for participating in a transaction processing. • `| recovered` Recovered from the status file. • `| recovered(orphan)` Xid was recovered from `javax.transaction.xa.XAResource`, but because no corresponding transaction existed, a new transaction was generated.	
2	0x9401	A	global transaction	`| marked rollback(`*reason-for-transition*[#2]`)`	*Global-transaction-ID*
3	0x9402	A	global transaction	`| rolling back(`*reason-for-transition*[#3]`)`	*Global-transaction-ID*
12	0x9403	A	global transaction	• `| committed` Committed. • `| rolled back` Rolled back. • `| heuristic commit` Committed forcibly. • `| heuristic rollback` Rolled back forcibly. • `| heuristic mixed` Committed and rolled back partially. • `| heuristic hazard` Not clear if committed or rolled back. • `| unknown` Unknown if committed or rolled back. • `| invalid status` Concluded with a status other than those described above.	*Global-transaction-ID*
4	0x9404	B	• *XAResource-identifier* • None	• `> prepare` Prepare is issued. • `> commit(second phase)` Second-phase commit is issued. • `> commit(one phase)` First-phase commit is issued. • `> rollback` Rollback is issued. • `> forget` Forget is issued. • `> recover` recover is issued.	• *Global-transaction-ID* • None

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
5	0x9405	B	• *XAResource-identifier* • None	In the case of normal return • `< prepare` • `< commit(second phase)` • `< commit(one phase)` • `< rollback` • `< forget` • `< recover` In the case of abnormal return (when an unexpected value is returned, or when an exception occurs) • `<!prepare` • `<!commit(second phase)` • `<!commit(one phase)` • `<!rollback` • `<!forget` • `<!recover`	• *Global-transaction-ID:result*[#4] • None
6	0x9406	B	resource	• `> prepare` Prepare is issued. • `> commit(second phase)` Second-phase commit is issued. • `> commit(one phase)` First-phase commit is issued. • `> rollback` Rollback is issued. • `> forget` Forget is issued.	*Global-transaction-ID*
9	0x9407	B	resource	In the case of normal return • `< prepare` • `< commit(second phase)` • `< commit(one phase)` • `< rollback` • `< forget` In the case of abnormal return (when an unexpected value is returned, or when an exception occurs) • `<!prepare` • `<!commit(second phase)` • `<!commit(one phase)` • `<!rollback` • `<!forget`	• *Global-transaction-ID* • *Global-transaction-ID:result*[#5]
7	0x9408	B	subordinate transaction	• `> prepare` A prepare instruction is received. • `> commit(second phase)` A second-phase commit instruction is received. • `> commit(one phase)` A first-phase commit instruction is received.	*Global-transaction-ID*

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
				• `> rollback` A rollback instruction is received. • `> forget` A forget instruction is received.	
8	0x9409	B	subordinate transaction	In the case of normal return • `< prepare` • `< commit(second phase)` • `< commit(one phase)` • `< rollback` • `< forget` In the case of abnormal return (when an unexpected value is returned, or when an exception occurs) • `<!prepare` • `<!commit(second phase)` • `<!commit(one phase)` • `<!rollback` • `<!forget`	• *Global-transaction-ID* • *Global-transaction-ID*:*result*[#6]
13	0x9410	B	• None • *XAResource-identifier*	`> get xaresource`	• None • *Global-transaction-ID*
14	0x9411	B	• None • *XAResource-identifier*	In the case of normal return • `< get xaresource` In the case of abnormal return (when an unexpected value is returned, or when an exception occurs) • `<!get xaresource`	• None • *result*[#7] • *Global-transaction-ID* • *Global-transaction-ID*:*result*[#7]
10	0x9412	B	• *Status-file-name* • *Status-file-name*:*entry-number*[#8]	• `> write(`*contents-to-be-written*[#9]`)` • `> read(`*contents-to-be-read*[#10]`)`	• None • *Global-transaction-ID*
11	0x9413	B	• *Status-file-name* • *Status-file-name*:*entry-number*[#8]	In the case of normal return • `< write` • `< read` In the case of abnormal return (when an unexpected value is returned, or when an exception occurs) • `<!write` • `<!read`	• None • *result*[#11] • *Global-transaction-ID* • *Global-transaction-ID*:*result*[#11]

Legend:

A: Standard

B: Advanced

#1

Corresponds to the numbers from Figure 8-54 to Figure 8-59.

#2

Any one of the following is output as the reason for transition:

- operation
  The instruction is received from outside the in-process OTS.
- server call
  An attempt to send a call to a server in another node has failed.
- superior
  An instruction for participating in the transaction processing was received from another node, but that transaction was already in the `MarkedRollback` status.
- sync before
  The callback processing to a JTA for which the conclusion processing is in progress has failed.

#3

Any one of the following is output as the reason for transition:

- operation
  The instruction is received from outside the in-process OTS.
- timeout
  The global transaction has timed out.
- superior
  The instruction is received from a superior transaction or the `cjrollbacktrn` command.
- forgotten
  It is determined that the transaction that outputs the conclusion instruction to the subordinate transaction, or `javax.transaction.xa.XAResource` does not exist.
- end
  An attempt to execute `end` for `javax.transaction.xa.XAResource` has failed.
- prepare
  An attempt to execute `prepare` for `javax.transaction.xa.XAResource` has failed.
- write prepared
  An attempt to write `prepared` in the status file has failed.
- write committing
  An attempt to write `committing` in the status file has failed.

#4

Any of the following is output as the result in the case of `prepare`, `commit`, `rollback`, or `forget`

- Return value
- Error code of the XAException
- toString() of the exception (an exception other than the XAException)

Any of the following is output as the result in the case of `recover`:

- Number of recovered Xids
- Null (when the Xid array itself is `null`)
- Error code of the XAException
- toString() of the exception (an exception other than the XAException)

#5

Any of the following is output as the result in the case of `prepare`:

- Returned value
- toString() of the exception

In cases other than `prepare`, toString() of the exception is output as the result.

#6

Any of the following is output as the result in the case of `prepare`:

- Value to be returned
- toString() of the exception

In cases other than `prepare`, toString() of the exception is output as the result.

#7

Any of the following is output as the result:

- toString() of the exception
- Null (when the return value is `null`)

#8

This is the internal information.

#9

Any of the following are output as the contents to be written:

- management info
  Status file management information
- status file body
  Status file body
- prepared
  Prepared status
- committing
  Commit determined status
- heuristic commit
  Forced commit status
- heuristic rollback
  Forced rollback status
- heuristic mixed
  Partially committed and rolled back status
- heuristic hazard
  Unclear committed or rolled status
- forgotten
  Transaction conclusion complete status

#10

Any of the following are output as the contents to be read:

- management info
  Status file management information
- status file body
  Status file body

#11

Any of the following is output as the result:

- Writing size (unit: bytes)
- Reading size (unit: bytes)
- toString() of the exception

8.14 Trace collection points of standard output, standard error output, and user log

This subsection describes the trace collection points of the standard output, standard error output, and user log, and also describes the trace information that can be collected.

8.14.1 Trace collection points of standard output or standard error output

(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 8–93: Details of trace collection points in the case of standard output or standard error output

Event ID	No. in the figure#	Trace acquisition points	Level
0x9C00	1	When the output to standard output or standard error output starts	B
0x9C01	2	When the output to standard output or standard error output is complete	B

Legend:

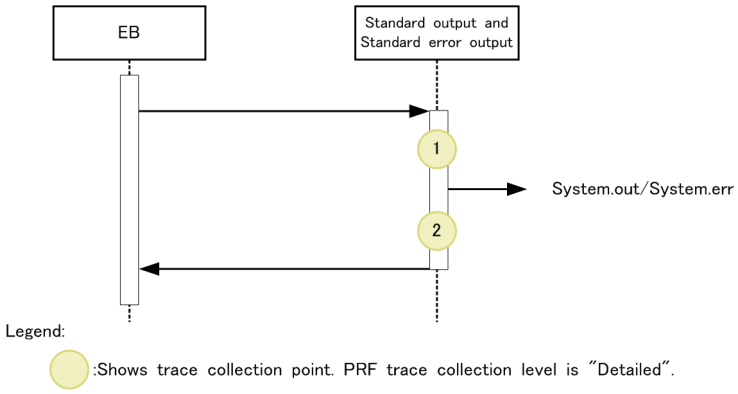
B: Advanced

#

Corresponds to the numbers in Figure 8-60.

The following figure shows the trace collection points in the case of standard output or standard error output.

Figure 8–60: Trace collection points of standard output or standard error output



(2) Trace information that can be collected

The following table describes the trace information that can be collected in the case of standard output or standard error output.

Table 8–94: Trace information that can be collected in the case of standard output or standard error output

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x9C00	B	Stream name (`out` or `err`)	--	--

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
2	0x9C01	B	Stream name (`out` or `err`)	--	--

Legend:

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-60.

8.14.2 Trace collection points of the user log

(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 8–95: Details of trace collection points of the user log

Event ID	No. in the figure#	Trace acquisition points	Level
0x9C02	1	When message output using the user log starts	B
0x9C03	2	When message output using the user log is complete	B

Legend:

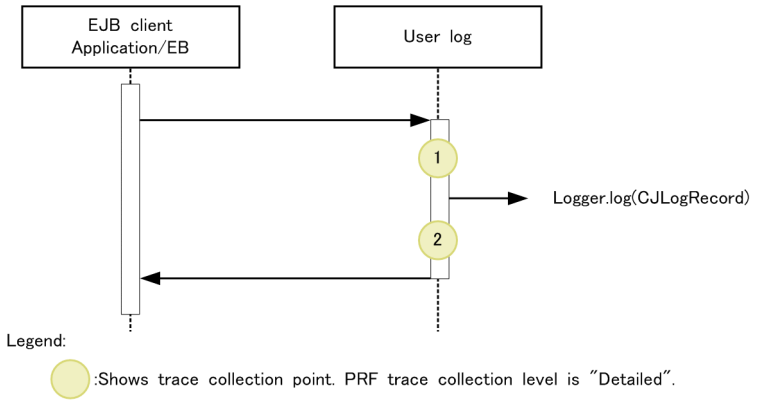
B: Advanced

#

Corresponds to the numbers in Figure 8-61.

The following figure shows the trace collection points of the user log.

Figure 8–61: Trace collection points of the user log



(2) Trace information that can be collected

The following table describes the trace information that can be collected in the user log.

Table 8–96: Trace information that can be collected in the user log

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x9C02	B	Application distinguished name	Message ID	--
2	0x9C03	B	Application distinguished name	Message ID	--

Legend:

B: Advanced

--: Not applicable

#

Corresponds to the numbers in Figure 8-61.

8.15 Trace collection points of a DI

This section describes the trace collection points of a DI, and the trace information that can be collected.

8.15.1 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 8–97: Details of trace collection points in a DI

Event ID	No. in the figure#	Trace acquisition points	Level
0x9D00	1	Immediately before injecting the dependency	A
0x9D01	2	Immediately after injecting the dependency	A

Legend:

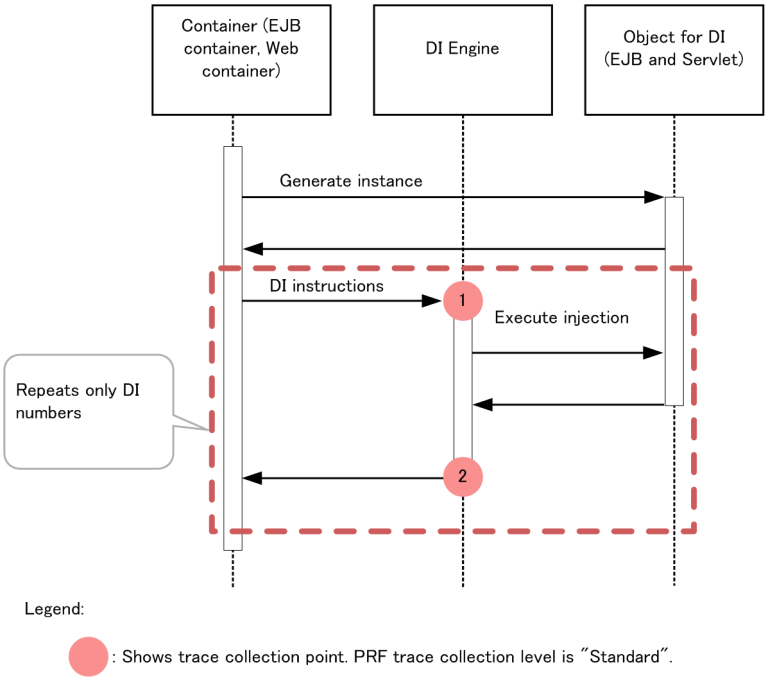
A: Standard

#

Corresponds to the numbers in Figure 8-62.

The following figure shows the trace collection points in a DI.

Figure 8–62: Trace collection points of a DI



8.15.2 Trace information that can be collected

The following table describes the trace information that can be collected in a DI.

Table 8–98: Trace information that can be collected in a DI

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x9D00	A	Target name of the destination where dependency is to be injected	Reference name to be injected	--
2	0x9D01	A	Target name of the destination where dependency is injected	Injected reference name	#2

Legend:

A: Standard

--: Not applicable

#1

Corresponds to the numbers in Figure 8-62.

#2

When the processing is performed normally, the entrance time is displayed.

When an exception occurs, the entrance time and exception are displayed.

8.16 Trace collection points of the batch application execution functionality

This section describes the trace collection points of the batch application execution functionality, and the trace information that can be collected.

8.16.1 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 8–99: Details of trace collection points in the batch application execution functionality

Event ID	Number in the figure[#1]	Trace acquisition point	Level
0x8E05	1[#2]	Just after receiving the batch application execution request	A
0xA100	2	Immediately before the execution of the batch application	A
0xA101	3	Immediately after the termination of the batch application	A
0x8E06	4[#2]	Immediately before sending the termination results of the batch application	A

Legend:

A: Standard

#1

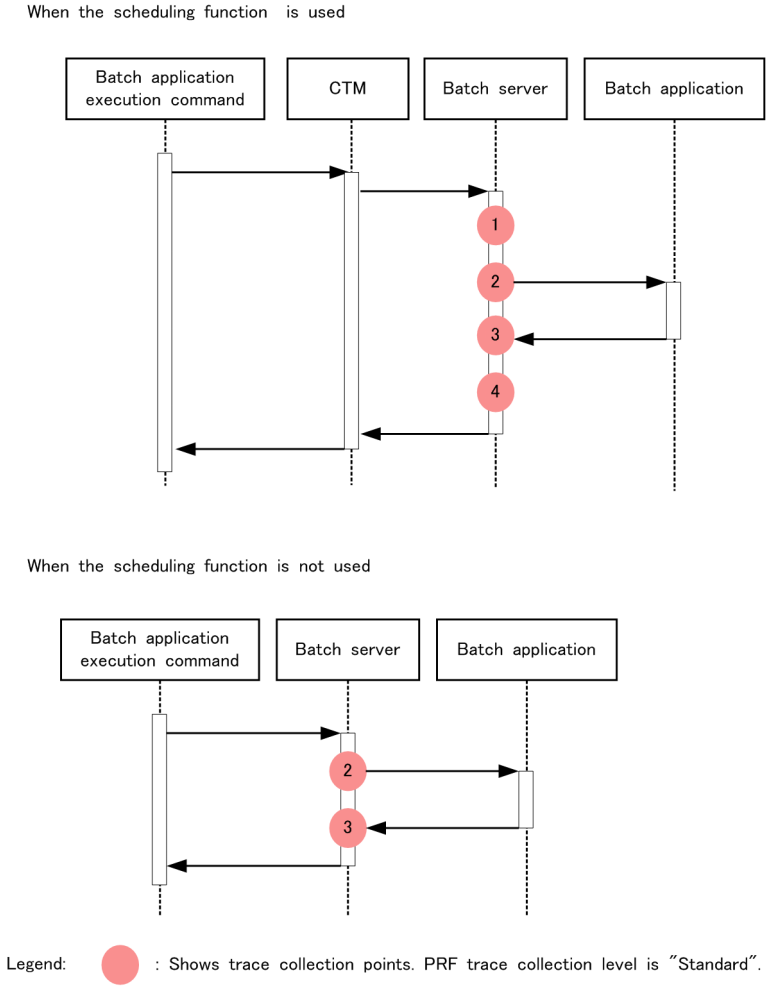
Corresponds to the numbers in Figure 8-63.

#2

The trace is only collected when the scheduling function is used.

The following figure shows the trace collection points in the batch application execution functionality.

Figure 8–63: Trace collection points in the batch application execution functionality



8.16.2 Trace information that can be collected

The following table describes the trace information that can be collected in the batch application execution functionality.

Table 8–100: Trace information that can be collected in the batch application execution functionality

No. in the figure[#1]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0x8E05	A	--	--	--
2	0xA100	A	Class name of the batch application (including the package name)	--	--
3	0xA101	A	Class name of the batch application (including the package name)	--	#2
4	0x8E06	A	--	--	--

Legend:

A: Standard

--: Not applicable

#1

Corresponds to the numbers in Figure 8-63.

#2

When the processing is performed normally, the entrance time is displayed.
When an exception occurs, the entrance time and exception are displayed.

8.17 Trace collection points of the TP1 inbound integrated function

This section describes the trace collection points of the TP1 inbound integrated function and the trace information that can be collected.

8.17.1 Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels:

(1) Details of trace collection points of the TP1 inbound integrated function

The following table describes the details of trace collection points of the TP1 inbound integrated function.

Table 8–101: Details of trace collection points of the TP1 inbound integrated function

Event ID	Number in the figure#	Trace collection point	Level
0x842F	19	Immediately after invoking the message listener method of Message-driven Bean from the resource adapter	A
0x8430	22	Immediately before the message listener method of Message-driven Bean that is invoked from the resource adapter returns	A
0x8431	20	Immediately before the EJB container calls back the message listener method of Message-driven Bean	A
0x8432	21	Immediately after returning from the callback of the message listener method of Message-driven Bean	A
0x8825	17	Immediately before the completion of recreate process of the transaction	B
0x8826	18	Immediately after the completion of recreate process of the transaction	B
0x8B86	14	Immediately after invoking the method of `javax.resource.spi.work.WorkManager`	A
0x8B87	15	Immediately before returning the method of `javax.resource.spi.work.WorkManager`	A
0x8B8A	16	Immediately before invoking the method of `javax.resource.spi.work.Work`	A
0x8B8B	29	Immediately after returning the method of `javax.resource.spi.work.Work`	A
0x8B8C	33	Immediately after invoking `javax.resource.spi.XATerminator.prepare` (Xid xid)	B
0x8B8D	34	Immediately before returning `javax.resource.spi.XATerminator.prepare` (Xid xid)	B
0x8B8E	--	Immediately after invoking `javax.resource.spi.XATerminator.commit` (Xid xid, boolean onePhase)	B
0x8B8F	--	Immediately before returning `javax.resource.spi.XATerminator.commit` (Xid xid, boolean onePhase)	B

Event ID	Number in the figure#	Trace collection point	Level
0x8B90	--	Immediately after invoking `javax.resource.spi.XATerminator.rollback` (Xid xid)	B
0x8B91	--	Immediately before returning `javax.resource.spi.XATerminator.rollback` (Xid xid)	B
0x8B92	--	Immediately after invoking `javax.resource.spi.XATerminator.forget` (Xid xid)	B
0x8B93	--	Immediately before returning `javax.resource.spi.XATerminator.forget` (Xid xid)	B
0xAA00	8	Immediately after the message reception starts	A
0xAA01	12	Immediately after the sending of RPC response is completed	A
0xAA02	4	Immediately after the reading of the message from OpenTP1 starts	A
0xAA03	10	Immediately before disconnecting from OpenTP1 (when `rpc_close_after_send=Y` is specified in OpenTP1 definition)	A
0xAA04	5	Immediately before invoking socket `read`	B
0xAA05	6	Immediately after completing socket `read`	B
0xAA06	13	Immediately before inserting the message into the schedule queue once the message reception is completed.	A
0xAA08	23	Immediately before the sending of RPC response starts	A
0xAA09	27	Immediately before disconnecting the connection for response (when `rpc_close_after_send=true` is specified in TP1 inbound adaptor property)	A
0xAA0A	25	Immediately before invoking socket `write`	B
0xAA0B	26	Immediately after completing socket `write`	B
0xAA0C	30	Sending has failed. Immediately before the standby for retry starts	A
0xAA0D	31	Immediately before retry starts	A
0xAA10	11	• Immediately after disconnecting the connection, if an invalid message is received • Immediately after disconnecting the connection, if reception timeout occurs • Immediately after reception timer monitoring thread detects reception timeout for RPC requests and destroys the message	A
0xAA11	28	Immediately after an attempt to send the RPC response fails	A
0xAA12	2	Immediately before including the connection with OpenTP1 in reception monitoring	B
0xAA13	3	When message reception is detected in the connection with OpenTP1	B
0xAA14	32	Immediately after the completion of reception from OpenTP1 during synchronization point processing	A
0xAA15	35	Immediately after the completion of sending to OpenTP1 during synchronization point processing or immediately after an error occurs in the sending process	A
0xAA16	1	Immediately after the connection with OpenTP1 is established	A

Event ID	Number in the figure#	Trace collection point	Level
0xAA17	9	Immediately before disconnecting the connection with OpenTP1 or immediately after disconnection from OpenTP1, since a communication error is detected	A
0xAA18	7	Immediately after analyzing the received message	B
0xAA19	24	Immediately before sending the message	B

Legend:

A: Standard

B: Advanced

--: Not applicable

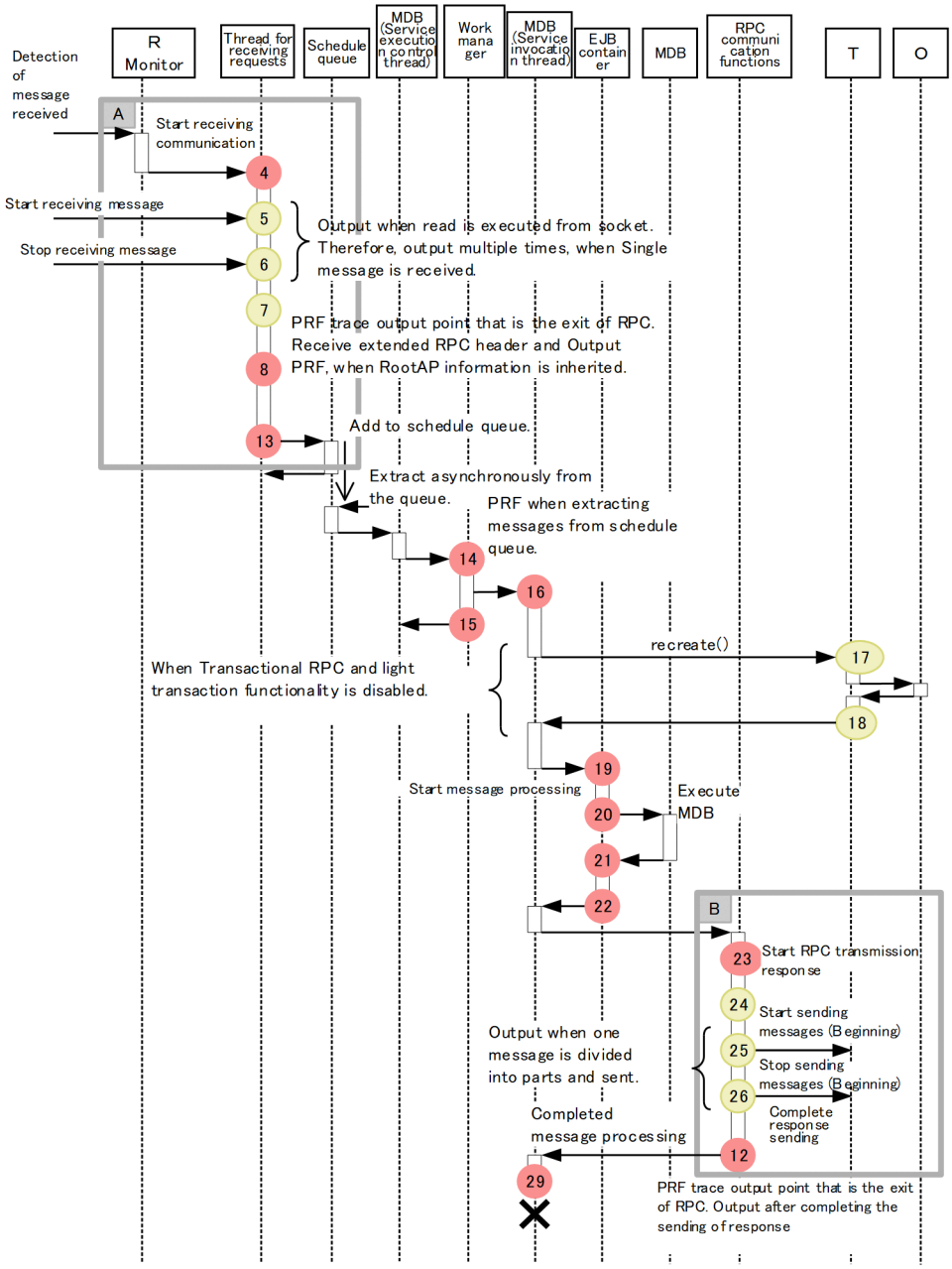
#

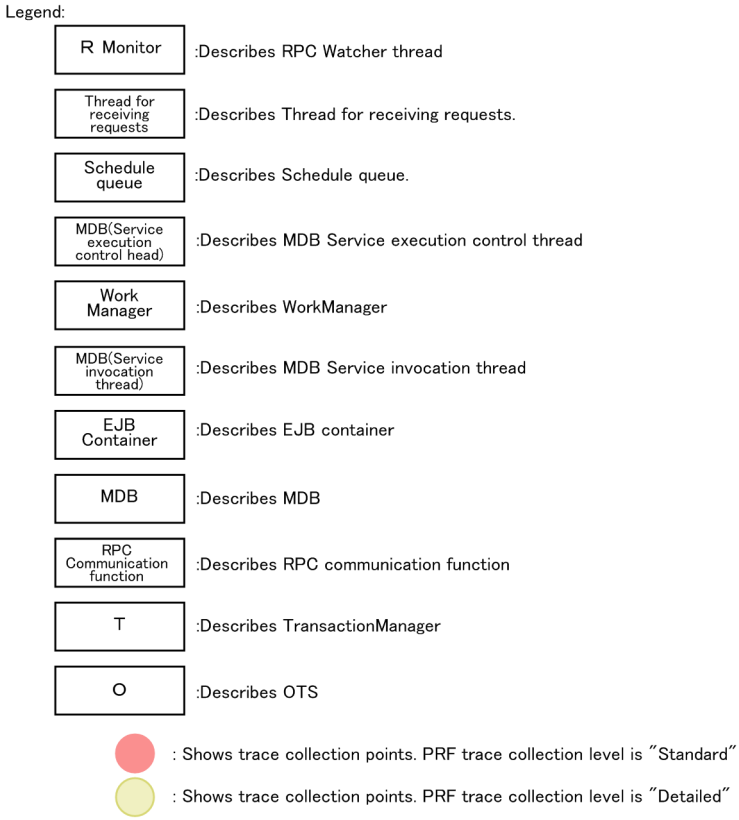
Corresponds to the numbers from Figures 8-64 to 8-70.

(2) Trace collection points of the TP1 inbound integrated function

The following figure shows the overall trace collection points in the TP1 inbound integrated function. Note that there are detailed trace collections points for A and B parts of the figure.

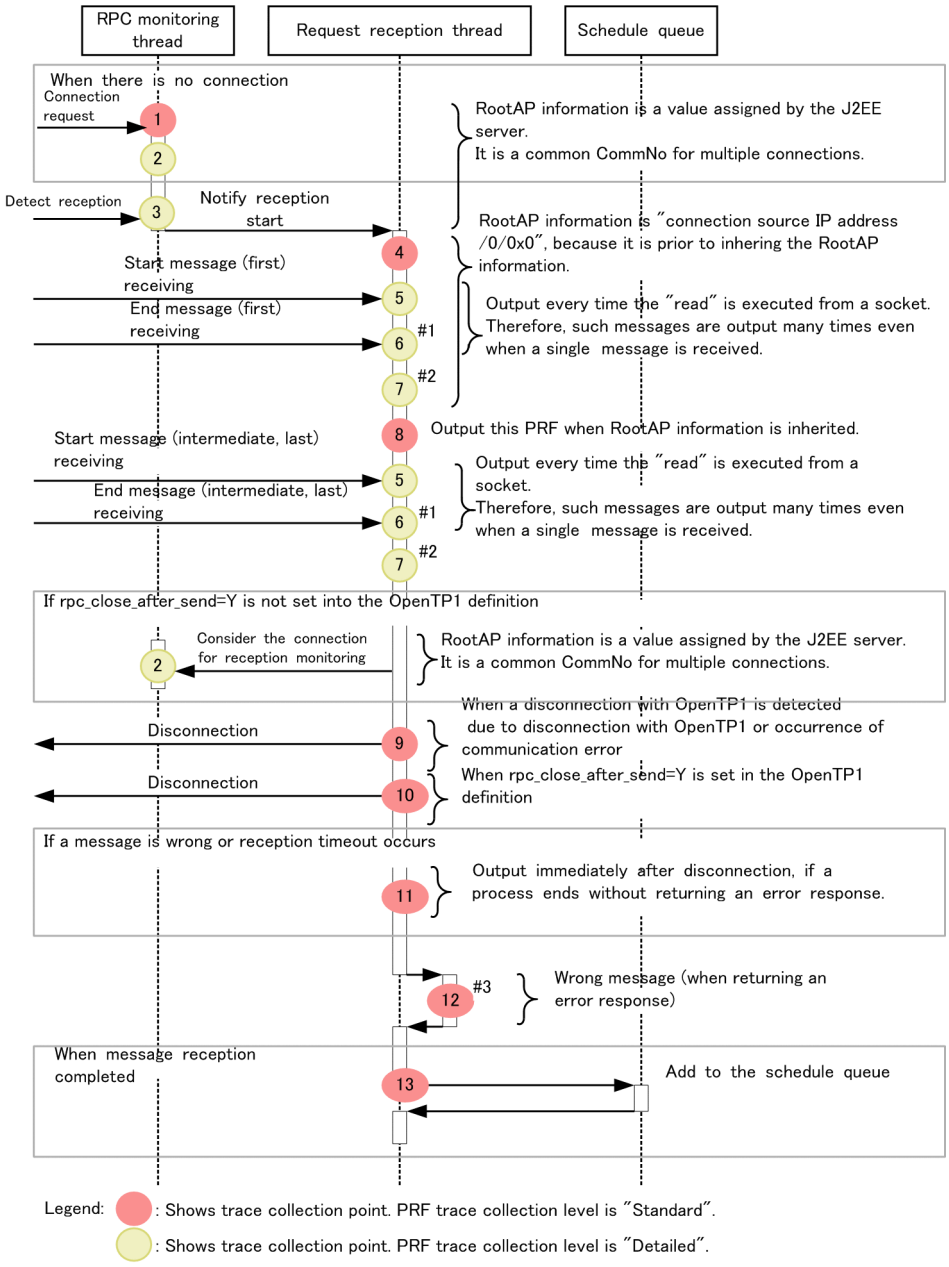
Figure 8–64: Overall trace collection points in the TP1 inbound integrated function





The following two figures show the detailed trace collections points for A and B parts in the above figure.

Figure 8–65: Trace collection points of part A



#1

Nothing is output, if an exception occurs in read.

#2

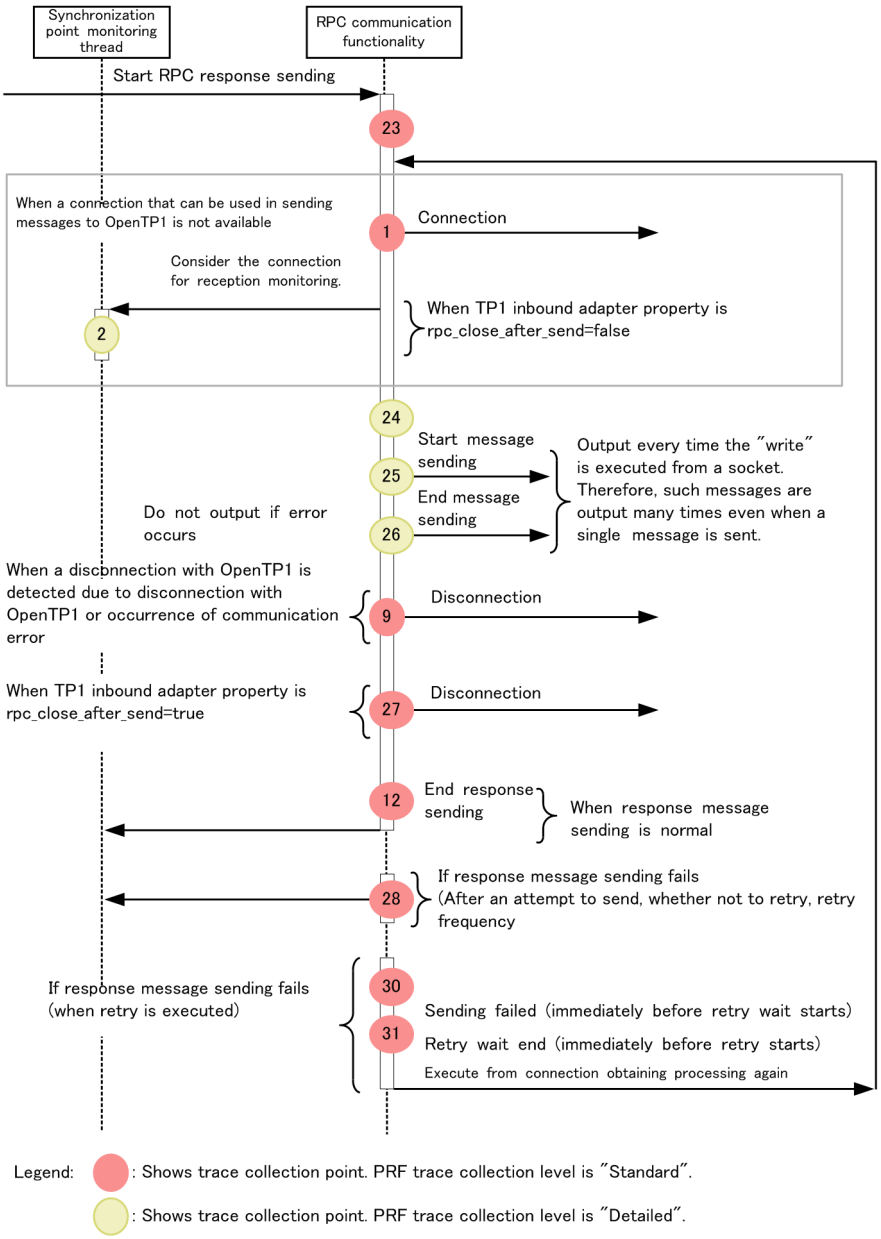
This is output after message reception. For segmented mails, the output is generated for each segmented mail. However, nothing is output for invalid messages.

#3

For processing of RPC response sending (PRF trace collection points other than 0xAA01), see *Figure 8-66*.

In addition to Figure 8-65, the PRF trace collection point for RPC reception is 0xAA10 that is output immediately after the reception timer monitoring thread detects a reception timeout for RPC requests and destroys the message.

Figure 8–66: Trace collection points of part B

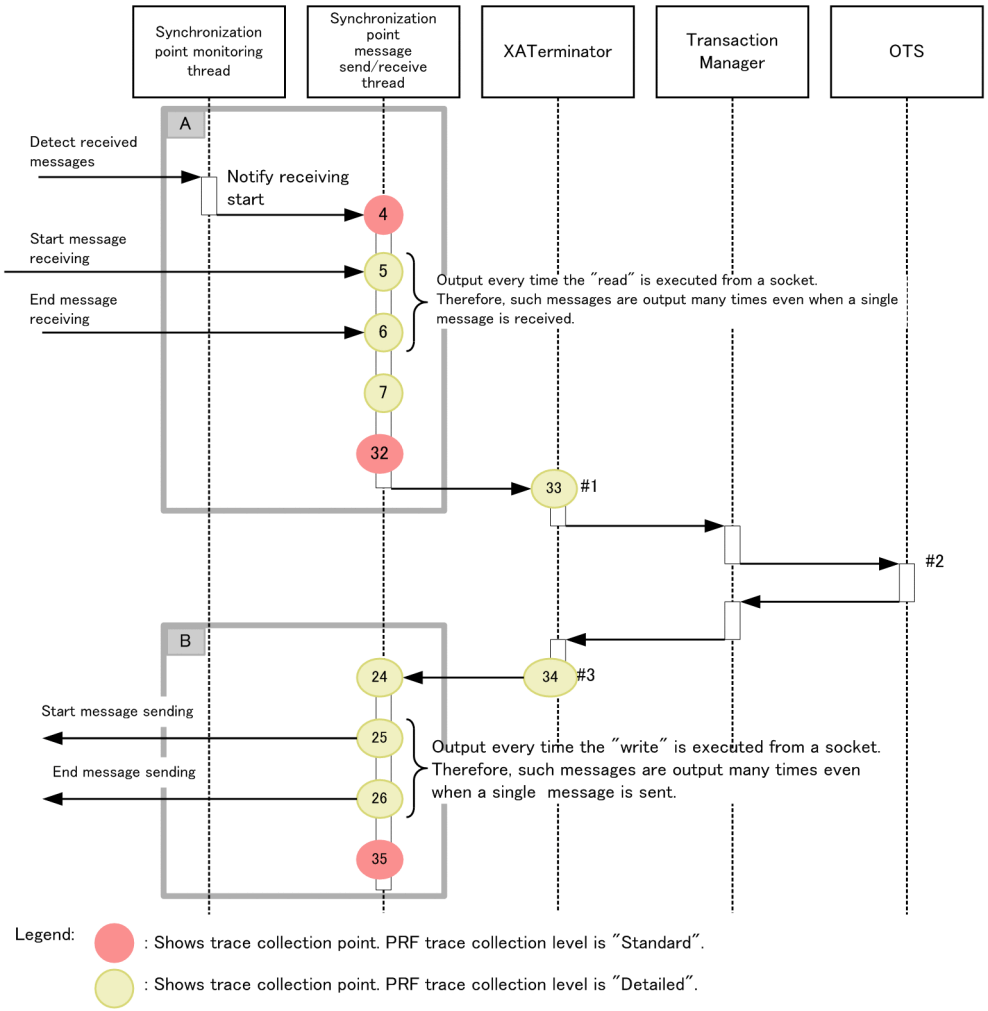


For the PRF trace output point of response sending in Figure 8-66, the trace is not only output during RPC communication, but also when J2EE applications stop. The root application information at this time is not the root application information inherited from OpenTP1, but the root application information generated by the J2EE server. Therefore, use the root application information, output to the interface name of 0xAA08 when the J2EE applications stop, to associate the PRF trace output points for stopping J2EE applications and the other PRF trace output points for RPC communication.

(3) Trace point collection in synchronous point processing

The following figure describes the trace collection points in the synchronous point processing. Note that points A and B in the figure have further detailed trace collection points.

Figure 8–67: Overview of the trace collection points in the synchronous point processing



#1

When the processing request is passed on to XATerminator, different Event IDs are output for each method. The following table describes the mapping between each method and its Event ID.

Table 8–102: Method and Event ID mapping

Method	Event ID	Number used in the figure
`prepare` method	0x8B8C	33
`commit` method	0x8B8E	--
`rollback` method	0x8B90	--
`forget` method	0x8B92	--

Legend:

--: Not used in the figure

#2

For OTS output points, see *8.13 Trace collection points of an OTS*.

#3

When the processing request is passed on from XATerminator to the synchronous point mail sending thread, different Event IDs are output for each method. The following table describes the mapping between each method and its Event ID.

Table 8–103: Method and Event ID mapping

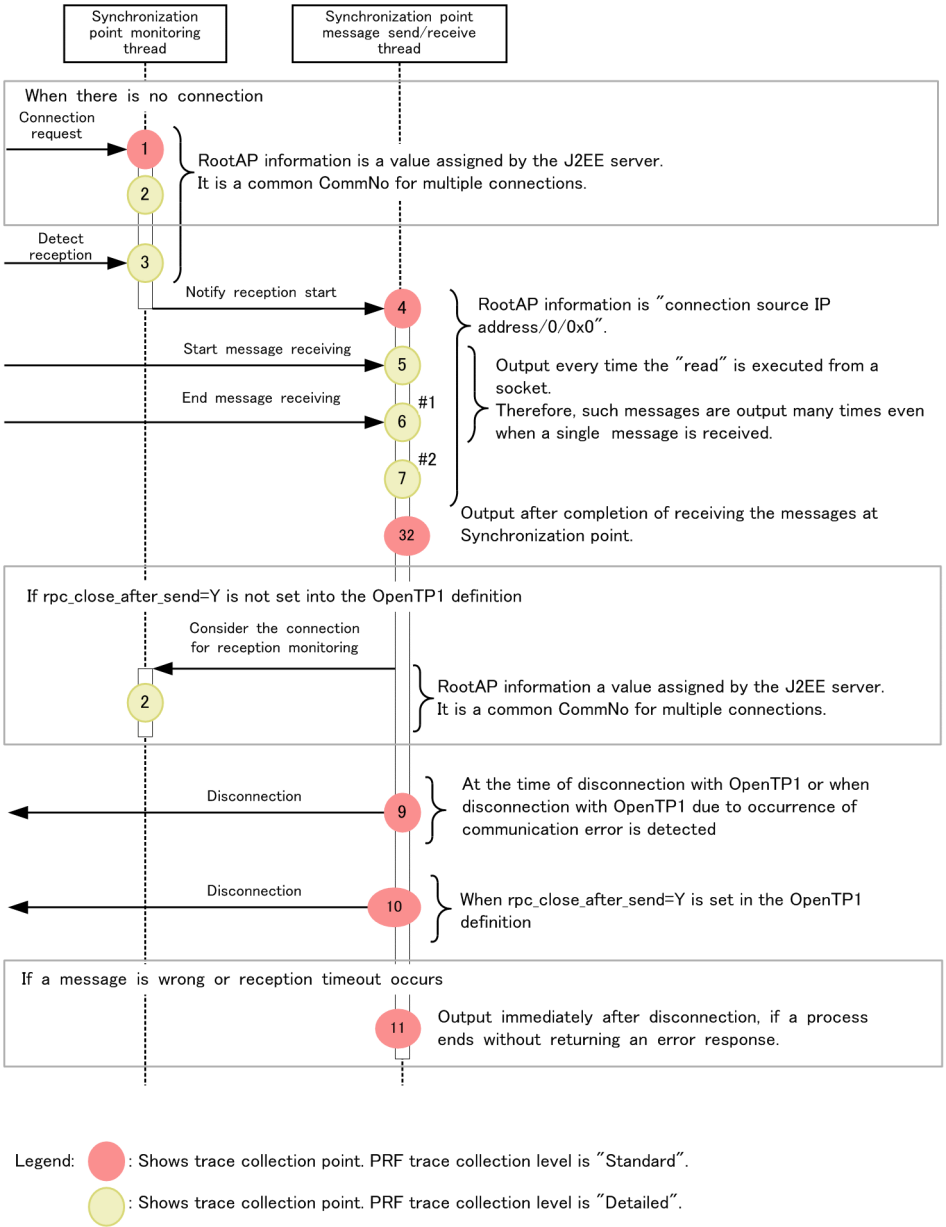
Method	Event ID	Number used in the figure
`prepare` method	0x8B8D	34
`commit` method	0x8B8F	--
`rollback` method	0x8B91	--
`forget` method	0x8B93	--

Legend:

--: Not used in the figure

For the detailed trace collection points of A and B in the figure, see the following 2 figures.

Figure 8–68: Trace collection point of the A part in the synchronous point processing



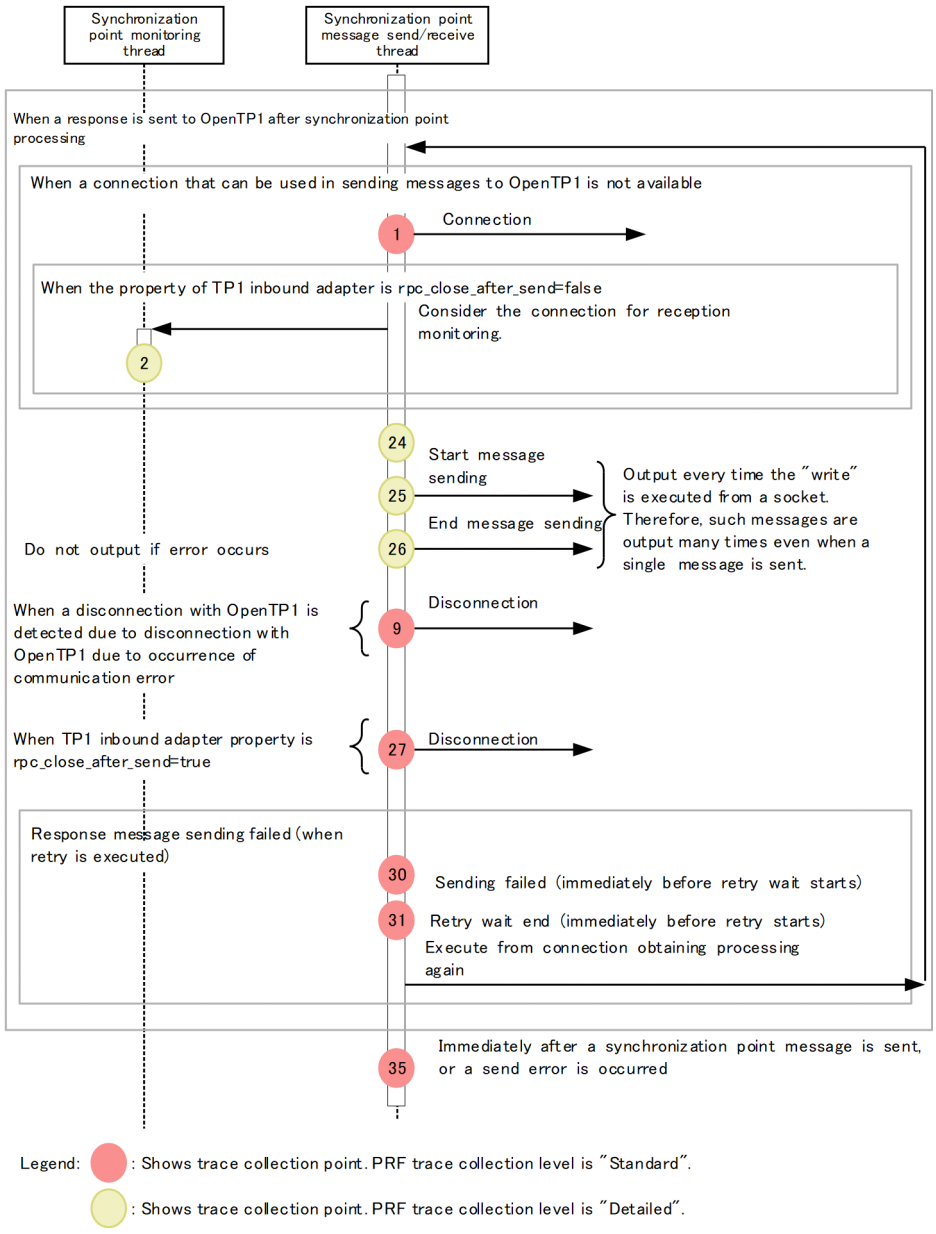
#1

Nothing is output, if read throws an exception.

#2

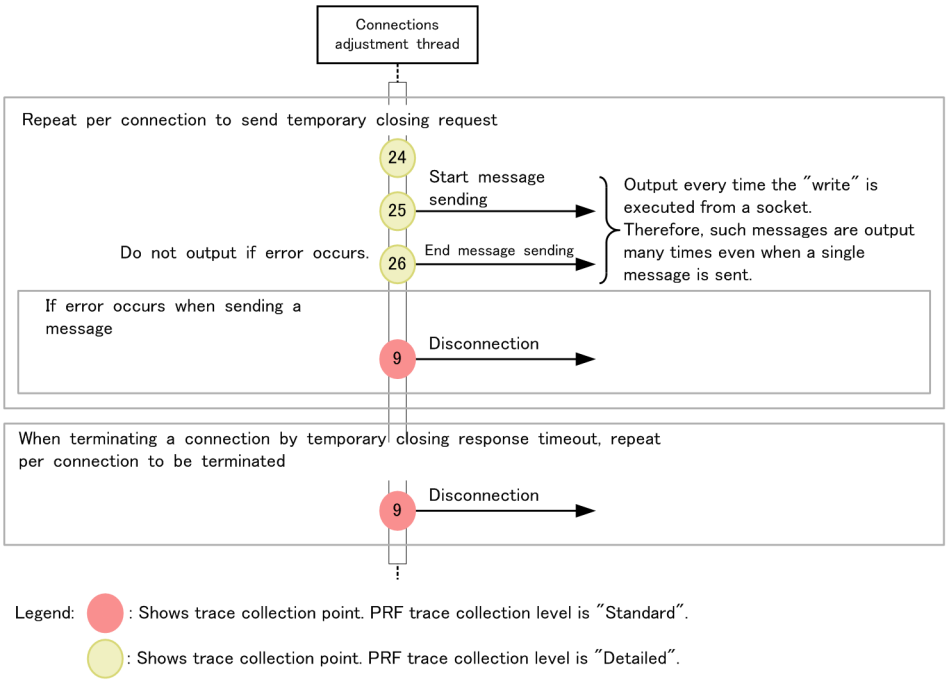
Outputs after receiving the mail. For segmented mails, output is generated for each segmented mail. However, nothing is output, for invalid mails.

Figure 8–69: Trace collection point of the B part in the synchronous point processing



The following figure shows the trace collection points for the connection count adjustment thread.

Figure 8–70: Trace collection points for the connection count adjustment thread



8.17.2 Trace information that can be collected

The following table describes the trace information that can be collected for the TP1 inbound integrated function.

Table 8–104: Trace information that can be collected for the TP1 inbound integrated function

Number in the figure[#1]	Event ID	Level	Information that can be collected		
			Interface name	Operation name	Option
1	0xAA16	A	IP address of OpenTP1: listen port number of OpenTP1[#2]	--	Listen port number of TP1 inbound adapter side
2	0xAA12	B	IP address of OpenTP1: listen port number of OpenTP1[#2]	--	Listen port number of TP1 inbound adapter side
3	0xAA13	B	IP address of OpenTP1: listen port number of OpenTP1[#2]	--	Listen port number of TP1 inbound adapter side
4	0xAA02	A	IP address of OpenTP1: listen port number of OpenTP1[#2]	--	Listen port number of TP1 inbound adapter side
5	0xAA04	B	Request size	--	--
6	0xAA05	B	Read size	--	--
7	0xAA18	B	IP address of OpenTP1: listen port number of OpenTP1[#2]	Message type	Listen port number of TP1 inbound adapter side
8	0xAA00	A	Service group name	Service name	Node identifier of OpenTP1 that invokes the

Number in the figure[#1]	Event ID	Level	Information that can be collected		
			Interface name	Operation name	Option
					service[#3], transaction global identifier (for transactional RPC)
9	0xAA17	A	IP address of OpenTP1: listen port number of OpenTP1[#2]	--	Listen port number of TP1 inbound adapter side
10	0xAA03	A	IP address of OpenTP1: listen port number of OpenTP1[#2]	--	Listen port number of TP1 inbound adapter side
11	0xAA10	A	Service group name[#4]	Service name [#5]	Entrance time
12	0xAA01	A	Service group name	Service name	When successful: Entrance time In the case of an error: Entrance time and error response code
13	0xAA06	A	Service group name	Service name	Node identifier of OpenTP1 that invokes the service[#3] , number of messages in the queue
14	0x8B86	A	Work class name	Method name	Resource adapter display name
15	0x8B87	A	Work class name	Method name	When normal: Entrance time In the case of an error: Entrance time and exception
16	0x8B8A	A	Work class name	Method name	Resource adapter display name
17	0x8825	B	--	--	Global transaction ID of OpenTP1 (byte column)
18	0x8826	B	--	--	When normal: Entrance time For an error: Entrance time and exception
19	0x842F	A	Bean name	Method name	--
20	0x8431	A	Bean name	Method name	--
21	0x8432	A	Bean name	Method name	When normal: Entrance time For an error: Entrance time and exception

Number in the figure[#1]	Event ID	Level	Information that can be collected		
			Interface name	Operation name	Option
22	0x8430	A	Bean name	Method name	When normal: Entrance time In the case of an error: Entrance time and exception
23	0xAA08	A	Root application information[#6]	--	--
24	0xAA19	B	IP address of OpenTP1: listen port number of OpenTP1[#2]	Message type	Listen port number of TP1 inbound adapter side
25	0xAA0A	B	Writing size	--	--
26	0xAA0B	B	Written size	--	--
27	0xAA09	A	IP address of OpenTP1: listen port number of OpenTP1[#2]	--	Listen port number of TP1 inbound adapter side
28	0xAA11	A	Service group name	Service name	Entrance time
29	0x8B8B	A	Work class name	Method name	When normal: Entrance time For an error: Entrance time and exception
30	0xAA0C	A	--	--	--
31	0xAA0D	A	--	--	--
32	0xAA14	A	Node identifier of OpenTP1	Output one of the following • prepare • commit • rollback	Transaction global identifier
33	0x8B8C	B	--	--	Global transaction ID of xid argument of XATerminator (byte column)
34	0x8B8D	B	--	--	When normal: Entrance time For an error: Entrance time and exception
35	0xAA15	A	--	In normal cases output one of the following:[#7] • prepared • read only • committed • rolled back No output in the case of an error.	Entrance time
--	0x8B8E	B	--	--	Global transaction ID of xid argument of XATerminator (byte column)

Number in the figure[#1]	Event ID	Level	Information that can be collected		
			Interface name	Operation name	Option
--	0x8B8F	B	--	--	When normal: Entrance time For an error: Entrance time and exception
--	0x8B90	B	--	--	Global transaction ID of xid argument of XATerminator (byte column)
--	0x8B91	B	--	--	When normal: Entrance time For an error: Entrance time and exception
--	0x8B92	B	--	--	Global transaction ID of xid argument of XATerminator (byte column)
--	0x8B93	B	--	--	When normal: Entrance time For an error: Entrance time and exception

Legend:

A: Standard

B: Advanced

--: Not applicable

#1

Corresponds to the numbers from Figures 8-64 to 8-70.

#2

If the listen port of OpenTP1 cannot be determined, "IP address of OpenTP1: --" is output.

#3

If the PRF information is inherited from OpenTP1 that invokes the service, "root RPC node identifier" is output. If the information is not inherited, "invocation source node ID" is output.

#4

If the service group name output in the interface name cannot be determined, "--" is output.

#5

If the service name output in the operation name cannot be determined, "--" is output.

#6

Outputs only when the J2EE applications stop. The following contents are output:

- If the PRF information is inherited from OpenTP1 that invokes the service, the root application information inherited from OpenTP1 is output.
- If the PRF information is not inherited from OpenTP1 that invokes the service, the root application information numbered by the TP1 inbound adapter is output.

#7

These are output in the following respective conditions:

- prepared: When prepare is completed
- read only: When the result of prepare is read only (such as when access to the resouce is by reference) or when the transaction setting of Message-driven Bean (service) that is executed is CMT NotSupported or BMT
- committed: When commit is completed
- rolled back: When rollback is completed. This includes the cases when the result of prepare and commit is treated as rollback.

8.18 Trace collection points of Cosminexus JMS Provider

This section describes the trace collection points of Cosminexus JMS Provider and the trace information that can be collected.

8.18.1 Trace collection points of the JMS ConnectionFactory interface and the trace information that can be collected

(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 8–105: Details of the trace collection points in the JMS ConnectionFactory interface

Event ID	Number in the figure#	Trace collection point	Level
0xA600	1	When the processing of `ConnectionFactory.createConnection()` starts	A
0xA601	4	When the processing of `ConnectionFactory.createConnection()` ends	A
0xA602	1	When the processing of `ConnectionFactory.createConnection(uname,pwd)` starts	A
0xA603	4	When the processing of `ConnectionFactory.createConnection(uname,pwd)` ends	A
0xA604	1	When the processing of `QueueConnectionFactory.createQueueConnection()` starts	A
0xA605	4	When the processing of `QueueConnectionFactory.createQueueConnection()` ends	A
0xA606	1	When the processing of `QueueConnectionFactory.createQueueConnection(uname,pwd)` starts	A
0xA607	4	When the processing of `QueueConnectionFactory.createQueueConnection(uname,pwd)` ends	A
0xA608	1	When the processing of `TopicConnectionFactory.createTopicConnection()` starts	A
0xA609	4	When the processing of `TopicConnectionFactory.createTopicConnection()` ends	A
0xA60A	1	When the processing of `TopicConnectionFactory.createTopicConnection(uname,pwd)` starts	A
0xA60B	4	When the processing of `TopicConnectionFactory.createTopicConnection(uname,pwd)` ends	A
0xA60C	2	When the processing of `ManagedConnectionFactory.matchManagedConnections()` starts	B

Event ID	Number in the figure#	Trace collection point	Level
0xA60D	3	When the processing of `ManagedConnectionFactory.matchManagedConnections()` ends	B

Legend:

A: Standard

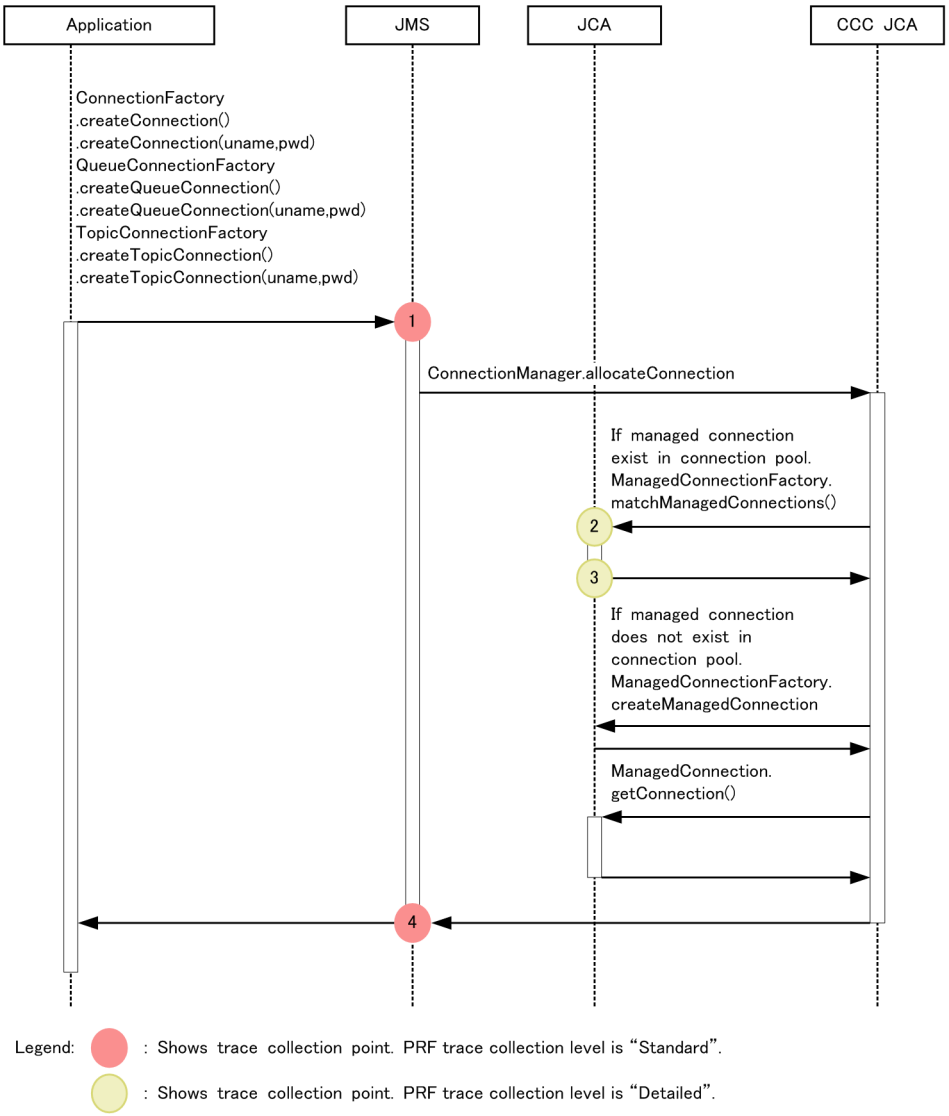
B: Advanced

#

Corresponds to the numbers in Figure 8-71.

The following figure shows the trace collection points of the `JMS ConnectionFactory` interface.

Figure 8–71: Trace collection points in JMS ConnectionFactory interface



(2) Trace information that can be collected

Trace information (interface name, operation name, and option) cannot be collected for the `JMS ConnectionFactory` interface.

8.18.2 Trace collection points of the JMS Connection interface and the trace information that can be collected

(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 8–106: Details of the trace collection points in JMS Connection interface

Event ID	Number in the figure#	Trace collection point	Level
0xA60E	1	When the processing of `Connection.createSession(transacted,acknowledgeMode)` starts	A
0xA60F	4	When the processing of `Connection.createSession(transacted,acknowledgeMode)` ends	A
0xA610	1	When the processing of `QueueConnection.createQueueSession(transacted,acknowledgeMode)` starts	A
0xA611	4	When the processing of `QueueConnection.createQueueSession(transacted,acknowledgeMode)` ends	A
0xA612	1	When the processing of `TopicConnection.createTopicSession(transacted,acknowledgeMode)` starts	A
0xA613	4	When the processing of `TopicConnection.createTopicSession(transacted,acknowledgeMode)` ends	A
0xA614	5	When the processing of `Connection.close()` starts	A
0xA615	8	When the processing of `Connection.close()` ends	A
0xA616	2	Invocation by CJMSP Broker immediately before creating the session	A
0xA617	3	Invocation by CJMSP Broker just after creating the session	A
0xA618	6	Invocation by CJMSP Broker immediately before disconnection	A
0xA619	7	Invocation by CJMSP Broker just after disconnection	A

Legend:

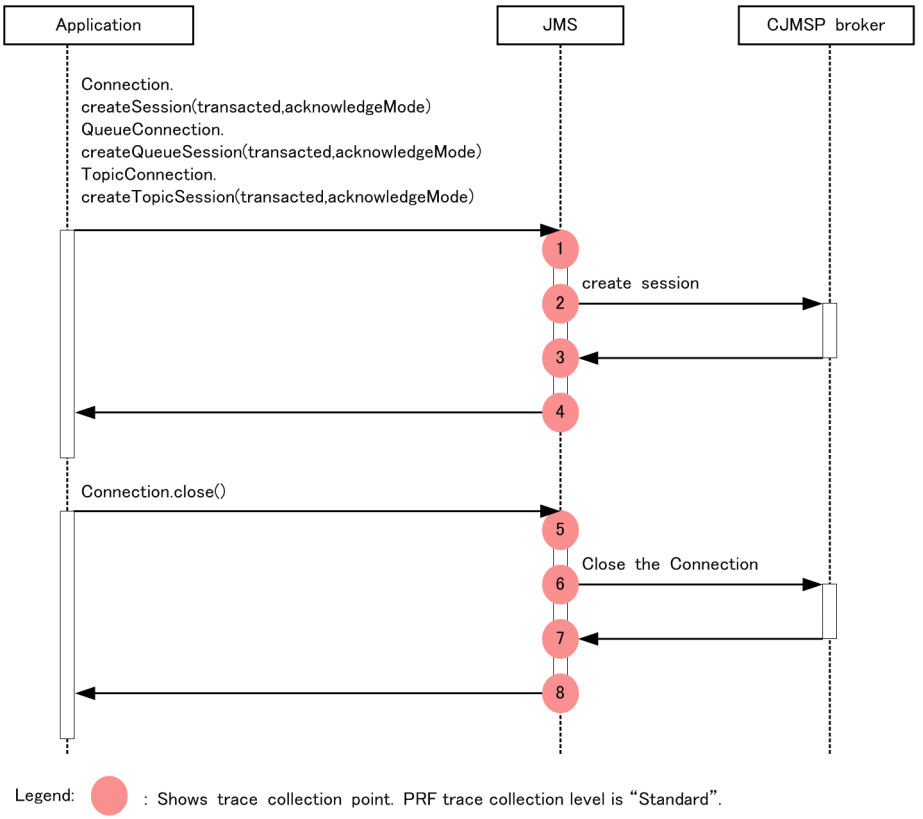
A: Standard

#

Corresponds to the numbers in Figure 8-72.

The following figure shows the trace collection points of the `JMS Connection` interface.

Figure 8–72: Trace collection points of the JMS Connection interface



(2) Trace information that can be collected

The following table describes the trace information that can be collected for the `JMS Connection` interface.

Table 8–107: Trace information that can be collected for the JMS Connection interface

Number in the figure#	Event ID	Level	Information that can be collected		
			Interface name	Operation name	Option
1	0xA60E	A	Transaction	Ack mode	--
	0xA610	A			
	0xA612	A			
2	0xA616	A	--	--	--
3	0xA617	A	--	--	--
4	0xA60F	A	--	--	--
	0xA611	A			
	0xA613	A			
5	0xA614	A	--	--	--
6	0xA618	A	--	--	--
7	0xA619	A	--	--	--
8	0xA615	A	--	--	--

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-72.

Important note

The event IDs 0xA618 and 0xA619 are also output when the Message-driven Bean applications are not running.

Reference note

The Acknowledgement mode is output using a numerical value. The numerical value is mapped to the types of Acknowledgement modes. The mapping is as follows:

- `AUTO_ACKNOWLEDGE` = 1
- `CLIENT_ACKNOWLEDGE` = 2
- `DUPS_OK_ACKNOWLEDGE` = 3
- `SESSION_TRANSACTED` = 0

The numerical values of the respective modes are defined in the JMS specifications.

8.18.3 Trace collection points of the JMS session interface and the trace information that can be collected

(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 8–108: Details of trace collection points in the JMS session interface

Event ID	Number in the figure#	Trace collection point	Level
0xA61E	5	When the processing of `Session.createSubscriber(topic)` starts	A
0xA61F	8	When the processing of `Session.createSubscriber(topic)` ends	A
0xA620	1	When the processing of `Session.createBrowser(queue)` starts	A
0xA621	4	When the processing of `Session.createBrowser(queue)` ends	A
0xA622	1	When the processing of `Session.createBrowser(queue,selector)` starts	A
0xA623	4	When the processing of `Session.createBrowser(queue,selector)` ends	A
0xA624	5	When the processing of `Session.createSubscriber(topic,selector,noLocal)` starts	A

Event ID	Number in the figure#	Trace collection point	Level
0xA625	8	When the processing of `Session.createSubscriber(topic,selector,noLocal)` ends	A
0xA626	5	When the processing of `Session.createDurableSubscriber(topic,name)` starts	A
0xA627	8	When the processing of `Session.createDurableSubscriber(topic,name)` ends	A
0xA628	5	When the processing of `Session.createDurableSubscriber(topic,name,selector,noLocal)` starts	A
0xA629	8	When the processing of `Session.createDurableSubscriber(topic,name,selector,noLocal)` ends	A
0xA62A	5	When the processing of `Session.createConsumer(destination)` starts	A
0xA62B	8	When the processing of `Session.createConsumer(destination)` ends	A
0xA62C	5	When the processing of `Session.createConsumer(destination,selector)` starts	A
0xA62D	8	When the processing of `Session.createConsumer(destination,selector)` ends	A
0xA62E	5	When the processing of `Session.createConsumer(destination,selector,noLocal)` starts	A
0xA62F	8	When the processing of `Session.createConsumer(destination,selector,noLocal)` ends	A
0xA630	5	When the processing of `Session.createReceiver(queue)` starts	A
0xA631	8	When the processing of `Session.createReceiver(queue)` ends	A
0xA632	5	When the processing of `Session.createReceiver(queue,selector)` starts	A
0xA633	8	When the processing of `Session.createReceiver(queue,selector)` ends	A
0xA634	9	When the processing of `Session.createPublisher(topic)` starts	A
0xA635	12	When the processing of `Session.createPublisher(topic)` ends	A
0xA636	9	When the processing of `Session.createProducer(destination)` starts	A
0xA637	12	When the processing of `Session.createProducer(destination)` ends	A
0xA638	9	When the processing of `Session.createSender(queue)` starts	A
0xA639	12	When the processing of `Session.createSender(queue)` ends	A
0xA63A	13	When the processing of `Session.createQueue()` starts	A
0xA63B	16	When the processing of `Session.createQueue()` ends	A

Event ID	Number in the figure#	Trace collection point	Level
0xA63C	13	When the processing of `Session.createTopic()` starts	A
0xA63D	16	When the processing of `Session.createTopic()` ends	A
0xA63E	13	When the processing of `Session.createTemporaryQueue()` starts	A
0xA63F	16	When the processing of `Session.createTemporaryQueue()` ends	A
0xA640	13	When the processing of `Session.createTemporaryTopic()` starts	A
0xA641	16	When the processing of `Session.createTemporaryTopic()` ends	A
0xA642	17	When the processing of `Session.unsubscribe()` starts	A
0xA643	20	When the processing of `Session.unsubscribe()` ends	A
0xA644	21	When the processing of `Session.commit()` starts	A
0xA645	24	When the processing of `Session.commit()` ends	A
0xA646	21	When the processing of `Session.rollback()` starts	A
0xA647	24	When the processing of `Session.rollback()` ends	A
0xA648	25	When the processing of `Session.recover()` starts	A
0xA649	26	When the processing of `Session.recover()` ends	A
0xA64A	25	When the processing of `Session.close()` starts	A
0xA64B	26	When the processing of `Session.close()` ends	A
0xA64C	2	Invocation by CJMSP Broker immediately before invoking destination authentication for the browser	A
0xA64D	3	Invocation by CJMSP Broker just after invoking destination authentication for the browser	A
0xA64E	6	Invocation by CJMSP Broker immediately before invoking consumer	A
0xA64F	7	Invocation by CJMSP Broker just after invoking consumer	A
0xA650	10	Invocation by CJMSP Broker immediately before invoking producer	A
0xA651	11	Invocation by CJMSP Broker just after invoking producer	A
0xA652	14	Invocation by CJMSP Broker immediately before invoking create destination	A
0xA653	15	Invocation by CJMSP Broker just after invoking create destination	A
0xA654	18	Invocation by CJMSP Broker immediately before invoking create persistence subscriber	A
0xA655	19	Invocation by CJMSP Broker just after invoking create persistence subscriber	A
0xA656	22	Invocation by CJMSP Broker immediately before invoking session commit	A
0xA657	23	Invocation by CJMSP Broker just after invoking session commit	A
0xA658	22	Invocation by CJMSP Broker immediately before invoking session rollback	A
0xA659	23	Invocation by CJMSP Broker just after invoking session rollback	A
0xA65A	22	Invocation by CJMSP Broker immediately before invoking session recovery	A
0xA65B	23	Invocation by CJMSP Broker just after invoking session recovery	A

Legend:

A: Standard

#

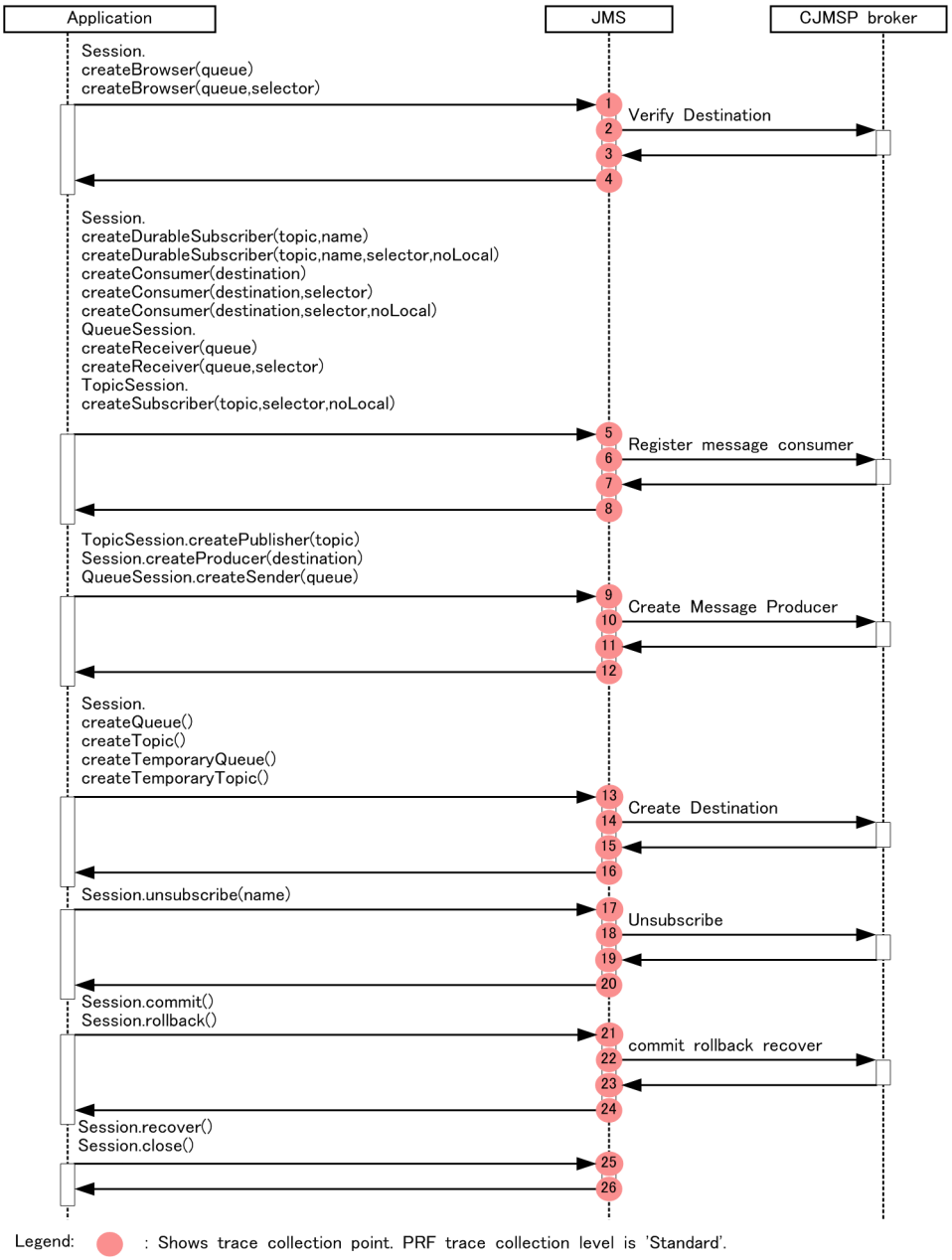
Corresponds to the numbers in Figure 8-73.

Important note

- The methods corresponding to the event IDs 0xA644 and 0xA645, 0xA646 and 0x647, `Session.commit()`, and `Session.rollback()` are not supported.
- The event IDs 0xA65A and 0xA65B are not output. The event IDs are output when `XATransaction` is used.

The following figure shows the trace collection points in the JMS session interface.

Figure 8–73: Trace collection points in the JMS session interface



Reference note

`destination` is created during the creation of producer or consumer. Therefore, the event IDs 0xa652 and 0xa653 might be invoked in the `createProducer()` and `createConsumer()` methods.

(2) Trace information that can be collected

The following table describes the trace information that can be collected for the JMS session interface.

Table 8–109: Trace information that can be collected for the JMS session interface

Number in the figure#	Event ID	Level	Information that can be collected		
			Interface name	Operation name	Option
1	0xA620	A	Queue name	--	--
	0xA622	A		Selector string	
2	0xA64C	A	--	--	--
3	0xA64D	A	--	--	--
4	0xA621	A	--	--	--
	0xA623	A			
5	0xA61E	A	Topic name	--	--
	0xA624	A		Selector string	
	0xA626	A		--	
	0xA628	A		Selector string	
	0xA62A	A	Destination name	--	
	0xA62C	A		Selector string	
	0xA62E	A			
	0xA630	A	Queue name	--	
	0xA632	A		Selector string	
6	0xA64E	A	--	--	--
7	0xA64F	A	--	--	--
8	0xA61F	A	--	--	--
	0xA625	A			
	0xA627	A			
	0xA629	A			
	0xA62B	A			
	0xA62D	A			
	0xA62F	A			
	0xA631	A			
	0xA633	A			

Number in the figure#	Event ID	Level	Information that can be collected		
			Interface name	Operation name	Option
9	0xA634	A	Topic name	--	--
	0xA636	A	Destination name	--	--
	0xA638	A	Queue name	--	--
10	0xA650	A	--	--	--
11	0xA651	A	--	--	--
12	0xA635	A	--	--	--
	0xA637	A			
	0xA639	A			
13	0xA63A	A	Queue name	--	--
	0xA63C	A	Topic name	--	--
	0xA63E	A	--	--	--
	0xA640	A			
14	0xA652	A	--	--	--
15	0xA653	A	--	--	--
16	0xA63B	A	--	--	--
	0xA63D	A			
	0xA63F	A			
	0xA641	A			
17	0xA642	A	--	--	--
18	0xA654	A	--	--	--
19	0xA655	A	--	--	--
20	0xA643	A	--	--	--
21	0xA644	A	--	--	--
	0xA646	A			
	0xA648	A			
22	0xA656	A	--	--	--
	0xA658	A			
	0xA65A	A			
23	0xA657	A	--	--	--
	0xA659	A			
	0xA65B	A			
24	0xA645	A	--	--	--
	0xA647	A			
	0xA649	A			

Number in the figure#	Event ID	Level	Information that can be collected		
			Interface name	Operation name	Option
25	0xA64A	A	--	--	--
26	0xA64B	A	--	--	--

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-73.

8.18.4 Trace collection points of the JMS messages, producer, consumer, and queue browser and the trace information that can be collected

(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 8–110: Details of the trace collection points in the JMS messages, producer, consumer, and queue browser

Event ID	Number in the figure#	Trace collection point	Level
0xA65C	1	When the processing of `MessageProducer.send(msg)` starts	A
0xA65D	4	When the processing of `MessageProducer.send(msg)` ends	A
0xA65E	1	When the processing of `MessageProducer.send(msg,deliveryMode,priority,timeToLive)` starts	A
0xA65F	4	When the processing of `MessageProducer.send(msg,deliveryMode,priority,timeToLive)` ends	A
0xA660	1	When the processing of `MessageProducer.send(destination,msg)` starts	A
0xA661	4	When the processing of `MessageProducer.send(destination,msg)` ends	A
0xA662	1	When the processing of `MessageProducer.send(destination,msg,deliveryMode,priority,timeToLive)` starts	A
0xA663	4	When the processing of `MessageProducer.send(destination,msg,deliveryMode,priority,timeToLive)` ends	A
0xA664	9	When the processing of `MessageProducer.close()` starts	A
0xA665	10	When the processing of `MessageProducer.close()` ends	A
0xA666	5	When the processing of `MessageConsumer.receive()` starts	A
0xA667	8	When the processing of `MessageConsumer.receive()` ends	A

Event ID	Number in the figure#	Trace collection point	Level
0xA61A	5	When the processing of `MessageConsumer.receive(timeout)` starts	A
0xA61B	8	When the processing of `MessageConsumer.receive(timeout)` ends	A
0xA668	5	When the processing of `MessageConsumer.receiveNoWait()` starts	A
0xA669	8	When the processing of `MessageConsumer.receiveNoWait()` ends	A
0xA66A	9	When the processing of `MessageConsumer.close()` starts	A
0xA66B	10	When the processing of `MessageConsumel.close()` ends	A
0xA66C	11	When the processing of `QueueBrowser.getEnumeration()` starts	A
0xA66D	14	When the processing of `QueueBrowser.getEnumeration()` ends	A
0xA66E	15	When the processing of `Message.acknowledge()` starts	A
0xA66F	18	When the processing of `Message.acknowledge()` ends	A
0xA670	2	Invocation by CJMSP Broker immediately before invoking the writing of the JMS messages	A
0xA671	3	Invocation by CJMSP Broker just after invoking the writing of the JMS messages	A
0xA672	6	Invocation by CJMSP Broker immediately before invoking the receiving of the JMS messages	A
0xA673	7	Invocation by CJMSP Broker just after invoking the receiving of the JMS messages	A
0xA674	12	Invocation by CJMSP Broker immediately before invoking the delivery of all the JMS messages	A
0xA675	13	Invocation by CJMSP Broker just after invoking the delivery of all the JMS messages	A
0xA676	16	Invocation by CJMSP Broker immediately before invoking the authentication of the JMS messages	A
0xA677	17	Invocation by CJMSP Broker just after invoking the authentication of the JMS messages	A

Legend:

A: Standard

#

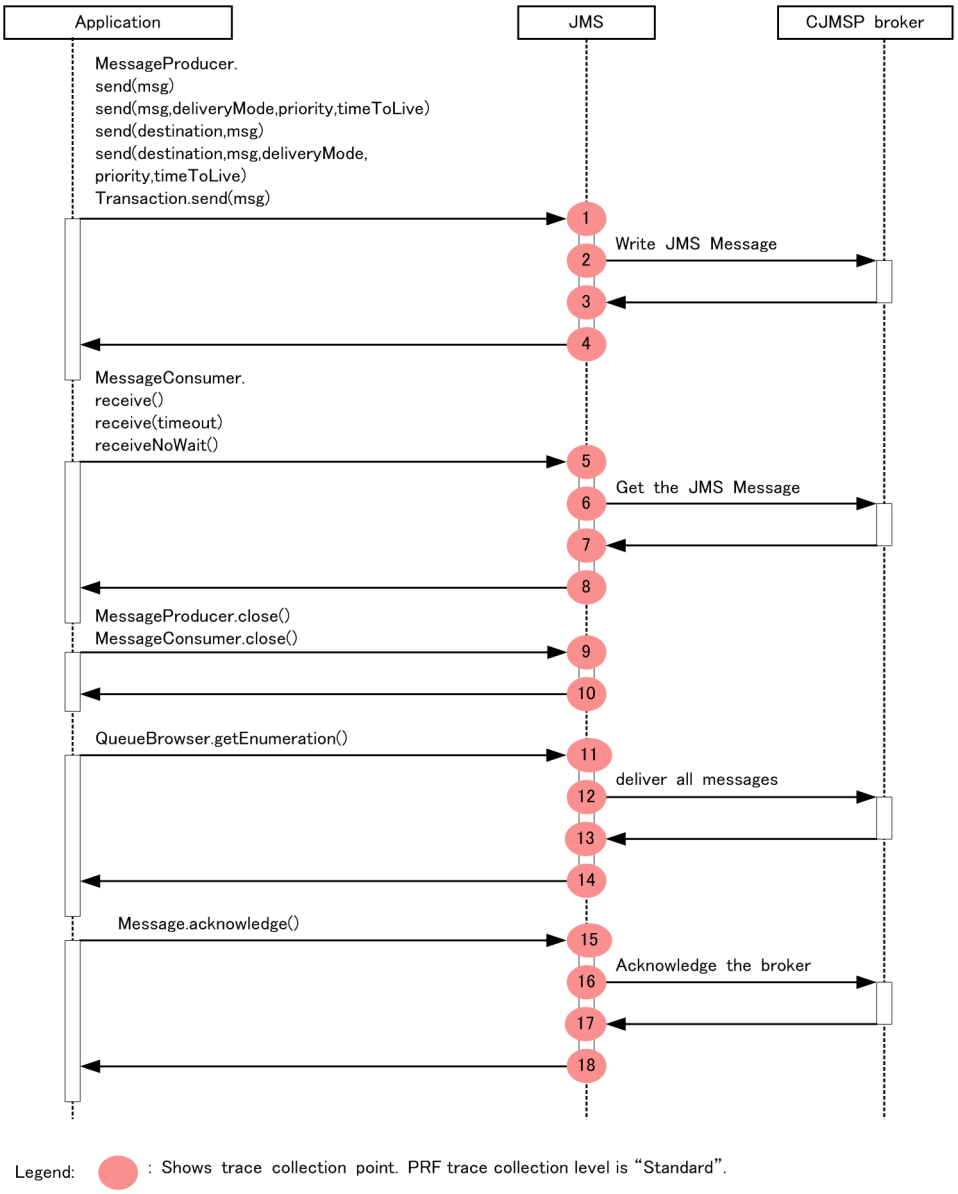
Corresponds to the numbers in Figure 8-74.

Important note

- When Transaction is used, or when the acknowledgement mode is not set to `CLIENT_ACKNOWLEDGE`, and if a message is received, the event IDs 0xA676 and 0xA677 are output.
- When a Message-driven Bean is executed, the root application information for the event IDs 0xA672 and 0xA673 becomes invalid.

The following figure shows the trace collection points in the JMS messages, producer, consumer, and queue browser.

Figure 8–74: Trace collection points of the JMS messages, producer, consumer, and queue browser



(2) Trace information that can be collected

The following table describes the trace information that can be collected for the JMS messages, producer, consumer, and queue browser.

Table 8–111: Trace information that can be collected for the JMS messages, producer, consumer, and queue browser

Number in the figure#	Event ID	Level	Information that can be collected		
			Interface name	Operation name	Option
1	0xA65C	A	Destination name	--	--
	0xA65E	A		Priority	
	0xA660	A		--	
	0xA662	A		Priority	

Number in the figure#	Event ID	Level	Information that can be collected		
			Interface name	Operation name	Option
2	0xA670	A	--	--	--
3	0xA671	A	--	--	--
4	0xA65D	A	--	--	--
	0xA65F	A			
	0xA661	A			
	0xA663	A			
5	0xA666	A	Destination name	0	--
	0xA61A	A		Timeout	
	0xA668	A		--	
6	0xA672	A	--	--	--
7	0xA673	A	--	--	--
8	0xA667	A	--	--	--
	0xA61B	A			
	0xA669	A			
9	0xA664	A	--	--	--
	0xA66A	A			
10	0xA665	A	--	--	--
	0xA66B	A			
11	0xA66C	A	--	--	--
12	0xA674	A	--	--	--
13	0xA675	A	--	--	--
14	0xA66D	A	--	--	--
15	0xA66E	A	--	--	--
16	0xA676	A	--	--	--
17	0xA677	A	--	--	--
18	0xA66F	A	--	--	--

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-74.

8.18.5 Trace collection points of CJMSP Broker when connecting to the CJMSP resource adapter and the trace information that can be collected

(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 8–112: Details of trace collection points in CJMSP Broker when connecting to the CJMSP resource adapter

Event ID	Number in the figure#	Trace collection points	Level
0xA678	1	When the processing of `ResourceAdapter.start(ctx)` starts	A
0xA679	4	When the processing of `ResourceAdapter.start(ctx)` ends	A
0xA67A	5	When the processing of `ResourceAdapter.stop()` starts	A
0xA67B	8	When the processing of `ResourceAdapter.stop()` ends	A
0xA67C	2	Invocation by CJMSP Broker immediately before invoking the `hello` method	A
0xA67D	3	Invocation by CJMSP Broker just after invoking the `hello` method	A
0xA67E	6	Invocation by CJMSP Broker immediately before invoking the `good bye` method	A
0xA67F	7	Invocation by CJMSP Broker just after invoking the `good bye` method	A

Legend:

A: Standard

#

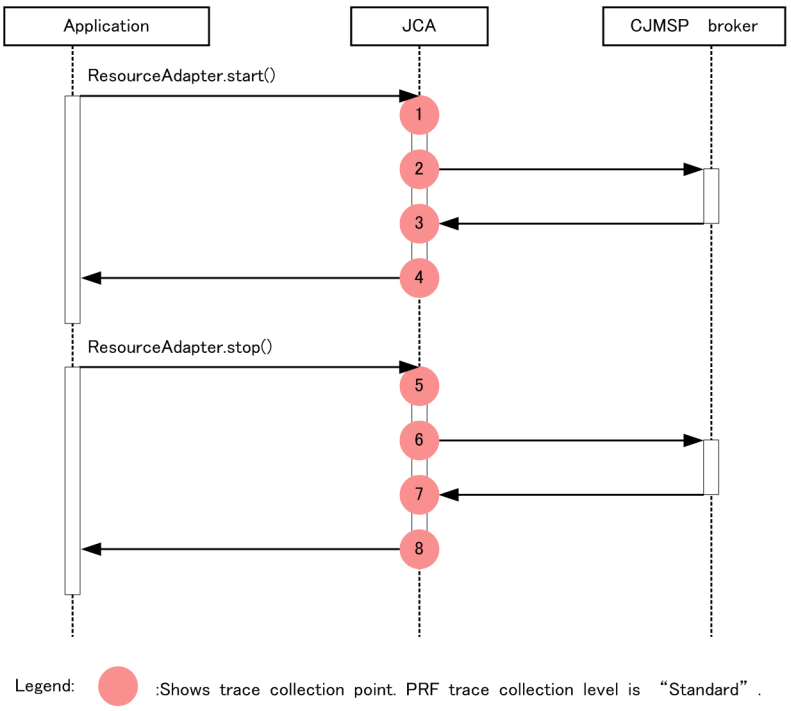
Corresponds to the numbers in Figure 8-75.

Important note

The event IDs 0xA67C and 0xA67D are also output when CJMSP Broker and the CJMSP resource adapter are communicating in order to establish the connection.

The following figure shows the trace collection points in CJMSP Broker when connecting to the CJMSP resource adapter.

Figure 8–75: Trace collection points of CJMSP Broker when connecting to the CJMSP resource adapter



(2) Trace information that can be collected

Trace information (interface name, operation name, and option) cannot be collected for CJMSP Broker when connecting to the CJMSP resource adapter.

8.18.6 Trace collection points of the transaction management in the CJMSP resource adapter and trace information that can be collected

(1) Transaction management in the CJMSP resource adapter (for LocalTransaction)

(a) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–113: Details of the trace collection points of the transaction management (for LocalTransaction) in the CJMSP resource adapter

Event ID	Number in the figure#	Trace collection point	Level
0xA656	6	Invocation by CJMSP Broker immediately before invoking session commit	A
0xA657	7	Invocation by CJMSP Broker just after invoking session commit	A
0xA658	6	Invocation by CJMSP Broker immediately before invoking session rollback	A

Event ID	Number in the figure#	Trace collection point	Level
0xA659	7	Invocation by CJMSP Broker just after invoking session rollback	A
0xA686	1	When the processing of `LocalTransaction.begin()` starts	A
0xA687	4	When the processing of `LocalTransaction.begin()` ends	A
0xA688	5	When the processing of `LocalTransaction.commit()` starts	A
0xA689	8	When the processing of `LocalTransaction.commit()` ends	A
0xA68A	5	When the processing of `LocalTransaction.rollback()` starts	A
0xA68B	8	When the processing of `LocalTransaction.rollback()` ends	A
0xA68C	2	Invocation by CJMSP Broker immediately before the transaction starts	A
0xA68D	3	Invocation by CJMSP Broker just after the transaction starts	A

Legend:

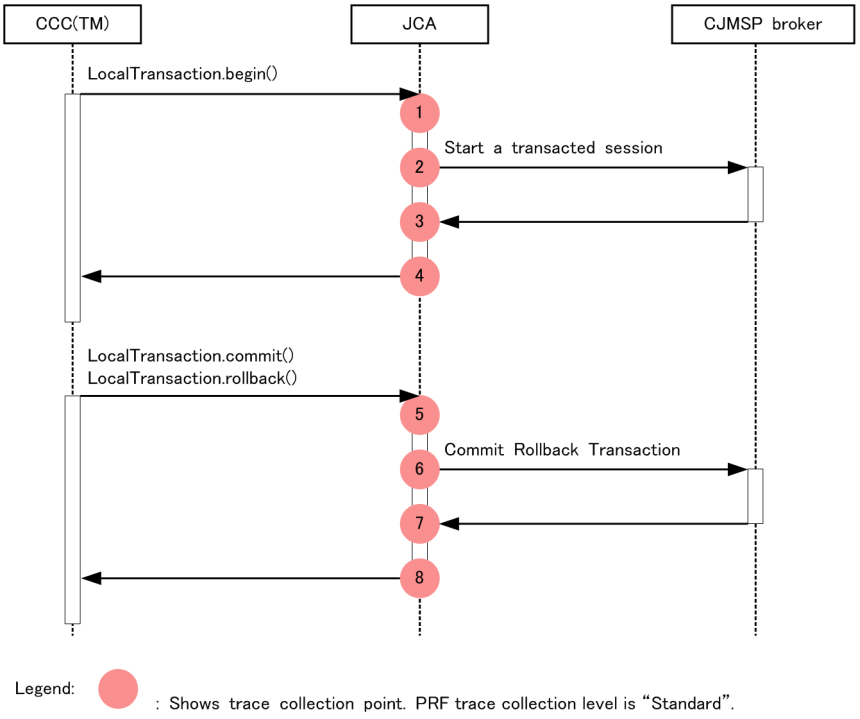
A: Standard

#

Corresponds to the numbers in Figure 8-76.

The following figure shows the trace collection points of the transaction management in the CJMSP resource adapter (for `LocalTransaction`).

Figure 8–76: Trace collection points of the transaction management in the CJMSP resource adapter (for LocalTransaction)



(b) Trace information that can be collected

Trace information (interface name, operation name, and option) cannot be collected for the transaction management in the CJMSP resource adapter (for `LocalTransaction`).

(2) Transaction management in the CJMSP resource adapter (for XAResource)

(a) Trace collection points and PRF trace collection levels

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–114: Details of the trace collection points of the transaction management in the CJMSP resource adapter (for XAResource)

Event ID	Number in the figure#	Trace collection points	Level
0xA656	2	Invocation by CJMSP Broker immediately before invoking session commit	A
0xA657	3	Invocation by CJMSP Broker just after invoking session commit	A
0xA658	2	Invocation by CJMSP Broker immediately before invoking session rollback	A
0xA659	3	Invocation by CJMSP Broker just after invoking session rollback	A
0xA68C	2	Invocation by CJMSP Broker immediately before the transaction starts	A
0xA68D	3	Invocation by CJMSP Broker just after the transaction starts	A
0xA692	1	When the processing of `XAResource.start()` starts	A
0xA693	4	When the processing of `XAResource.start()` ends	A
0xA694	1	When the processing of `XAResource.end()` starts	A
0xA695	4	When the processing of `XAResource.end()` ends	A
0xA696	1	When the processing of `XAResource.prepare()` starts	A
0xA697	4	When the processing of `XAResource.prepare()` ends	A
0xA698	1	When the processing of `XAResource.commit()` starts	A
0xA699	4	When the processing of `XAResource.commit()` ends	A
0xA69A	1	When the processing of `XAResource.rollback()` starts	A
0xA69B	4	When the processing of `XAResource.rollback()` ends	A
0xA69E	2	Invocation by Broker immediately before invoking stop `XAResource`	A
0xA69F	3	Invocation by Broker just after invoking stop `XAResource`	A
0xA6A0	2	Invocation by Broker immediately before preparing `XAResource`	A
0xA6A1	3	Invocation by Broker just after preparing `XAResource`	A

Legend:

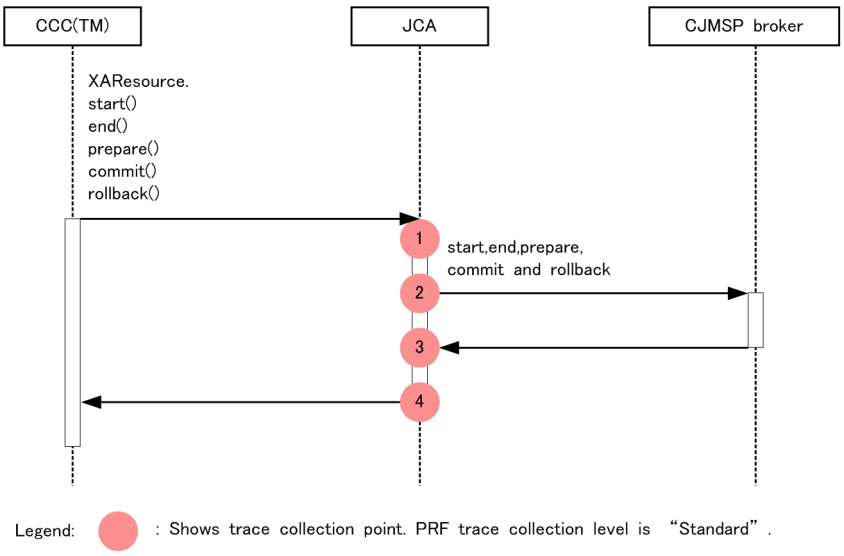
A: Standard

#

Corresponds to the numbers in Figure 8-77.

The following figure shows the trace collection points of the transaction management in the CJMSP resource adapter (for XAResource).

Figure 8–77: Trace collection points of the transaction management in the CJMSP resource adapter (for XAResource)



(b) Trace information that can be collected

The following table describes the trace information that can be collected for the transaction management in the CJMSP resource adapter (for XAResource).

Table 8–115: Trace information that can be collected for the transaction management in the CJMSP resource adapter (for XAResource)

Number in the figure#	Event ID	Level	Information that can be collected		
			Interface name	Operation name	Option
1	0xA692	A	Flag	--	--
	0xA694	A	Flag		
	0xA696	A	--		
	0xA698	A	Flag		
	0xA69A	A	--		
2	0xA656	A	--	--	--
	0xA658	A			
	0xA68C	A			
	0xA69E	A			
	0xA6A0	A			
3	0xA657	A	--	--	--
	0xA659	A			
	0xA68D	A			
	0xA69F	A			
	0xA6A1	A			
4	0xA693	A	--	--	--

Number in the figure#	Event ID	Level	Information that can be collected		
			Interface name	Operation name	Option
	0xA695	A			
	0xA697	A			
	0xA699	A			
	0xA69B	A			

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-77.

8.18.7 Trace collection points when Message-driven Bean is deployed from the CJMSP resource adapter and the trace information that can be collected

(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 8–116: Details of trace collection points when Message-driven Bean is deployed from the CJMSP resource adapter

Event ID	Number in the figure#	Trace collection point	Level
0xA6A6	1	When the processing of `ResourceAdapter.endpointActivation(endpointFactory,spec)` starts	A
0xA6A7	2	When the processing of `ResourceAdapter.endpointActivation(endpointFactory,spec)` ends	A
0xA6A8	3	When the processing of `ResourceAdapter.endpointDeactivation(endpointFactory,spec)` starts	A
0xA6A9	4	When the processing of `ResourceAdapter.endpointDeactivation(endpointFactory,spec)` ends	A
0xA6AA	5	When the processing of `MessageListener.onMessage()` starts	A
0xA6AB	6	When the processing of `MessageListener.onMessage()` ends	A

Legend:

A: Standard

#:

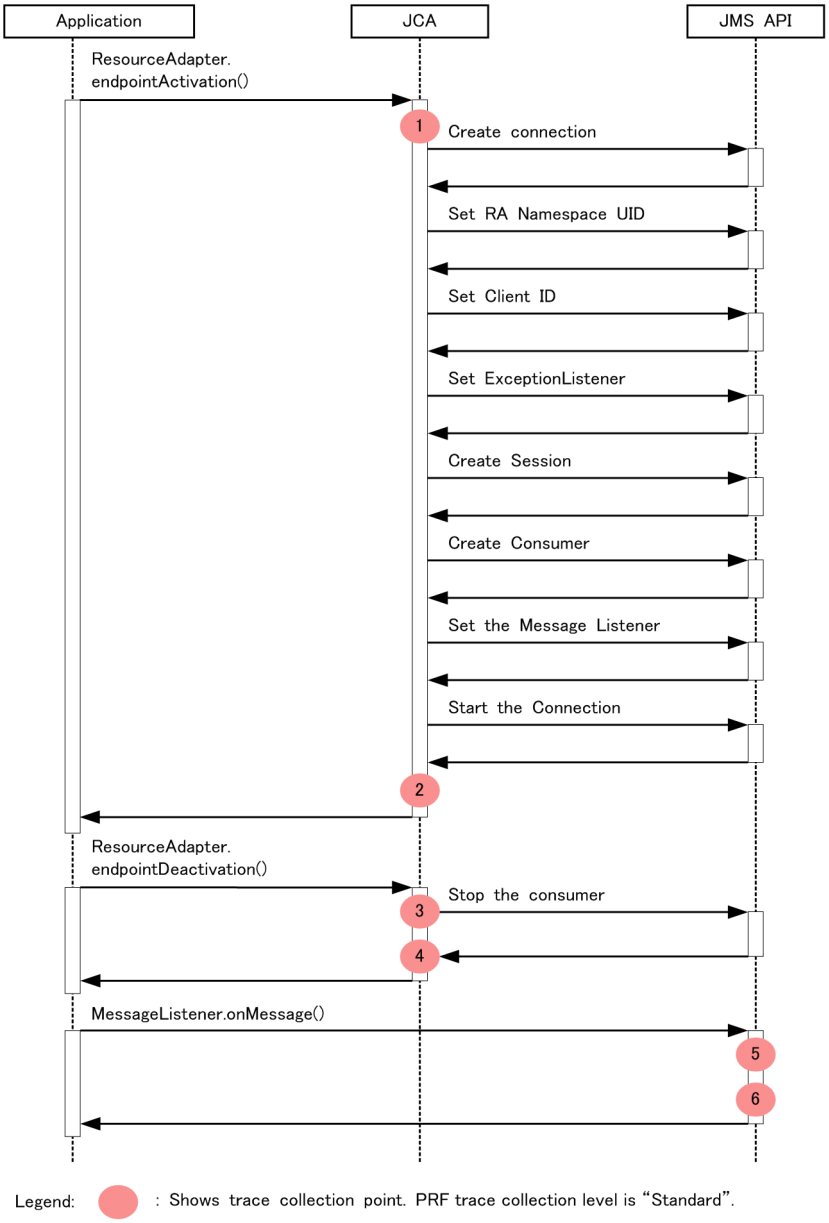
Corresponds to the numbers in Figure 8-78.

Important note

- 0xA67E and 0xA67F are output when you invoke `endpointDeactivation()`.
- 0xA67C are 0xA67D are output when you invoke `endpointActivation()`.

The following figure shows the trace collection points when Message-driven Bean is deployed from the CJMSP resource adapter.

Figure 8–78: Trace collection points when Message-driven Bean is deployed from the CJMSP resource adapter



(2) Trace information that can be collected

Trace information (interface name, operation name, and option) cannot be collected when Message-driven Bean is deployed from the CJMSP resource adapter.

8.19 Trace collection points of JavaMail

This section describes the trace collection points of JavaMail and the trace information that you can collect.

8.19.1 Trace collection points of JavaMail transmission and the trace information that you can collect

(1) Trace collection points and PRF trace collection levels

The following table describes the information such as Event ID, trace collection points, and PRF trace collection level.

Table 8–117: Details of trace collection points on JavaMail transmission

Event ID	Numbers used in the figures[#1]	Trace collection point	Level
0xAD00	1	Entry point of the `connect(String host, int port, String user, String password)` method of the `javax.mail.Transport` class	A
0xAD01	22	Exit point of the `connect(String host, int port, String user, String password)` method of the `javax.mail.Transport` class	A
0xAD02	23	Entry point of the `sendMessage(Message message, Address[] addresses)` method of the `javax.mail.Transport` class	A
0xAD03	38	Exit point of the `sendMessage(Message message, Address[] addresses)` method of the `javax.mail.Transport` class	A
0xAD04	39	Entry point of the `close` method of the `javax.mail.Transport` class	A
0xAD05	44	Exit point of the `close` method of the `javax.mail.Transport` class	A
0xAD06	2	Immediately before starting the process to fetch the connection	A
0xAD07	3	Immediately after ending the process to fetch the connection	A
0xAD08	26[#2]	Immediately before starting to send the entire recipient information	A
0xAD09	29[#2]	Immediately after sending the entire recipient information	A
0xAD0A	34[#3]	Immediately before starting to send the mail	A
0xAD0B	35[#3]	Immediately after sending the mail	A
0xAD0C	42	Immediately before starting the process to terminate the connection	B
	6	Immediately before starting the process to terminate the connection	B
0xAD0D	43	Immediately after ending the process to terminate the connection	B
	7	Immediately after ending the process to terminate the connection	B
0xAD0E	45	Entry point of the `send(Message msg, Address[] addresses)` method and `send(Message msg)` method of the `javax.mail.Transport` class	A

Event ID	Numbers used in the figures[#1]	Trace collection point	Level
0xAD0F	46	Exit point of the `send(Message msg, Address[] addresses)` method and `send(Message msg)` method of the `javax.mail.Transport` class	A
0xAD10	8[#4]	Immediately before issuing the `EHLO` or `HELO` command	A
0xAD11	9[#4]	Immediately after receiving a response to the `EHELO` or `HELO` command	A
0xAD12	10	Immediately before issuing the `AUTH` command	B
	14	Immediately before starting to send the user name and password	B
	16	Immediately before starting to notify the end of authentication	B
	18	Immediately before starting to send the user name	B
	20	Immediately before starting to send the password	B
	24	Immediately before starting to issue the `MAIL` command	B
	27[#5]	Immediately before starting to issue the `RCPT` command	B
	30	Immediately before starting to issue the `RSET` command	B
	32	Immediately before starting to issue the `DATA` command	B
	36	Immediately before starting to issue the notification for the end of mail text transmission	B
	40	Immediately before starting to issue the `QUIT` command	B
	47[#6]	Immediately after receiving a response to the `NOOP` command	B
	12	Immediately after starting to issue the `STARTTLS` command	B
0xAD13	11	Immediately after receiving a response to `AUTH`	B
	15	Immediately after receiving a response to the sent user name and password	B
	17	Immediately after notifying the end of authentication	B
	19	Immediately after receiving a response to the sent user name	B
	21	Immediately after receiving a response to the sent password	B
	25	Immediately after receiving a response to the `MAIL` command	B
	28[#5]	Immediately after receiving a response to the `RCPT` command	B
	31	Immediately after receiving a response to the `RSET` command	B
	33	Immediately after receiving a response to the `DATA` command	B
	37	Immediately after receiving a response to the notification for the end of mail text transmission	B
	41	Immediately after receiving a response to the `QUIT` command	B
	48[#6]	Immediately before starting to issue the `NOOP` command	B
	13	Immediately after receiving a response to the `STARTTLS` command	B
0xAD14	4	Immediately before receiving the server response on connection	B
0xAD15	5	Immediately after receiving the server response on connection	B

Legend:

A: Standard

B: Advanced

#1

Corresponds to the numbers used in Figure 8-79, Figure 8-80, or Figure 8-81.

#2

Fetched at the start and end of sending the entire recipient information.

#3

The data to be sent is read and sent to the mail server in the data transfer described in the Trace collection point column of point 32 or 33. For example, if a file is attached, the file is read and the data of the file is sent to the mail server.

#4

Sometimes 1 log each for `EHLO` and `HELO` is output, because if `EHLO` fails, the reconnection is tried by issuing `HELO`.

#5

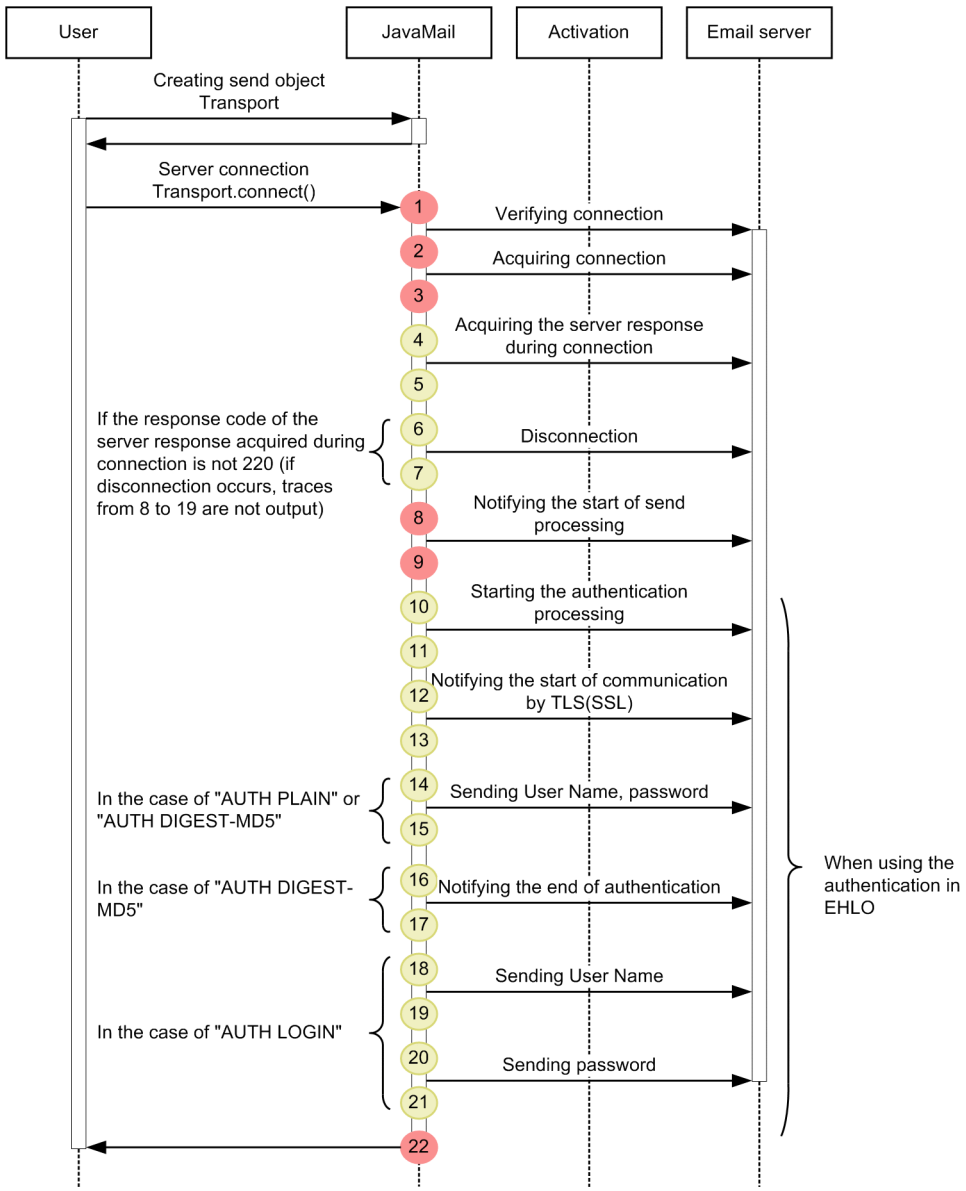
Fetched at the start and end of sending the individual recipient information.

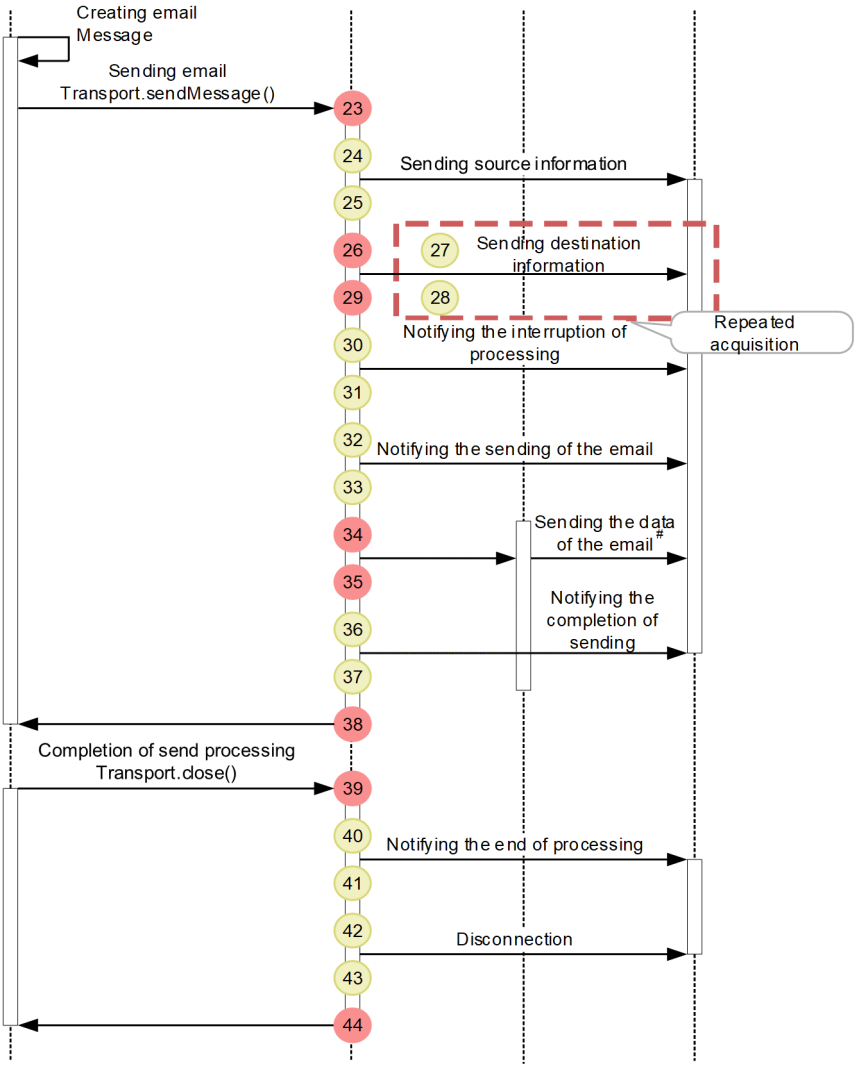
#6

Fetched only when connected with the server.

The following figure shows the trace collection points on JavaMail transmission.

Figure 8–79: Trace collection points on JavaMail transmission (when the sendMessage(Message message, Address[] addresses) method of the javax.mail.Transport class is used)



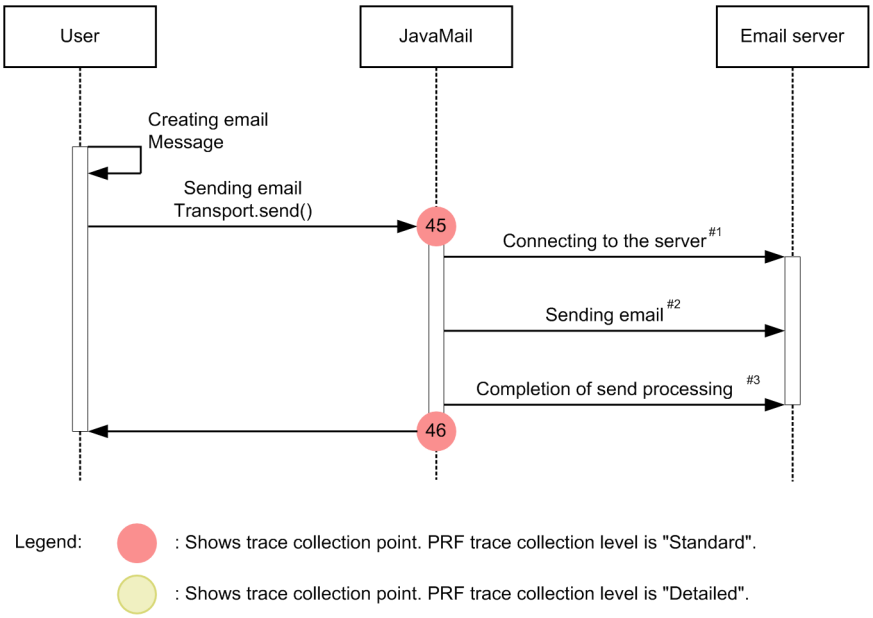


Legend: : Shows trace collection point. PRF trace collection level is "Standard".

: Shows trace collection point. PRF trace collection level is "Detailed".

#: When using an attachment, activation performs the data sending processing.

Figure 8–80: Trace collection points on JavaMail transmission (when the send(Message msg, Address[] addresses) method and send(Message msg) method of the javax.mail.Transport class are used)

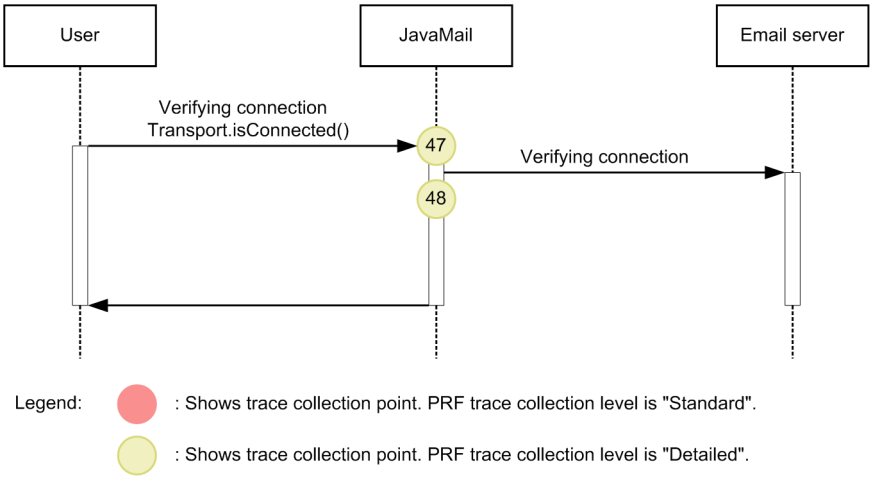


#1: The `connect` method of the `javax.mail.Transport` class is invoked internally, so the collection points from 1 to 20, shown in Figure 8-79, are also output.

#2: The `sendMessage` method of the `javax.mail.Transport` class is invoked internally, so the collection points from 21 to 36, shown in Figure 8-79, are also output.

#3: The `close` method of the `javax.mail.Transport` class is invoked internally, so the collection points from 37 to 42, shown in Figure 8-79, are also output.

Figure 8–81: Trace collection point on confirming the JavaMail connection



(2) Trace information that you can collect

The following table describes the trace information that you can collect on JavaMail transmission.

Table 8–118: Trace information that you can collect on JavaMail transmission

Numbers used in the figures[#1]	Event ID	Level	Interface name	Operation name	Option
1	0xAD00	A	--	--	--
2	0xAD06	A	--	--	#2
3	0xAD07	A	--	--	Exception class name in the event of an abnormality
4	0xAD14	B	String that shows communication[#3]	--	--
5	0xAD15	B	String that shows communication[#3]	Response code from the mail server[#4, #5, #6]	Exception class name in the event of an abnormality[#7]
6	0xAD0C	B	--	--	--
7	0xAD0D	B	--	--	Exception class name in the event of an abnormality
8	0xAD10	A	String that shows communication[#3]	--	--
9	0xAD11	A	String that shows communication[#3]	Response code from the mail server[#4, #6]	Exception class name in the event of an abnormality[#7]
10	0xAD12	B	String that shows communication[#3]	--	--
11	0xAD13	B	String that shows communication[#3]	Response code from the mail server[#4, #5, #6]	Exception class name in the event of an abnormality[#7]
12	0xAD12	B	String that shows communication [#3]	--	--
13	0xAD13	B	String that shows communication [#3]	Response code from the mail server [#4, #5, #6]	Name of the exception class if an error occurs [#7]
14	0xAD12	B	String that shows communication[#3]	--	--
15	0xAD13	B	String that shows communication[#3]	Response code from the mail server[#4, #5, #6]	Exception class name in the event of an abnormality[#7]
16	0xAD12	B	String that shows communication[#3]	--	--
17	0xAD13	B	String that shows communication[#3]	Response code from the mail server[#4, #5, #6]	Exception class name in the event of an abnormality[#7]
18	0xAD12	B	String that shows communication[#3]	--	--
19	0xAD13	B	String that shows communication[#3]	Response code from the mail server[#4, #5, #6]	Exception class name in the event of an abnormality[#7]
20	0xAD12	B	String that shows communication[#3]	--	--
21	0xAD13	B	String that shows communication[#3]	Response code from the mail server[#4, #5, #6]	Exception class name in the event of an abnormality[#7]

Numbers used in the figures[#1]	Event ID	Level	Interface name	Operation name	Option
22	0xAD01	A	--	--	Exception class name in the event of an abnormality
23	0xAD02	A	--	--	--
24	0xAD12	B	String that shows communication[#3]	--	--
25	0xAD13	B	String that shows communication[#3]	Response code from the mail server[#4, #5, #6]	Exception class name in the event of an abnormality[#7]
26	0xAD08	A	--	--	--
27	0xAD12	B	String that shows communication[#3]	--	--
28	0xAD13	B	String that shows communication[#3]	Response code from the mail server[#4, #5, #6]	Exception class name in the event of an abnormality[#7]
29	0xAD09	A	--	--	Exception class name in the event of an abnormality
30	0xAD12	B	String that shows communication[#3]	--	--
31	0xAD13	B	String that shows communication[#3]	Response code from the mail server[#4, #5, #6]	Exception class name in the event of an abnormality[#7]
31	0xAD12	B	String that shows communication[#3]	--	--
32	0xAD13	B	String that shows communication[#3]	Response code from the mail server[#4, #5, #6]	Exception class name in the event of an abnormality[#7]
33	0xAD0A	A	--	--	--
34	0xAD0B	A	--	--	Exception class name in the event of an abnormality
35	0xAD12	B	String that shows communication[#3]	--	--
36	0xAD13	B	String that shows communication[#3]	Response code from the mail server[#4, #5, #6]	Exception class name in the event of an abnormality[#7]
37	0xAD03	A	--	--	Exception class name in the event of an abnormality
38	0xAD04	A	--	--	--
39	0xAD12	B	String that shows communication[#3]	--	--
40	0xAD13	B	String that shows communication[#3]	Response code from the mail server[#4, #5, #6]	Exception class name in the event of an abnormality[#7]
41	0xAD0C	B	--	--	--
42	0xAD0D	B	-	-	Exception class name in the event of an abnormality
43	0xAD05	A	--	--	Exception class name in the event of an abnormality

Numbers used in the figures[#1]	Event ID	Level	Interface name	Operation name	Option
44	0xAD0E	A	Method name (`send(Message, Address[])` or `send(Message)`)	--	--
45	0xAD0F	A	Method name (`send(Message, Address[])` or `send(Message)`)	--	Exception class name in the event of an abnormality
46	0xAD12	B	String that shows communication[#3]	--	--
47	0xAD13	B	String that shows communication[#3]	Response code from the mail server[#4, #5, #6]	Exception class name in the event of an abnormality[#7]

Legend:

A: Standard

B: Advanced

--: Not applicable

#1

Corresponds to the numbers used in Figure 8-79, Figure 8-80, or Figure 8-81.

#2

Outputs the following items:

host-name port-number timeout-value-when-fetching-connection communication-timeout-value

Each of the items must be output by separating them with a single byte space.

`0` is output if the timeout is set to infinite. `-1` is output, if the user omits the timeout setting.

(Example) `localhost 25 10000 10000`

#3

The following table describes the output contents of the strings that show communication:

Sr. No.	String that shows communication	Communication process
1	-- (Not applicable)	Fetch connection
2	connect-mail-server	Fetch server response on connection
3	`EHLO` command arguments	Issue `EHLO` command
4	`HELO` command arguments	Issue `HELO` command
5	AUTH LOGIN	Issue `AUTH` command (LOGIN as argument)
6	AUTH PLAIN	Issue `AUTH` command (PLAN as argument)
7	AUTH DIGEST-MD5	Issue `AUTH` command (DIGEST-MD5 as argument)
8	SEND USER	Send user name
9	SEND PASS	Send password
10	SEND USER PASS	Send user name and password
11	AUTH END	Notify authentication end

Sr. No.	String that shows communication	Communication process
12	QUIT	Issue `QUIT` command
13	`MAIL` command arguments	Issue `MAIL` command
14	`RCPT` command arguments	Issue `RCPT` command
15	`RSET`	Issue `RSET` command
16	`NOOP`	Issue `NOOP` command
17	`DATA`	Issue `DATA` command
18	SEND MAIL	Send mail body data
19	.	Notifying the end of the mail text transmission
20	`STARTTLS`	Issuing the `STARTTLS` command

#4

The following table describes the output contents of responses:

Response	Output contents
Response code is correct (satisfies the RFC specifications)	Response code
Response code is incorrect (does not satisfy the RFC specifications)	Header 4 letters of the 1st line of the response (See KDJE59111-E message for details) • If the 1st line is less than 4 letters Output all the letters that exist in the 1st line • No response Output empty string

#5

In the `QUIT` command, response code is output only if `mail.smtp.quitwait` or `mail.smtps.quitwait` is `true`.

#6

Response code is output only if the response is received. Response code is not output, if the response is not received due to the abnormal conditions such as occurrence of IOException in communication.

#7

If the response code does not satisfy the RFC specifications and if EOF is detected, the exception class is not output.

8.19.2 Trace collection points on JavaMail receipt and the trace information that you can collect

(1) Trace collection points and PRF trace collection levels

The following table describes the information such as Event ID, trace collection points, and PRF trace collection level.

Table 8–119: Details of trace collection points on JavaMail receipt

Event ID	Numbers used in the figures[#1]	Trace collection points	Level
0xAD80	1	Entry point of the `connect(String host, int port, String user, String password)` method of the `javax.mail.Store` class	A

Event ID	Numbers used in the figures[#1]	Trace collection points	Level
0xAD81	14	Exit point of the `connect(String host, int port, String user, String password)` method of the `javax.mail.Store` class	A
0xAD82	2	Immediately before starting the process to fetch the connection	A
0xAD83	3	Immediately after ending the process to fetch the connection	A
0xAD84	15	Entry point of the `open(int)` method of the `javax.mail.Folder` class	A
0xAD85	18	Exit point of the `open(int)` method of the `javax.mail.Folder` class	A
0xAD86	25	Entry point of the `close(boolean)` method of the `javax.mail.Folder` class	A
0xAD87	36	Exit point of the `close(boolean)` method of the `javax.mail.Folder` class	A
0xAD88	34	Immediately before starting the process to terminate the connection	B
	6	Immediately before starting the process to terminate the connection	B
0xAD89	35	Immediately after ending the process to terminate the connection	B
	7	Immediately after ending the process to terminate the connection	B
0xAD8A	19[#2]	Immediately before starting the process to fetch the entire message information	A
0xAD8B	22[#2]	Immediately after ending the process to fetch the entire message information.	A
0xAD8C	28[#2]	Immediately before issuing the `DELE` command for all the messages to be deleted	A
0xAD8D	31[#2]	Immediately after receiving the response to the `DELE` command for all the messages to be deleted	A
0xAD8E	10	Immediately before starting to issue the `USER` command	A
	23[#3]	Immediately before starting to issue the `LIST`, `UIDL`, `RETR`, or `TOP` commands	A
	26	Immediately before starting to issue the RSET command	A
	8	Immediately before starting to issue the `CAPA` command	A
0xAD8F	11	Immediately after receiving the response to the `USER` command	A
	24[#3]	Immediately after receiving the response to the `LIST`, UIDL, RETR, or TOP command	A
	27	Immediately after receiving the response to the `RSET` command	A
	9	Immediately after receiving the response to the `CAPA` command	A
0xAD90	12	Immediately before starting to issue the `PASS` command	B
	16	Immediately before starting to issue the `STAT` command	B
	20[#4, #5]	Immediately before starting to issue the `TOP` or `LIST` command	B
	29[#5]	Immediately before starting to issue the `DELE` command	B
	32	Immediately before starting to issue the `QUIT` command	B

Event ID	Numbers used in the figures[#1]	Trace collection points	Level
	37[#6]	Immediately before starting to issue the NOOP command	B
0xAD91	13	Immediately after receiving the response to the PASS command	B
	17	Immediately after receiving the response to the STAT command	B
	21[#4, #5]	Immediately after receiving the response to the TOP command	B
	30[#5]	Immediately after receiving the response to the DELE command	B
	33	Immediately after receiving the response to the QUIT command	B
	38[#6]	Immediately after receiving the response to the NOOP command	B
0xAD92	4	Immediately before receiving the server response on connection	B
0xAD93	5	Immediately after receiving the server response on connection	B

Legend:

A: Standard

B: Advanced

#1

Corresponds to the numbers used in Figure 8-82 or Figure 8-83.

#2

Fetched at the start and end of receiving the entire mail information.

#3

If not called by specifying ENVELOPE in the 2nd argument of the fetch method of the javax.mail.Folder class.

#4

Only if called by specifying ENVELOPE in the 2nd argument of the fetch method of the javax.mail.Folder class.

#5

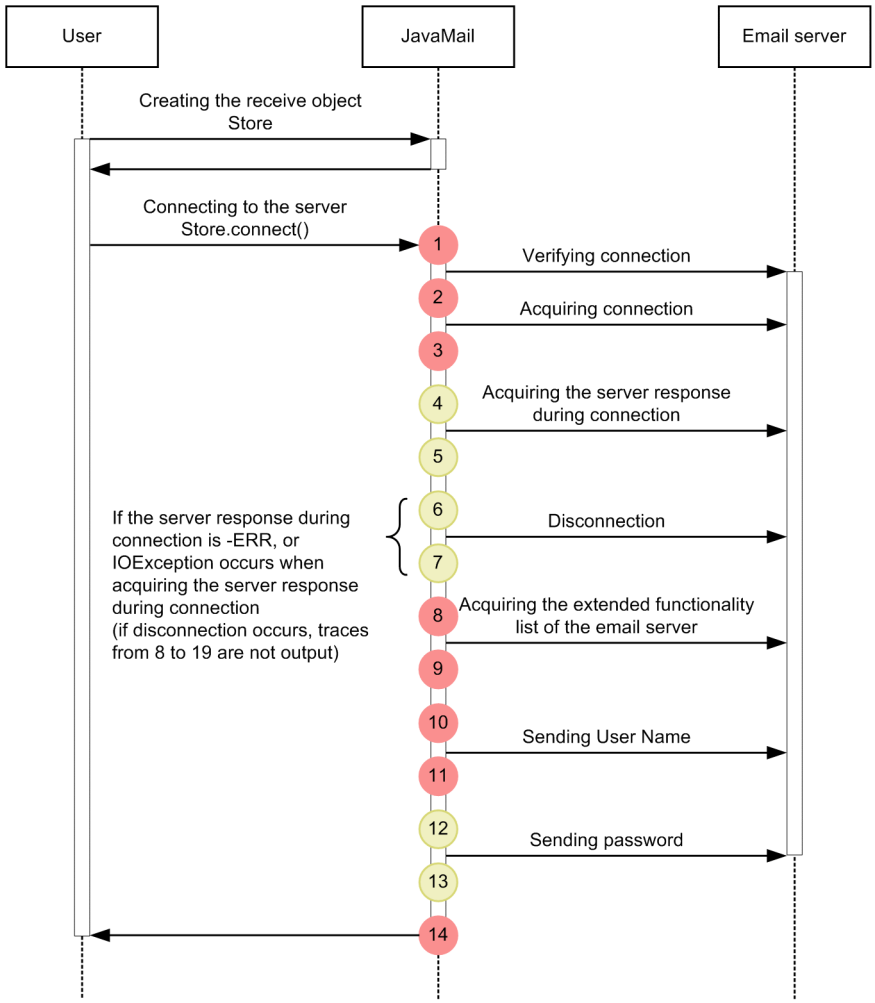
Fetched at the start and end of receiving the individual mail information.

#6

Collected only when connected to the server.

The following figure shows the trace collection points on JavaMail receipt.

Figure 8–82: Trace collection points on JavaMail receipt



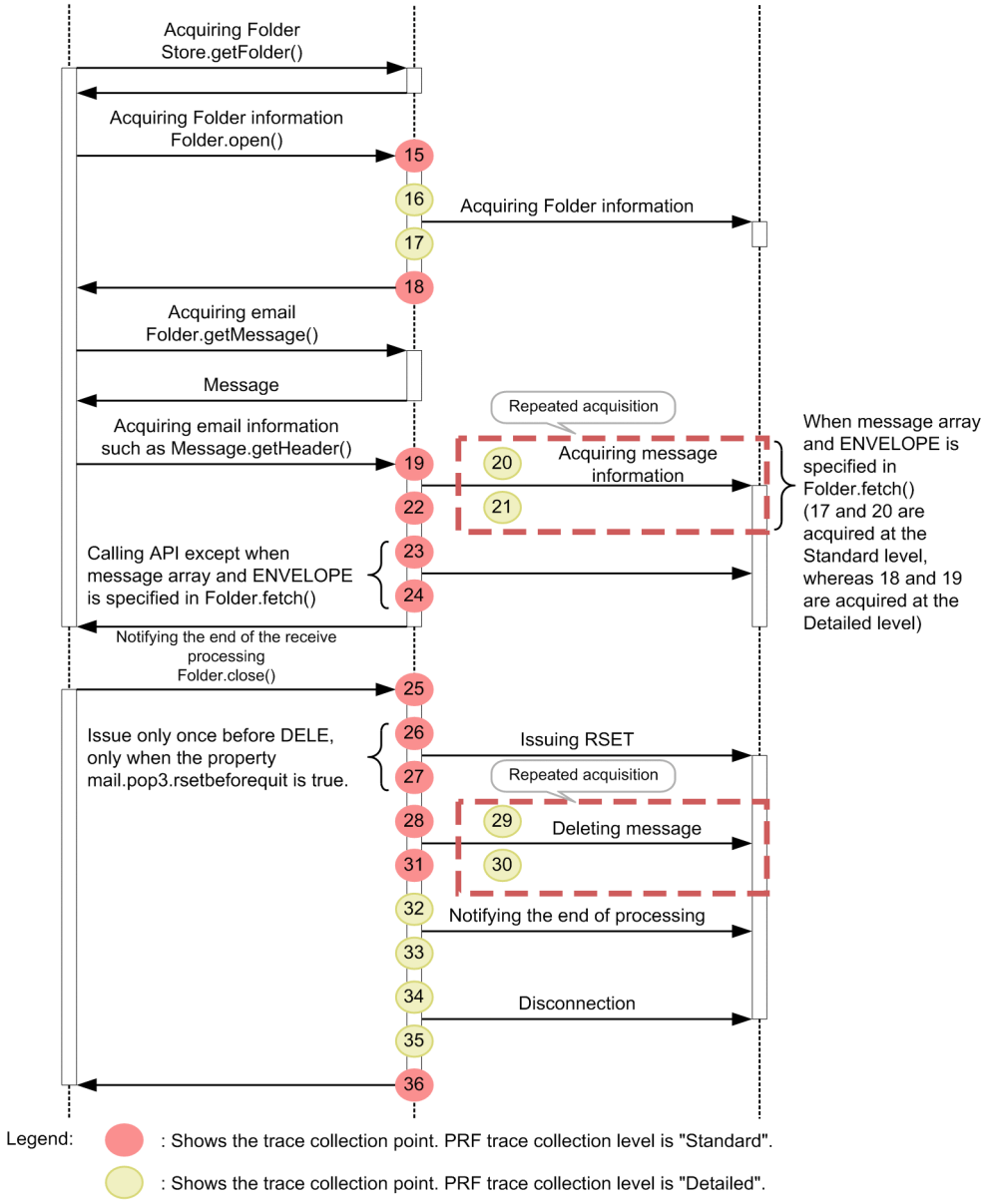
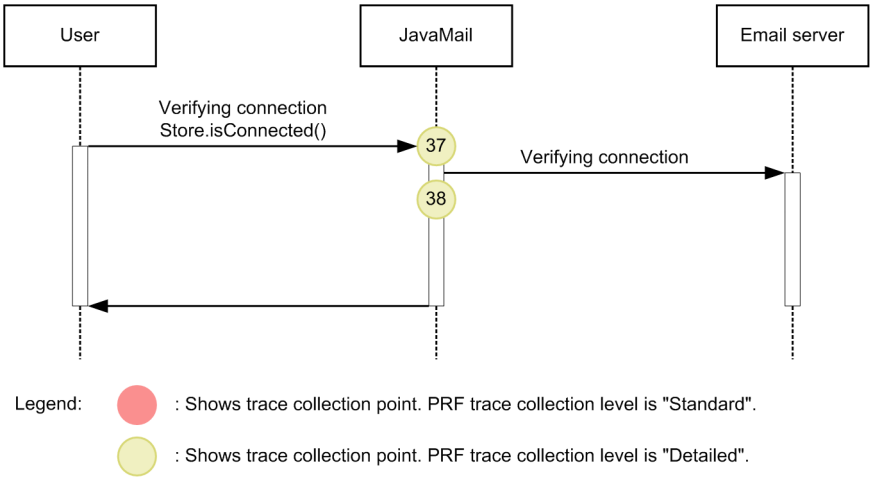


Figure 8–83: Trace collection point on confirming the JavaMail connection



(2) Trace information that you can collect

The following table describes the trace information that you can collect on JavaMail receipt.

Table 8–120: Trace information that you can collect on JavaMail receipt

Numbers used in the figures[#1]	Event ID	Level	Interface name	Operation name	Option
1	0xAD80	A	--	--	--
2	0xAD82	A	--	--	#2
3	0xAD83	A	--	--	Exception class name in the event of an abnormality
4	0xAD92	B	String that shows communication[#3]	--	--
5	0xAD93	B	String that shows communication[#3]	Response code from the mail server[#4]	Exception class name in the event of an abnormality
6	0xAD88	B	--	--	--
7	0xAD89	B	--	--	Exception class name in the event of an abnormality
8	0xAD8E	A	String that shows communication [#3]	--	--
9	0xAD8F	A	String that shows communication [#3]	Response code from the mail server [#4]	Name of the exception class if an error occurs
10	0xAD8E	A	String that shows communication[#3]	--	--
11	0xAD8F	A	String that shows communication[#3]	Response code from the mail server[#4]	Exception class name in the event of an abnormality
12	0xAD90	B	String that shows communication[#3]	--	--
13	0xAD91	B	String that shows communication[#3]	Response code from the mail server[#4]	Exception class name in the event of an abnormality
14	0xAD81	A	--	--	Exception class name in the event of an abnormality
15	0xAD84	A	--	--	--
16	0xAD90	B	String that shows communication[#3]	--	--
17	0xAD91	B	String that shows communication[#3]	Response code from the mail server[#4]	Exception class name in the event of an abnormality
18	0xAD85	A	--	--	Exception class name in the event of an abnormality
19[#5]	0xAD8A	A	--	--	--
20	0xAD90	B	String that shows communication[#3]	--	--

Numbers used in the figures[#1]	Event ID	Level	Interface name	Operation name	Option
21[#6]	0xAD91	B	String that shows communication[#3]	Response code from the mail server[#4]	Exception class name in the event of an abnormality
22[#5]	0xAD8B	A	--	--	Exception class name in the event of an abnormality
23	0xAD8E	A	String that shows communication[#3]	--	--
24[#6]	0xAD8F	A	String that shows communication[#3]	Response code from the mail server[#4]	Exception class name in the event of an abnormality
25	0xAD86	A	--	--	--
26	0xAD8E	A	String that shows communication[#3]	--	--
27	0xAD8F	A	String that shows communication[#3]	Response code from the mail server[#4]	Exception class name in the event of an abnormality
28	0xAD8C	A	--	--	--
29	0xAD90	B	String that shows communication[#3]	--	--
30	0xAD91	B	String that shows communication[#3]	Response code from the mail server[#4]	Exception class name in the event of an abnormality
31	0xAD8D	A	--	--	Exception class name in the event of an abnormality
32	0xAD90	B	String that shows communication[#3]	--	--
33	0xAD91	B	String that shows communication[#3]	Response code from the mail server[#4]	Exception class name in the event of an abnormality
34	0xAD88	B	--	--	--
35	0xAD89	B	--	--	Exception class name in the event of an abnormality
36	0xAD87	A	--	--	Exception class name in the event of an abnormality
37	0xAD90	B	String that shows communication[#3]	--	--
38	0xAD91	B	String that shows communication[#3]	Response code from the mail server[#4]	Exception class name in the event of an abnormality

Legend:

A: Standard

B: Advanced

--: Not applicable

#1

Corresponds to the numbers used in Figure 8-82 or Figure 8-83.

#2

Outputs the following items:

host-name port-number timeout-value-when-fetching-connection communication-timeout-value

Each of the items should be output by separating them with a space.

`0` is output if the timeout is set to infinite. `-1` is output, if the user omits the timeout setting.

(Example) `localhost 25 10000 10000`

#3

The following table describes the output contents of the strings that show communication:

Sr. No.	String that shows communication	Communication process
1	-- (Not applicable)	Fetch connection
2	connect-mail-server	Fetch server response on connection
3	USER	Issue `USER` command
4	PASS	Issue `PASS` command
5	QUIT	Issue `QUIT` command
6	STAT	Issue `STAT` command
7	`LIST` command arguments	Issue `LIST` command
8	`UIDL` command arguments	Issue `UIDL` command
9	`RETR` command arguments	Issue `RETR` command
10	`TOP` command arguments	Issue `TOP` command
11	`DELE` command arguments	Issue `DELE` command
12	`NOOP`	Issue `NOOP` command
13	`RSET`	Issue `RSET` command
14	`CAPA`	Issuing the `CAPA` command

#4

Response code is output only if the 1st line of the response is received. Response code is not output, if the response is not received due to the abnormal conditions such as occurrence of IOException in communication.

#5

The following table describes the output contents of responses:

Response	Output contents
Response code is correct (satisfies the RFC specifications)	Response code
Response code is incorrect (does not satisfy the RFC specifications)	Header 4 letters of the 1st line of the response (See KDJE59112-E message for details) • If the 1st line is less than 4 letters Output all the letters present in the 1st line • No response Output empty string

#6

For the commands (`RETR` command, `TOP` command, and `UIDL` command) that receive multiple-line response, return code `0` is output on fetching the 1st line of the response.

8.20 Trace collection points of JSF 2.2

This section describes the trace collection points of JSF 2.2 and the trace information that can be collected.

8.20.1 Trace collection points and the trace information that can be collected

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–121: Details of trace collection points of JSF 2.2

Event ID	No. in the figure#	Trace collection points	Level
0xAF20	1	Immediately before a custom converter is invoked	A
0xAF21	2	Immediately after the processing of the custom converter ends	A
0xAF22	3	Immediately before a custom validator is invoked	A
0xAF23	4	Immediately after the processing of the custom validator ends	A
0xAF24	5	Immediately before `ValueChangeListener` is invoked	A
0xAF25	6	Immediately after the processing of `ValueChangeListener` ends	A
0xAF26	7	Immediately before `ActionListener` is invoked	A
0xAF27	8	Immediately after the processing of `ActionListener` ends	A
0xAF28	9	Immediately before `AjaxBehaviorListener` is invoked	A
0xAF29	10	Immediately after the processing of `AjaxBehaviorListener` ends	A
0xAF2A	11	Immediately before `Action Method` is invoked	A
0xAF2B	12	Immediately after the processing of `Action Method` ends	A
0xAF2C	13	Immediately before `ComponentSystemEventListener` is invoked	A
0xAF2D	14	Immediately after the processing of `ComponentSystemEventListener` ends	A

Legend:

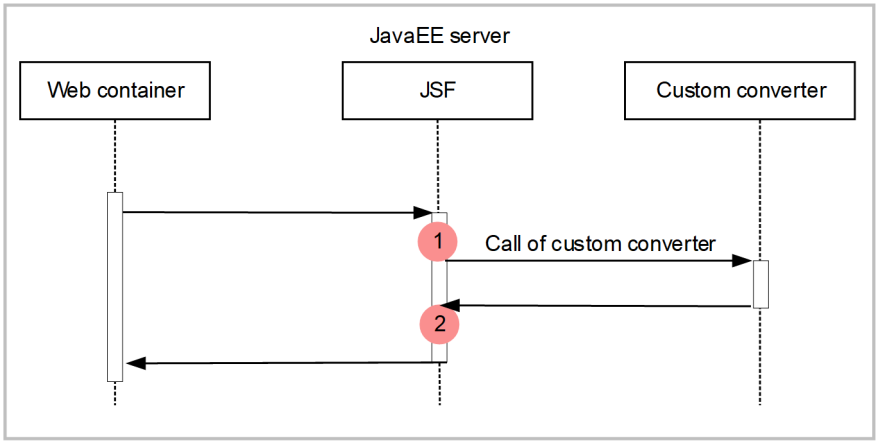
A: Standard

#

Corresponds to the numbers in Figure 8-84 through Figure 8-90.

The following figures show the trace collection points of JSF 2.2.

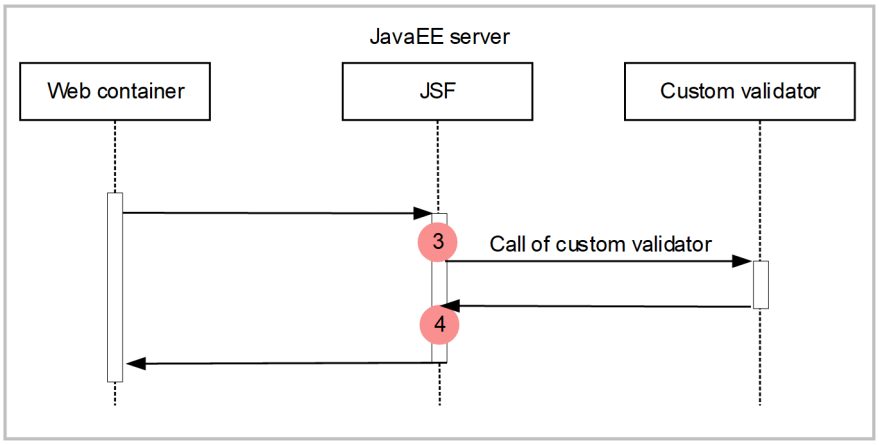
Figure 8–84: Collection points for trace-based performance analysis for JSF 2.2 (custom converter)



Legend:

: Shows trace collection point. PRF trace collection level is "Standard".

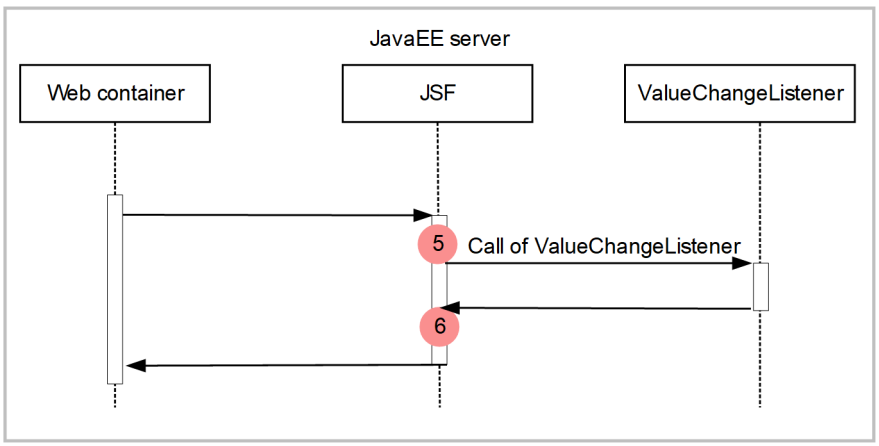
Figure 8–85: Collection points for trace-based performance analysis for JSF 2.2 (custom validator)



Legend:

: Shows trace collection point. PRF trace collection level is "Standard".

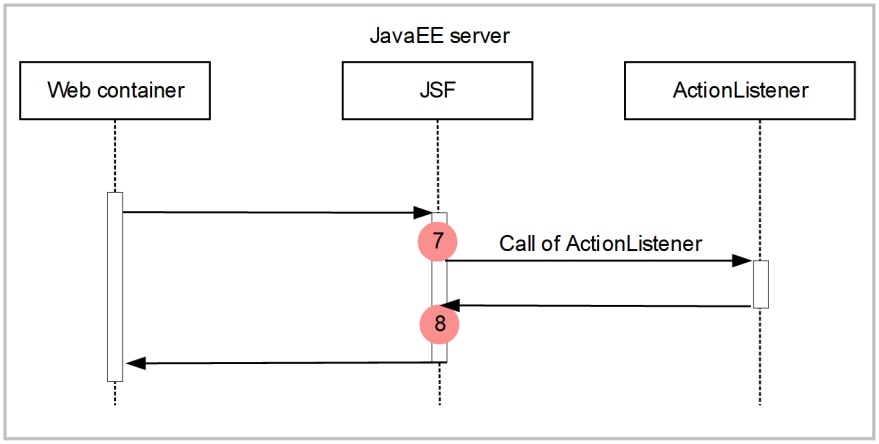
Figure 8–86: Collection points for trace-based performance analysis for JSF 2.2 (ValueChangeListener)



Legend:

: Shows trace collection point. PRF trace collection level is "Standard".

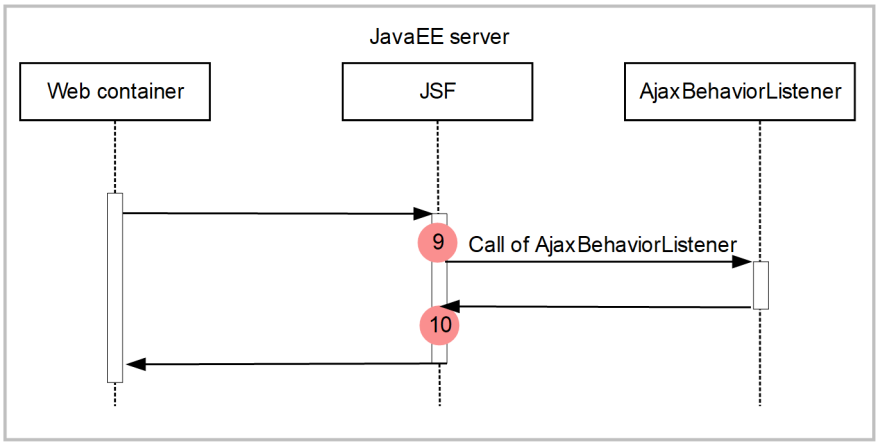
Figure 8–87: Collection points for trace-based performance analysis for JSF 2.2 (ActionListener)



Legend:

: Shows trace collection point. PRF trace collection level is "Standard".

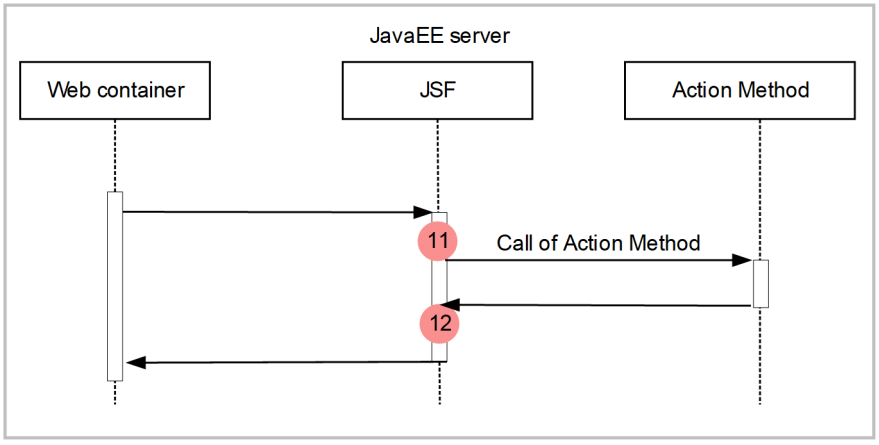
Figure 8–88: Collection points for trace-based performance analysis for JSF 2.2 (AjaxBehaviorListener)



Legend:

: Shows trace collection point. PRF trace collection level is "Standard".

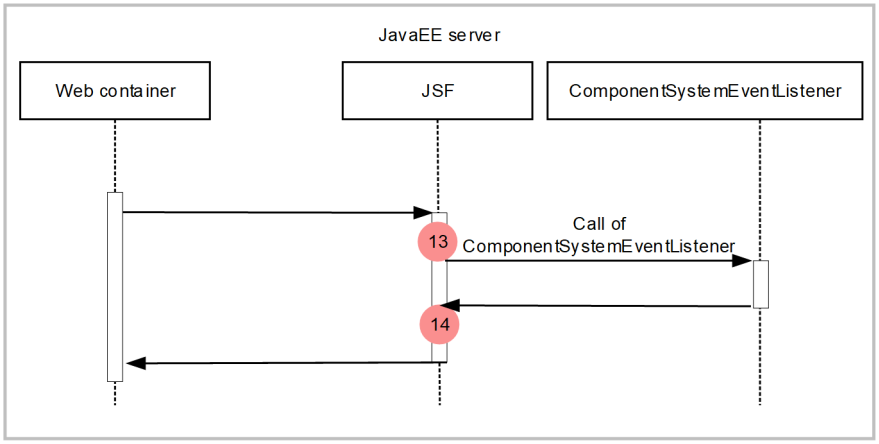
Figure 8–89: Trace collection points for JSF (Action Method)



Legend:

: Shows trace collection point. PRF trace collection level is "Standard".

Figure 8–90: Trace collection points for JSF 2.2 (ComponentSystemEventListener)



Legend:

: Shows trace collection point. PRF trace collection level is "Standard".

8.20.2 Trace information that can be collected

The following table describes the trace information that can be collected in JSF 2.2.

Table 8–122: Trace information that can be collected in JSF 2.2

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xAF20	A	Client ID	Class name and method name of the custom converter	--
2	0xAF21	A	Client ID	Class name and method name of the custom converter	• In a normal state: None • In an abnormal state: Exception name
3	0xAF22	A	Client ID	Class name of the custom validator. If the custom validator is invoked by using `MethodExpression`, `MethodExpression` is acquired. Note When `MethodExpression` with multiple arguments is used, the columns in the trace based performance analysis file (CSV format) might shift.	--
4	0xAF23	A	Client ID	Class name of the custom validator. If the custom validator is invoked by using `MethodExpression`, `MethodExpression` is acquired. Note When `MethodExpression` with multiple arguments is used,	• In a normal state: None • In an abnormal state: Exception name

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
				the columns in the trace based performance analysis file (CSV format) might shift.	
5	0xAF24	A	Client ID	Class name of `ValueChangeListener`. If `ValueChangeListener` is invoked by using `MethodExpression`, `MethodExpression` is acquired.	--
6	0xAF25	A	Client ID	• In a normal state: If `ValueChangeListener` is invoked by using `MethodExpression` and ends successfully, and if a method with one argument is invoked, `one argument` is output. If a method with no argument is invoked, `no argument` is output. • In an abnormal state: None.	• In a normal state: None • In an abnormal state: Exception name
7	0xAF26	A	Client ID	Class name of `ActionListener`. If `ActionListener` is invoked by using `MethodExpression`, `MethodExpression` is acquired.	--
8	0xAF27	A	Client ID	• In a normal state: If `ActionListener` is invoked by using `MethodExpression` and ends successfully, and if a method with one argument is invoked, `one argument` is output. If a method with no argument is invoked, `no argument` is output. • In an abnormal state: None.	• In a normal state: None • In an abnormal state: Exception name
9	0xAF28	A	Client ID	Class name of `AjaxBehaviorListener`. If `AjaxBehaviorListener` is invoked by using `MethodExpression`, `MethodExpression` is acquired.	--
10	0xAF29	A	Client ID	• In a normal state: When `AjaxBehaviorListener` is invoked by using `MethodExpression` and ends successfully, and if a method with one argument is invoked, `one argument` is output.	• In a normal state: None • In an abnormal state: Exception name

No. in the figure#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
				If a method with no argument is invoked, `no argument` is output. • In an abnormal state: None.	
11	0xAF2A	A	Client ID	--	--
12	0xAF2B	A	Client ID	--	• In a normal state: None • In an abnormal state: Exception name
13	0xAF2C	A	Client ID	Class name of `ComponentSystemEventListener`. If `ComponentSystemEventListener` is invoked by using `MethodExpression`, `MethodExpression` is acquired.	--
14	0xAF2D	A	Client ID	• In a normal state: If `ComponentSystemEventListener` is invoked by using `MethodExpression` and ends successfully, and if a method with one argument is invoked, `one argument` is output. If a method with no argument is invoked, `no argument` is output. • In an abnormal state: None.	• In a normal state: None • In an abnormal state: Exception name

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-84 through Figure 8-90.

8.20.3 Data output to the exception log

If an exception name is output to the additional information (OPT) column at a trace based performance analysis collection point, detailed information about the exception is output to the exception log at the same time. You can analyze the failure by comparing the exception name output by the trace based performance analysis with the exception log. However, if you intentionally raise exceptions in a batch application or other application and use them to control the processing of jobs, the amount of log output might increase because detailed information is output to the exception log each time an exception is raised. When executing such an application, set a larger size for the exception log in advance.

8.21 Trace collection points of CDI

This section describes the trace collection points of CDI and the trace information that can be collected.

8.21.1 Trace collection points of CDI and the trace information that can be collected

This subsection 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 2.2 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 8–123: Details of the trace collection points when a combination of JSF 2.2 and CDI is used

Event ID	No. in the figure#	Trace collection points	Level
0xb002	1	When the reading of the JSF 2.2 settings required for using CDI starts	A
0xb003	2	When the reading of the JSF 2.2 settings required for using CDI ends (normal termination)	A
0xb004	3	When the JSF 2.2 preparations required for using CDI start	A
0xb005	4	When the JSF 2.2 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 8-91 and Figure 8-92.

The following figure shows the trace collection points.

Figure 8–91: Trace collection points when a combination of JSF 2.2 and CDI is used (when the JSF 2.2 settings are read and prepared)

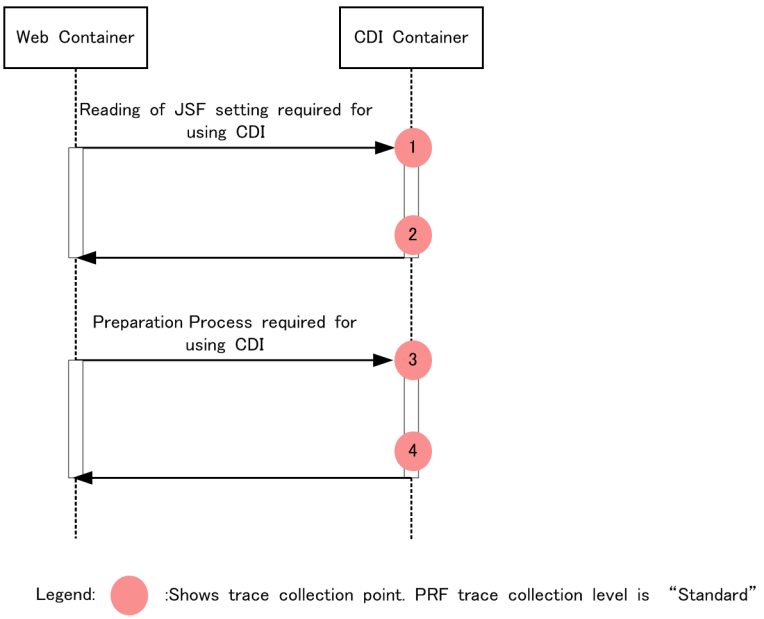
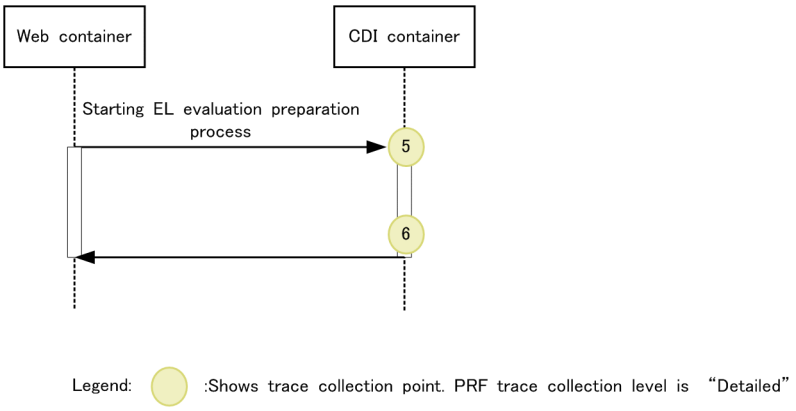


Figure 8–92: Trace collection points when a combination of JSF 2.2 and CDI is used (for EL evaluation)



(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 8–124: 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:

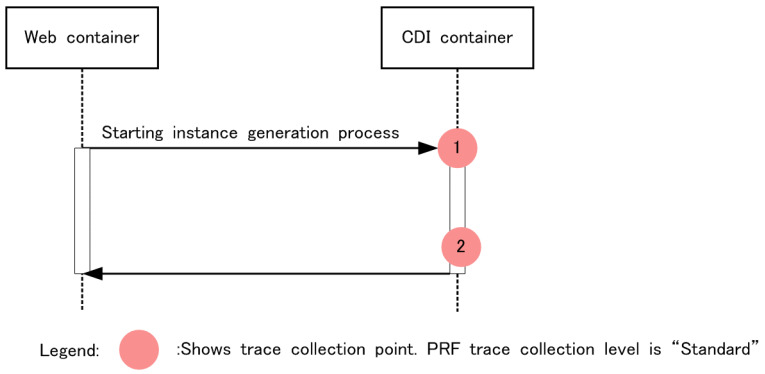
A: Standard

#:

Corresponds to the numbers in Figure 8-93.

The following figure shows the trace collection points.

Figure 8–93: Trace collection points when a combination of servlets, filters, listeners, and CDI is used



(2) Trace information that can be collected

(a) When a combination of JSF 2.2 and CDI is used

The following table describes the trace information that can be collected when a combination of JSF 2.2 and CDI is used.

Table 8–125: Trace information that can be collected when a combination of JSF 2.2 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 8-91 and Figure 8-92.

#2:

The trace information for the reading of the JSF 2.2 settings required for using CDI and for the JSF 2.2 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 2.2 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 8–126: 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 8-93.

Note that the trace information is collected when the servlet/filter/listener interfaces are generated.

8.22 Trace collection points when a J2EE server is started or terminated

Trace information can be collected when the startup processing of a J2EE server finishes, and when the termination processing of the J2EE server starts.

8.22.1 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 8–127: Details of trace collection points in a J2EE server

Event ID	Trace acquisition points	Level
0x8FFE	When the startup processing of the J2EE server is complete (message KDJE30028-I is output)	A
0x8FFF	When the shutdown of the J2EE server starts (message KDJE30031-I is output)	A

Legend:
A: Standard

8.22.2 Trace information that can be collected

The trace information that can be collected when a J2EE server is started or terminated is as follows:

- Event ID
  0x8FFE, 0x8FFF
- PRF trace collection level
  All *Standard*.
- Interface name, operation name, and optional
  The information is not output.

8.23 Trace collection points of an application

The application trace is output in the user-extended trace based performance analysis. The user-extended trace based performance analysis outputs the trace information when the methods specified in the configuration file for the user-extended trace based performance analysis are invoked.

This section describes the trace points and trace information for the user-extended trace based performance analysis.

8.23.1 Trace collection points and PRF trace collection levels

The following table describes the points at which the user-extended trace based performance analysis outputs the trace.

Table 8–128: Trace collection points of the user-extended trace based performance analysis

Trace collection points	No. in the figure[#1]	Explanation	Level
Normal entry of method	1	Immediately after a method is invoked.	Trace collection levels specified in the user-extended trace based performance analysis configuration file[#2]
Normal exit of method	2	Immediately before a method terminates normally.	
Abnormal exit of method	3	Immediately before a method terminates abnormally due to an exception or error.	

Legend:

A: Standard

#1:

Corresponds to the numbers in Figure 8-94.

#2:

For details on the specification of the trace collection levels specified in the user-extended trace based performance analysis configuration file, see *7.5.3 Settings for the methods to be traced by the user-extended trace based performance analysis*.

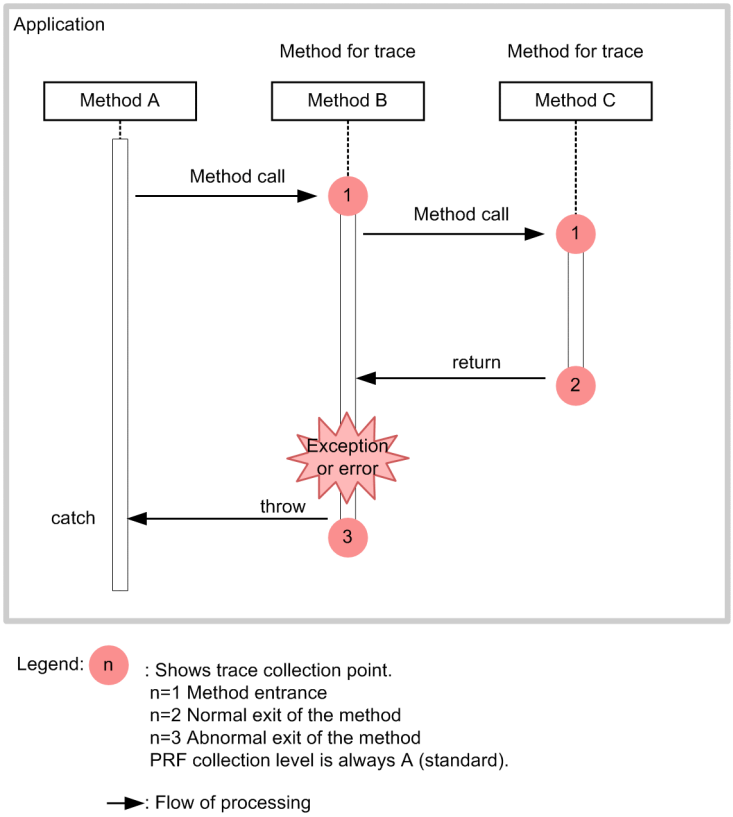
Reference note

If a method to be traced terminates because an exception or error occurred in the method to be traced or in the invocation destination method of the method to be traced, the trace is collected as the abnormal exit of the method.

If an exception or error is processed by the `try-catch` syntax in a method to be traced, and if an exception or error is not thrown at the invocation source of the method to be traced, the trace is collected as the normal exit of the method.

The following figure shows the trace collection points in the user-extended trace based performance analysis.

Figure 8–94: Trace collection points of the user-extended trace based performance analysis



8.23.2 Trace information that can be collected

This subsection describes the trace information of the user-extended trace based performance analysis.

(1) Trace information of the user-extended trace based performance analysis

The following table describes the trace information of the user-extended trace based performance analysis.

Table 8–129: Trace information of the user-extended trace based performance analysis

No. in the figure[#1]	Information that you can acquire				
	Event ID (Event)	Return code (Rc)	Interface name (INT)	Operation information (OPR)	Additional information[#2] (OPT/ASCII[#3])
1	Event ID specified in the configuration file for the user-extended trace based performance analysis. When not specified, `0xae00`.	0	Identity ID specified by the user in the configuration file for the user-extended trace based performance analysis.	--	The following information is output: • Package name • Class name • Method name
2	Event ID specified in the configuration file for the user-extended trace based	0	Identity ID specified by the user in the configuration file for the user-extended	Line number of the text executed last by the method [#4].	The following information is output: • Package name • Class name

No. in the figure[#1]	Information that you can acquire				
	Event ID (Event)	Return code (Rc)	Interface name (INT)	Operation information (OPR)	Additional information[#2] (OPT/ASCII[#3])
	performance analysis + 1. When not specified, `0xae01`.		trace based performance analysis.		• Method name
3	Event ID specified in the configuration file for the user-extended trace based performance analysis +1. When not specified, `0xae01`.	1	Identity ID specified by the user in the configuration file for the user-extended trace based performance analysis.	Class name of the exception or error[#5].	The following information is output: • Package name • Class name • Method name

Legend:

--: No output.

#1:

Corresponds to the numbers in Figure 8-94.

#2:

The user-extended trace based performance analysis outputs the method name with the output level specified in the `jvm.userprf.LogLevel` property. For details on the output levels and the information output at each level, see *(2) Output levels*.

#3:

If the additional information output to the ASCII area exceeds 256 ASCII characters, 256 characters are output from the beginning.

#4:

Output when `true` is specified for the `jvm.userprf.LineNumber` property. For details on the `jvm.userprf.LineNumber` property, see *14.3 Properties used in JavaVM* in the manual *uCosminexus Application Server Definition Reference Guide*.

#5:

When `true` is specified for the `jvm.userprf.ThrowableName` property, the class name is output with the output level specified in the `jvm.userprf.LogLevel` property, and the editing method specified in the `jvm.userprf.ThrowableNameEditMethod` property. For details on the `jvm.userprf.ThrowableName` and `jvm.userprf.ThrowableNameEditMethod` properties, see *14.3 Properties used in JavaVM* in the manual *uCosminexus Application Server Definition Reference Guide*.

(2) Output levels

You specify the trace information output level for the user-extended trace based performance analysis in the `jvm.userprf.LogLevel` property.

The following table describes the output levels and the trace information that is output.

Table 8–130: Specification of the output levels and the trace information that is output

Specification in the jvm.userprf.LogLevel property	Additional information (OPT/ASCII)	Operation information for abnormal exit of the method (OPR)
`class`	Class name	Class name of the exception or error
`package`	Fully qualified class name	Fully qualified class name of the exception or error
`method`	Fully qualified class name + method name	
`signature`	Fully qualified class name + method name + method argument type	

For details on the `jvm.userprf.LogLevel` property, see *14.3 Properties used in JavaVM* in the manual *uCosminexus Application Server Definition Reference Guide*.

(3) Example of output of the user-extended trace based performance analysis

The following are the examples of output of the user-extended trace based performance analysis.

(a) Example when the output contents of the trace information are not changed

The following is an example of settings when the output contents of the trace information are not changed:

Example of settings in the Easy Setup definition file

```
...
<param>
<param-name>UserPrfText</param-name>
<param-value>
<![CDATA[
com.sample.ClassA.method1(int),test00,false
]]>
</param-value>
</param>
<param>
<param-name>add.jvm.arg</param-name>
<param-value>-Djvm.userprf.Enable=true</param-value>
</param>
...
```

Output contents

Event	Rc	INT	OPR	ASCII
0xae00	0	test00	(Blank)	ClassA
0xae01	0	test00	(Blank)	ClassA

The trace information is output as follows in this example:

- The event ID settings are omitted for the method to be traced, so if the method to be traced is invoked, the default value of `0xae00` is output as the event ID at the method entry, and `0xae01` is output as the event ID at the method exit.

(b) Example for the output of the line number in the operation information

The following is an example of settings when the configuration file for the user-extended trace based performance analysis (`/test/setting.txt`) is used to output the line number in the operation information:

Example of settings in the Easy Setup definition file

```
...
<param>
<param-name>add.jvm.arg</param-name>
<param-value>-Djvm.userprf.Enable=true</param-value>
<param-value>-Djvm.userprf.File=/test/setting.txt</param-value>
<param-value>-Djvm.userprf.LineNumber=true</param-value>
</param>
...
```

Example of settings in the configuration file for the user-extended trace based performance analysis (`setting.txt`)

```
com.sample.ClassA.method1(java.lang.String),test00,false,0xae77
com.sample.ClassB.method2(boolean),test01.false
```

Output contents

`jvm.userprf.LineNumber=true` is specified in this example, so the line number executed last by each method is output in the operation information (OPR area) if the method exits normally.

Event	Rc	INT	OPR	ASCII
0xae77	0	test00	(Blank)	ClassA
0xae78	0	test00	324	ClassA
0xae00	0	test01	(Blank)	ClassB
0xae01	0	test01	15	ClassB

The trace information is output as follows in this example:

- In the second line specified in the configuration file for the user-extended trace based performance analysis, the event ID settings are omitted for the method to be traced, so if the method to be traced is invoked, the default value of `0xae00` is output as the event ID at the method entry, and `0xae01` is output as the event ID at the method exit.

(c) Example for the output of the exception or error class name

The following is an example of settings when the class name of the thrown exception is output in the operation information (OPR area) if the method exits abnormally. Note that this example describes the case in which ClassC exists in the subclass ClassA, and ClassA.method1 is overridden in ClassC.

Example of settings in the Easy Setup definition file

```
...
<param>
<param-name>UserPrfText</param-name>
<param-value>
<![CDATA[
com.sample.ClassA.method1(),test00,true,0xae0a
]]>
</param-value>
</param>
<param>
<param-name>add.jvm.arg</param-name>
<param-value>-Djvm.userprf.Enable=true</param-value>
<param-value>-Djvm.userprf.LineNumber=true</param-value>
<param-value>-Djvm.userprf.ThrowableName=true</param-value>
</param>
...
```

Output contents

Event	Rc	INT	OPR	ASCII
0xae0a	0	test00	(Blank)	ClassA
0xae0b	0	test00	324	ClassA
0xae0a	0	test00	(Blank)	ClassC
0xae0b	1	test00	IOException	ClassC

The trace information is output as follows in this example:

- `true` is set in the subclass flag, so the information of `ClassC.method1` that overrides `ClassA.method1` is also output.
- `jvm.userprf.LineNumber=true` is specified, so the line number executed last by each method is output in the operation information (OPR area) if the method exits normally.
- `jvm.userprf.ThrowableName=true` is set, so the class name of the thrown exception is output in the operation information (OPR area) if the method exits abnormally. However, the `jvm.userprf.LogLevel` property is not specified, so only the class name of the thrown exception is output.

(d) Example of changing the output method of the exception or error class name

The following is an example of settings when the output method of the exception or error class name is changed.

Example of settings in the Easy Setup definition file

```
...
<param>
<param-name>UserPrfText</param-name>
<param-value>
<![CDATA[
com.sample.ClassA.method1(),test00,false
]]>
</param-value>
</param>
<param>
<param-name>add.jvm.arg</param-name>
<param-value>-Djvm.userprf.Enable=true</param-value>
<param-value>-Djvm.userprf.ThrowableName=true</param-value>
<param-value>-Djvm.userprf.ThrowableNameEditMethod=FRONT_CUT</param-value>
<param-value>-Djvm.userprf.LogLevel=method</param-value>
</param>
...
```

Output contents

Event	Rc	INT	OPR	ASCII
0xae00	0	test00	(Blank)	com.sample.ClassA.method1
0xae01	1	test00	*ment.IllegalClassFormatException	com.sample.ClassA.method1

The trace information is output as follows in this example:

- The event ID settings are omitted for the method to be traced, so if the method to be traced is invoked, the default value of `0xae00` is output as the event ID at the method entry, and `0xae01` is output as the event ID at the method exit.
- `jvm.userprf.ThrowableName=true` is specified, so the exception or error class name is output to the operation information (OPR area) if the method exits abnormally. However, the thrown exception name is 33 characters or more and `jvm.userprf.ThrowableNameEditMethod=FRONT_CUT` is specified, so the name is output with the front part omitted. The omitted part is displayed using `*` (asterisk).
- `jvm.userprf.LogLevel=method` is specified, so the fully qualified class name + method name is output in the additional information (ASCII area).

8.24 Trace collection points of JAX-RS

This section describes the trace collection points of JAX-RS and the trace information that can be collected.

8.24.1 Trace collection points and trace information that can be collected

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–131: Details of trace collection points in JAX-RS

Event ID	No. in the figures#	Trace acquisition points	Level
0xD000	1	At the start of the HTTP method call	A
0xD001	2	At the end of the HTTP method call	A
0xD002	3	Before sending the HTTP message of the client library	A
0xD003	4	After receiving the HTTP message of the client library	A
0xD004	5	Before calling a web resource	A
0xD005	6	After calling a web resource	A

Legend:

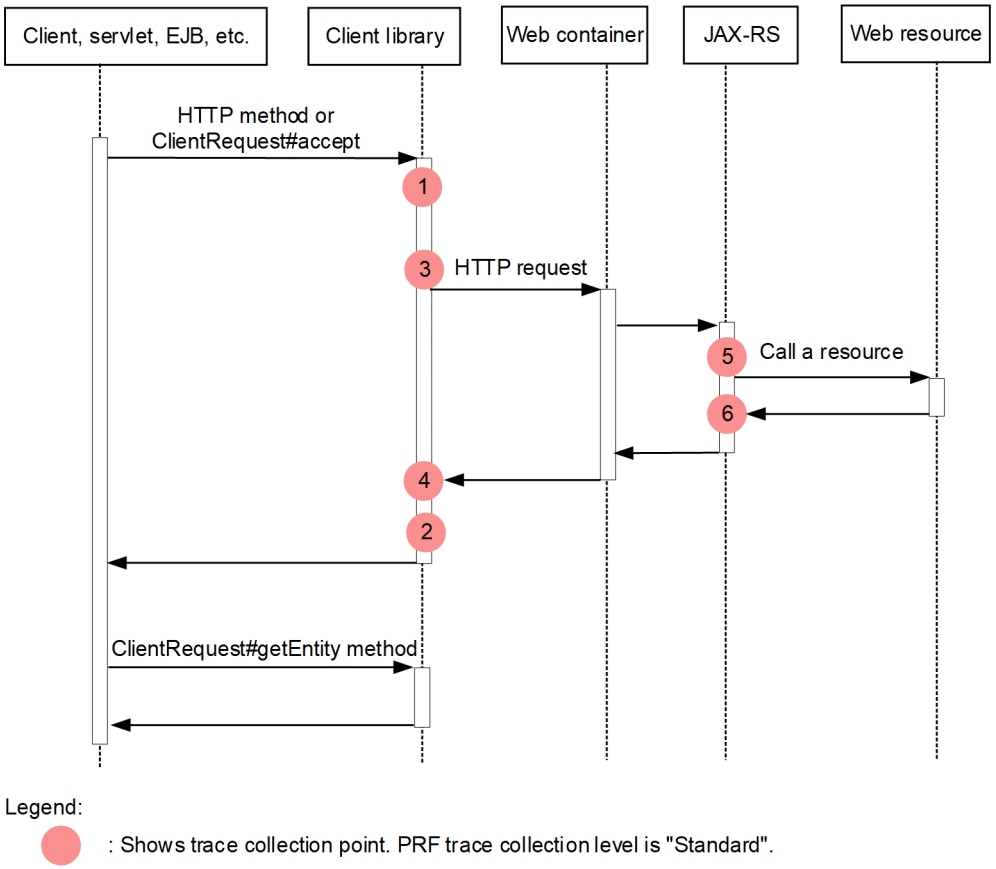
A: Standard

#

Corresponds to the numbers in Figure 8-95.

The following figure showsthe trace collection points in JAX-RS.

Figure 8–95: Trace collection points of JAX-RS



8.24.2 Trace information that can be collected

The following table describes the trace information that can be collected in JAX-RS.

Table 8–132: Trace information that can be collected in JAX-RS

No. in the figures#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xD000	A	Class name	Method name	--
2	0xD001	A	Class name	Method name	• In a normal state: None • In an abnormal state: Exception name
3	0xD002	A	Class name	Method name	Endpoint URI
4	0xD003	A	Class name	Method name	• In a normal state: None • In an abnormal state: Exception name

No. in the figures#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
5	0xD004	A	Class name	Method name	The call type is one of the following: • ObjectOut Invoker • ResponseOut Invoker • TypeOutInvoker • VoidOutInvoker • VoidToVoid Dispatcher
6	0xD005	A	Class name	Method name	• In a normal state: None • In an abnormal state: Exception name

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-95.

8.24.3 Data output to the exception log

If an exception name is output to the additional information (OPT) column at a trace based performance analysis collection point, detailed information about the exception is output to the exception log at the same time. You can analyze the failure by comparing the exception name output by the trace based performance analysis with the exception log. However, if you intentionally raise exceptions in a batch application or other application and use them to control the processing of jobs, the amount of log output might increase because detailed information is output to the exception log each time an exception is raised. When executing such an application, set a larger size for the exception log in advance.

8.25 Trace collection points of a Java batch

This section describes the trace collection points of a Java batch and the trace information that can be collected.

8.25.1 Trace collection points and trace information that can be collected

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–133: Details of trace collection points in a Java batch

Event ID	No. in the figures#	Trace acquisition points	Level
0xD020	1	Immediately after the `javax.batch.runtime.BatchRuntime.getJobOperator()` method starts	A
0xD021	2	Immediately before the `javax.batch.runtime.BatchRuntime.getJobOperator()` method ends	A
0xD022	3	Immediately after the `javax.batch.operations.JobOperator.start(String jobXMLName, Properties jobParameters)` method starts	A
0xD023	4	Immediately before the `javax.batch.operations.JobOperator.start(String jobXMLName, Properties jobParameters)` method ends	A
0xD024	5	Immediately after the `javax.batch.operations.JobOperator.restart(long executionId, Properties restartParameters)` method starts	A
0xD025	6	Immediately before the `javax.batch.operations.JobOperator.restart(long executionId, Properties restartParameters)` method ends	A
0xD026	7	Immediately after the `javax.batch.operations.JobOperator.stop(long executionId)` method starts	A
0xD027	8	Immediately before the `javax.batch.operations.JobOperator.stop(long executionId)` method ends	A
0xD028	9	Immediately after the `javax.batch.operations.JobOperator.abandon(long executionId)` method starts	A
0xD029	10	Immediately before the `javax.batch.operations.JobOperator.abandon(long executionId)` method ends	A
0xD02A	11	Immediately after the job starts	A
0xD02B	12	Immediately after the job ends	A
0xD02C	13	Immediately after the step processing starts	A
0xD02D	14	Immediately before the step processing ends	A

Event ID	No. in the figures#	Trace acquisition points	Level
0xD02E	15	Immediately before invoking the `open(Serializable checkpoint)` method of the implementation class of `javax.batch.api.chunk.ItemReader`	A
0xD02F	16	Immediately after the processing of the `open(Serializable checkpoint)` method of the implementation class of `javax.batch.api.chunk.ItemReader` is complete	A
0xD030	17	Immediately before invoking the `close()` method of the implementation class of `javax.batch.api.chunk.ItemReader`	A
0xD031	18	Immediately after the processing of the `close()` method of the implementation class of `javax.batch.api.chunk.ItemReader` is complete	A
0xD032	19	Immediately before invoking the `readItem()` method of the implementation class of `javax.batch.api.chunk.ItemReader`	A
0xD033	20	Immediately after the processing of the `readItem()` method of the implementation class of `javax.batch.api.chunk.ItemReader` is complete	A
0xD034	21	Immediately before invoking the `checkpointInfo()` method of the implementation class of `javax.batch.api.chunk.ItemReader`	A
0xD035	22	Immediately after the processing of the `checkpointInfo()` method of the implementation class of `javax.batch.api.chunk.ItemReader` is complete	A
0xD036	23	Immediately before invoking the `processItem(Object item)` method of the implementation class of `javax.batch.api.chunk.ItemProcessor`	A
0xD037	24	Immediately after the processing of the `processItem(Object item)` method of the implementation class of `javax.batch.api.chunk.ItemProcessor` is complete	A
0xD038	25	Immediately before invoking the `open(Serializable checkpoint)` method of the implementation class of `javax.batch.api.chunk.ItemWriter`	A
0xD039	26	Immediately after the processing of the `open(Serializable checkpoint)` method of the implementation class of `javax.batch.api.chunk.ItemWriter` is complete	A
0xD03A	27	Immediately before invoking the `close()` method of the implementation class of `javax.batch.api.chunk.ItemWriter`	A
0xD03B	28	Immediately after the processing of the `close()` method of the implementation class of `javax.batch.api.chunk.ItemWriter` is complete	A
0xD03C	29	Immediately before invoking the `writeItems(List<Object> items)` method of the implementation class of `javax.batch.api.chunk.ItemWriter`	A
0xD03D	30	Immediately after the processing of the `writeItems(List<Object> items)` method of the implementation class of `javax.batch.api.chunk.ItemWriter` is complete	A
0xD03E	31	Immediately before invoking the `checkpointInfo()` method of the implementation class of `javax.batch.api.chunk.ItemWriter`	A

Event ID	No. in the figures#	Trace acquisition points	Level
0xD03F0	32	Immediately after the processing of the `checkpointInfo()` method of the implementation class of `javax.batch.api.chunk.ItemWriter` is complete	A
0xD040	33	Immediately before invoking the `beforeJob()` method of the implementation class of `javax.batch.api.listener.JobListener`	A
0xD041	34	Immediately after the processing of the `beforeJob()` method of the implementation class of `javax.batch.api.listener.JobListener` is complete	A
0xD042	35	Immediately before invoking the `afterJob()` method of the implementation class of `javax.batch.api.listener.JobListener`	A
0xD043	36	Immediately after the processing of the `afterJob()` method of the implementation class of `javax.batch.api.listener.JobListener` is complete	A
0xD044	37	Immediately before invoking the `beforeStep()` method of the implementation class of `javax.batch.api.listener.StepListener`	A
0xD045	38	Immediately after the processing of the `beforeStep()` method of the implementation class of `javax.batch.api.listener.StepListener` is complete	A
0xD046	39	Immediately before invoking the `afterStep()` method of the implementation class of `javax.batch.api.listener.StepListener`	A
0xD047	40	Immediately after the processing of the `afterStep()` method of the implementation class of `javax.batch.api.listener.StepListener` is complete	A
0xD048	41	Immediately before invoking the `beforeChunk()` method of the implementation class of `javax.batch.api.chunk.listener.ChunkListener`	A
0xD049	42	Immediately after the processing of the `beforeChunk()` method of the implementation class of `javax.batch.api.chunk.listener.ChunkListener` is complete	A
0xD04A	43	Immediately before invoking the `afterChunk()` method of the implementation class of `javax.batch.api.chunk.listener.ChunkListener`	A
0xD04B	44	Immediately after the processing of the `afterChunk()` method of the implementation class of `javax.batch.api.chunk.listener.ChunkListener` is complete	A
0xD04C	45	Immediately before invoking the `onError(Exception ex)` method of the implementation class of `javax.batch.api.chunk.listener.ChunkListener`	A
0xD04D	46	Immediately after the processing of the `onError(Exception ex)` method of the implementation class of `javax.batch.api.chunk.listener.ChunkListener` is complete	A

Legend:

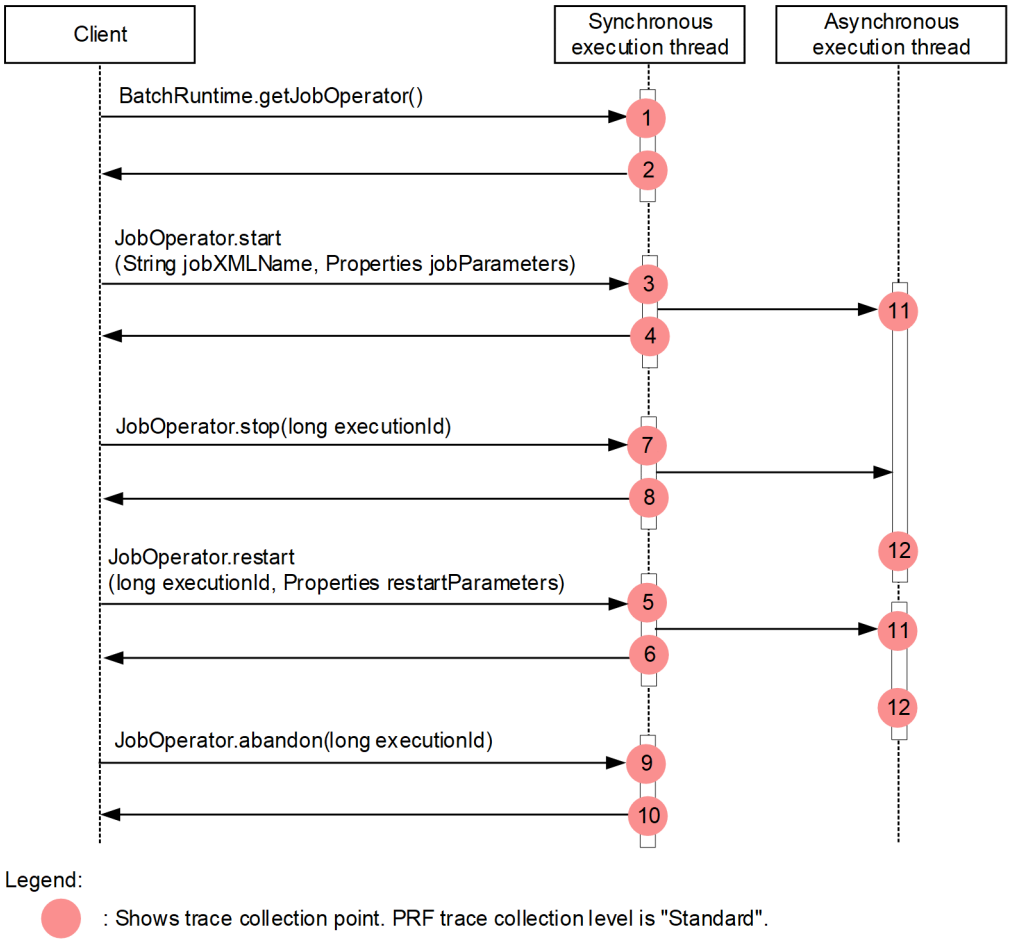
A: Standard

#

Corresponds to the numbers in Figure 8-96 through Figure 8-99.

The following figures show the trace collection points in Java batch.

Figure 8–96: Trace collection points of Java batch (points output between the client, synchronous execution thread, and asynchronous execution thread)



Legend:

: Shows trace collection point. PRF trace collection level is "Standard".

Figure 8–97: Trace collection points of Java batch (points output during job execution)

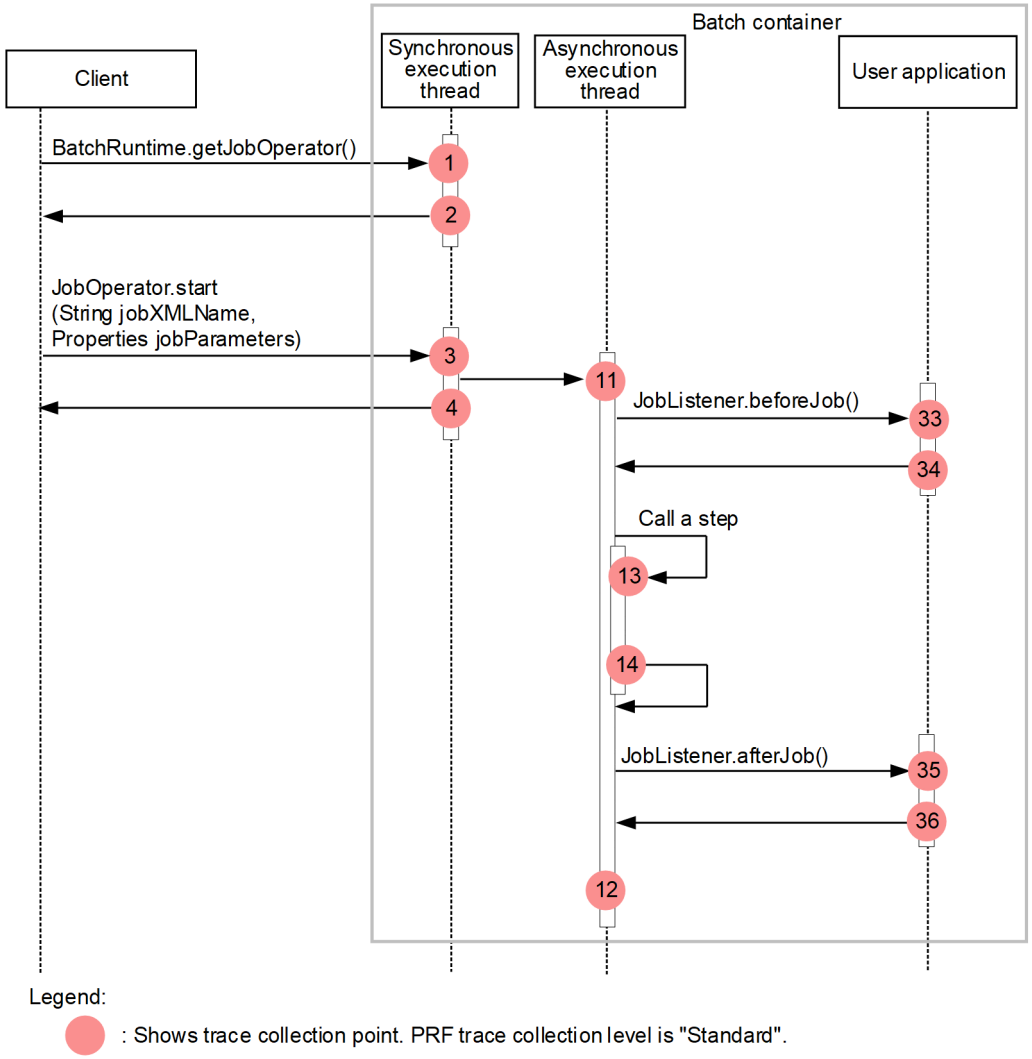


Figure 8–98: Trace collection points of Java batch (points output during step processing)

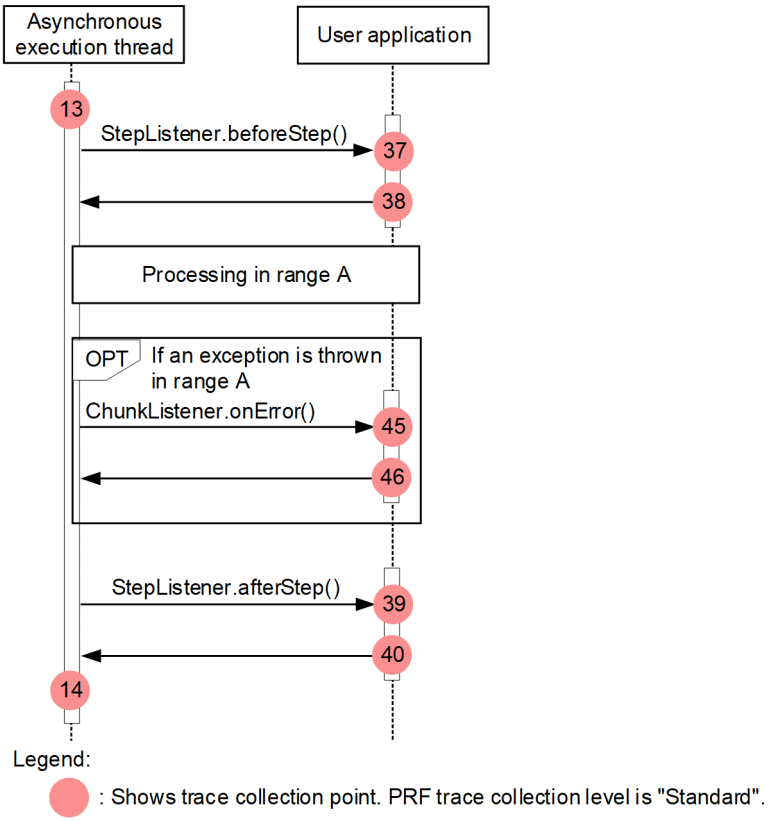
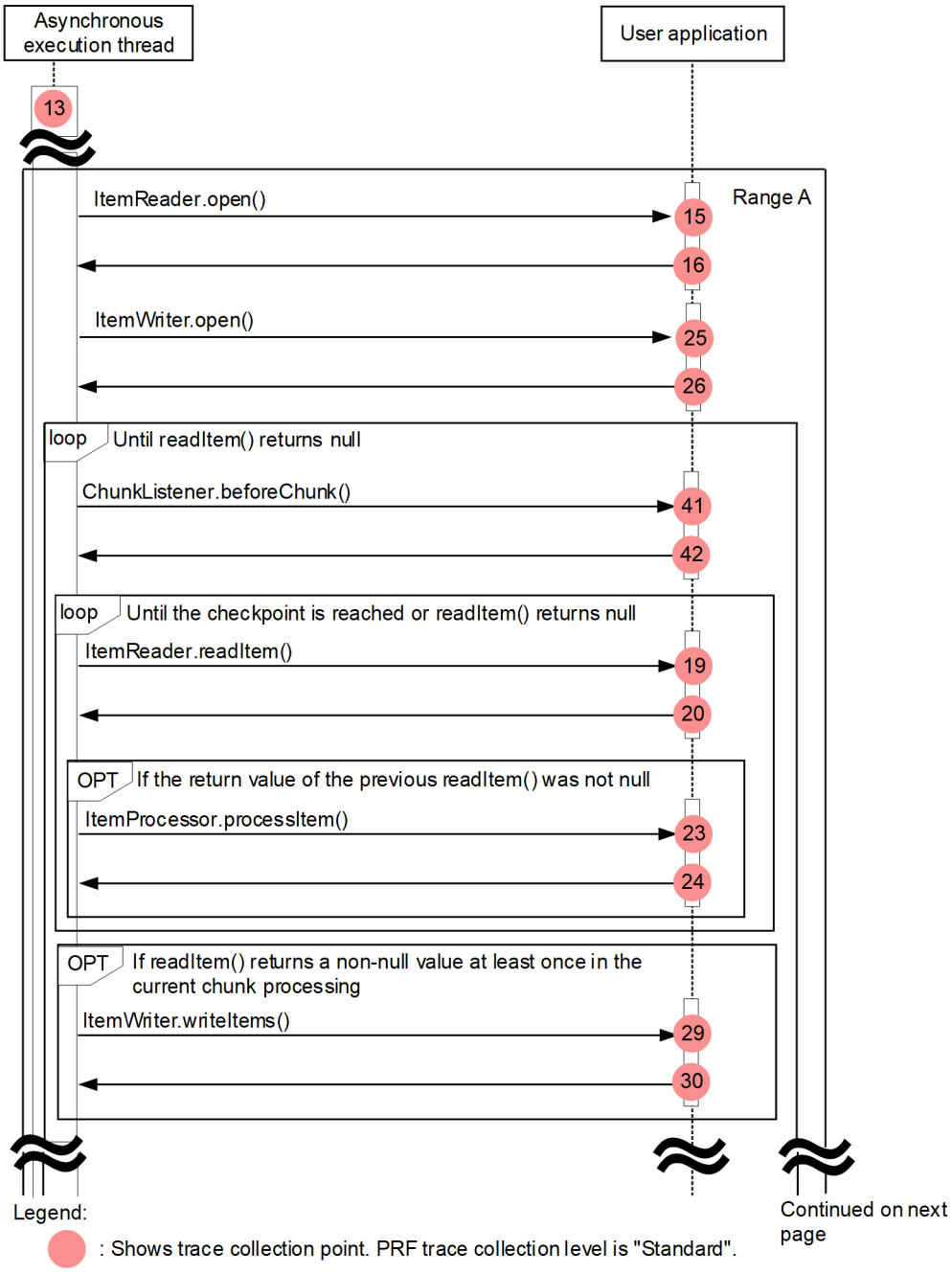
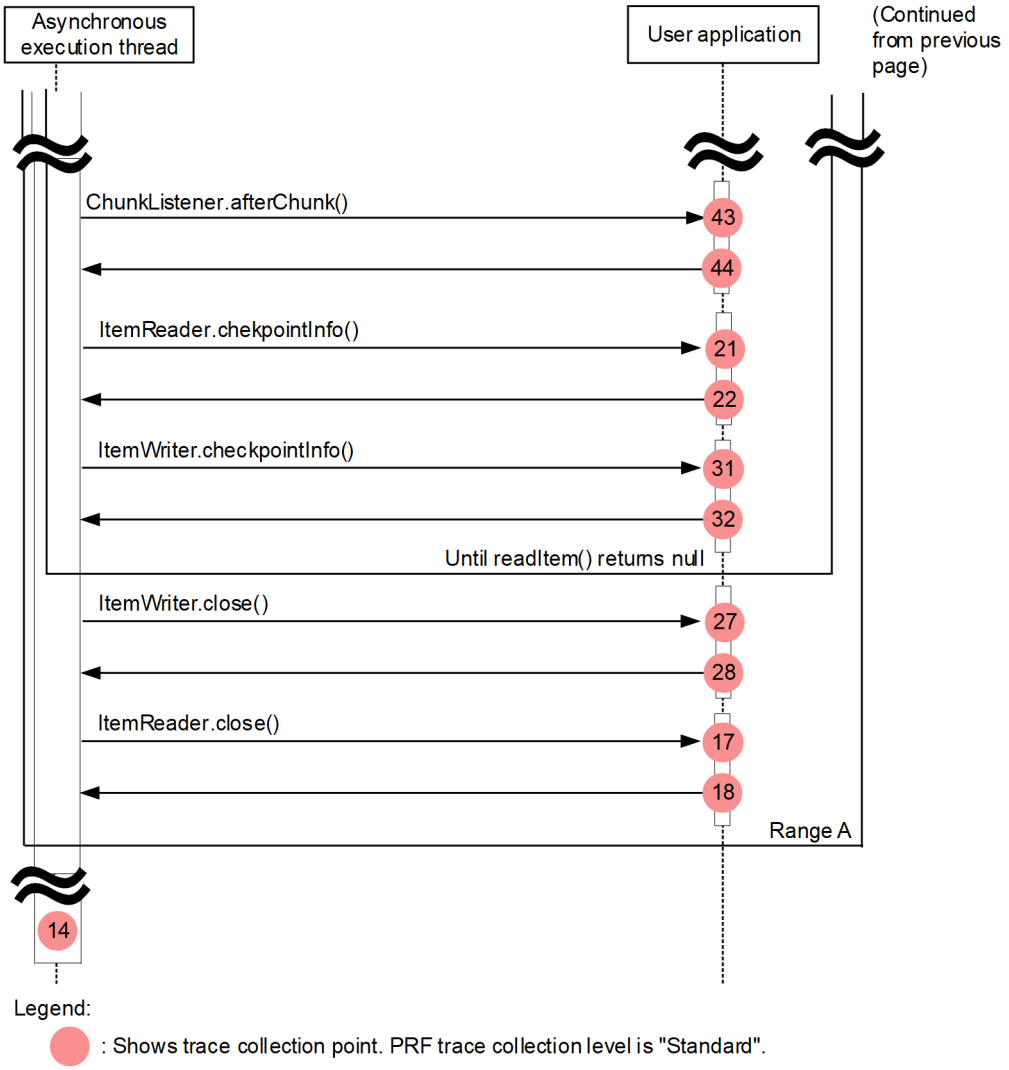


Figure 8–99: Trace collection points of Java batch (points output during step processing (processing in range A))





8.25.2 Trace information that can be collected

The following table describes the trace information that can be collected in a Java batch.

Table 8–134: Trace information that can be collected in a Java batch

No. in the figures#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xD020	A	--	--	--
2	0xD021	A	--	--	• In a normal state: None • In an abnormal state: Exception name
3	0xD022	A	--	--	jobXMLName, which is an argument of

No. in the figures#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
					`JobOperator.start()`
4	0xD023	A	instanceId	executionId	• In a normal state: None • In an abnormal state: Exception name
5	0xD024	A	--	executionId, which is an argument of `JobOperator.restart()`	--
6	0xD025	A	instanceId	Newly assigned executionId	• In a normal state: None • In an abnormal state: Exception name
7	0xD026	A	--	executionId, which is an argument of `JobOperator.stop()`	--
8	0xD027	A	--	--	• In a normal state: None • In an abnormal state: Exception name
9	0xD028	A	--	executionId, which is an argument of `JobOperator.abandon()`	--
10	0xD029	A	--	--	• In a normal state: None • In an abnormal state: Exception name
11	0xD02A	A	--	--	--
12	0xD02B	A	Job batch status	--	• In a normal state: Job exit status • In an abnormal state: Exception name
13	0xD02C	A	--	Step ID	--
14	0xD02D	A	• When TransitionElement is applied: Name of the applied TransitionElement (next, fail, end, or stop) • When the next attribute is applied: next	• When the next ExecutionElement exists: ID of the next ExecutionElement to be executed • In other cases: None	• In a normal state: Step exit status • In an abnormal state: Exception name

No. in the figures#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
			• In other cases: Step batch status		
15	0xD02E	A	Called class name	--	--
16	0xD02F	A	Called class name	--	• In a normal state: None • In an abnormal state: Name of the exception raised from UP
17	0xD030	A	Called class name	--	--
18	0xD031	A	Called class name	--	• In a normal state: None • In an abnormal state: Name of the exception raised from UP
19	0xD032	A	Called class name	--	--

20	0xD033	A	Called class name	• When the return value is `null`:
				`null`
				• When the return value is not `null`:
				`not null`
				• In an abnormal state:
				None
					• In a normal state:
					None
					• In an abnormal state:
					Name of the exception raised from UP

21	0xD034	A	Called class name	--	--
22	0xD035	A	Called class name	--	• In a normal state: None • In an abnormal state: Name of the exception raised from UP
23	0xD036	A	Called class name	--	--
24	0xD037	A	Called class name	--	• In a normal state: None • In an abnormal state: Name of the exception raised from UP
25	0xD038	A	Called class name	--	--

No. in the figures#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
26	0xD039	A	Called class name	--	• In a normal state: None • In an abnormal state: Name of the exception raised from UP
27	0xD03A	A	Called class name	--	--
28	0xD03B	A	Called class name	--	• In a normal state: None • In an abnormal state: Name of the exception raised from UP
29	0xD03C	A	Called class name	--	--
30	0xD03D	A	Called class name	--	• In a normal state: None • In an abnormal state: Name of the exception raised from UP
31	0xD03E	A	Called class name	--	--
32	0xD03F0	A	Called class name	--	• In a normal state: None • In an abnormal state: Name of the exception raised from UP
33	0xD040	A	Called class name	--	--
34	0xD041	A	Called class name	--	• In a normal state: None • In an abnormal state: Name of the exception raised from UP
35	0xD042	A	Called class name	--	--
36	0xD043	A	Called class name	--	• In a normal state: None

No. in the figures#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
					• In an abnormal state: Name of the exception raised from UP
37	0xD044	A	Called class name	--	--
38	0xD045	A	Called class name	--	• In a normal state: None • In an abnormal state: Name of the exception raised from UP
39	0xD046	A	Called class name	--	--
40	0xD047	A	Called class name	--	• In a normal state: None • In an abnormal state: Name of the exception raised from UP
41	0xD048	A	Called class name	--	--
42	0xD049	A	Called class name	--	• In a normal state: None • In an abnormal state: Name of the exception raised from UP
43	0xD04A	A	Called class name	--	--
44	0xD04B	A	Called class name	--	• In a normal state: None • In an abnormal state: Name of the exception raised from UP
45	0xD04C	A	Called class name	--	Exception, which is the argument of `ChunkListener.onError(Exception)`
46	0xD04D	A	Called class name	--	• In a normal state: None

No. in the figures#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
					• In an abnormal state: Name of the exception raised from UP

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-96 through Figure 8-99.

8.25.3 Data output to the exception log

If an exception name is output to the additional information (OPT) column at a trace based performance analysis collection point, detailed information about the exception is output to the exception log at the same time. You can analyze the failure by comparing the exception name output by the trace based performance analysis with the exception log. However, if you intentionally raise exceptions in a batch application or other application and use them to control the processing of jobs, the amount of log output might increase because detailed information is output to the exception log each time an exception is raised. When executing such an application, set a larger size for the exception log in advance.

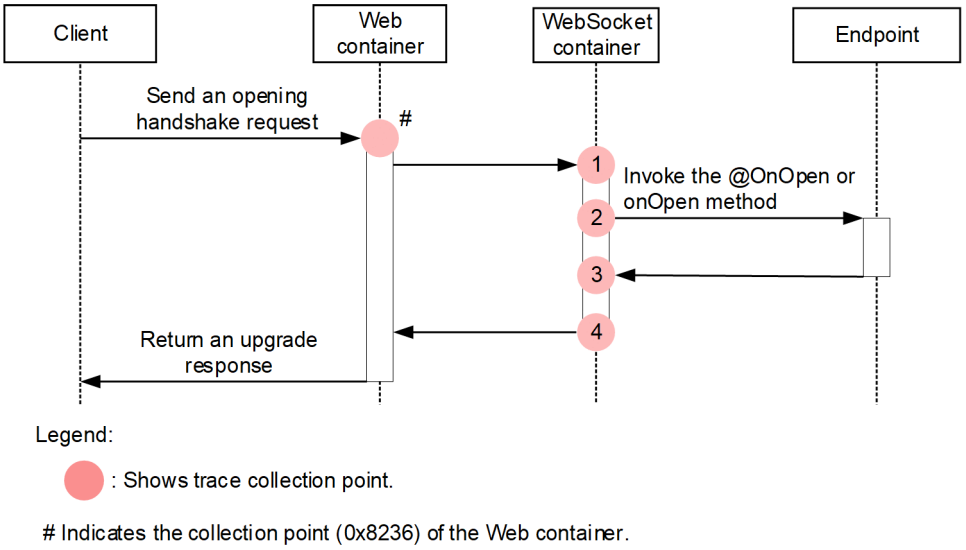
8.26 Trace collection points of WebSocket

This section provides the details of trace collection points in WebSocket.

8.26.1 When an opening handshake request is received

The following figure shows the trace collection points in WebSocket.

Figure 8–100: Trace collection points of WebSocket (when an opening handshake request is received)



The following table describes the event IDs, trace levels, trace collection points, and information that can be collected.

Table 8–135: Details of the trace collection points of WebSocket (when an opening handshake request is received)

Event ID	No. in the figures#	Level	Trace acquisition points	Information that you can acquire		
				Interface name	Operation name	Optional
0xE100	1	A	Immediately before the start of processing after receiving the opening handshake request	CONNECT	URI at the time of handshake	--
0xE120	2	A	Immediately before invoking a method annotated with `@OnOpen` or the `onOpen` method of the `javax.websocket.Endpoint` implementation class	Class name of the endpoint	Method name	--
0xE121	3	A	Immediately after the processing of the method annotated with `@OnOpen` or the processing of the `onOpen` method of the `javax.websocket.Endpoint` implementation class is complete	Class name of the endpoint	Method name	• In a normal state: None • In an abnormal state:

Event ID	No. in the figures#	Level	Trace acquisition points	Information that you can acquire		
				Interface name	Operation name	Optional
						Exception name
0xE101	4	A	Immediately after the completion of processing after receiving the opening handshake request	CONNECT	URI at the time of handshake	• In a normal state: Session ID • In an abnormal state: Exception name

Legend:

A: Standard

--: Not applicable

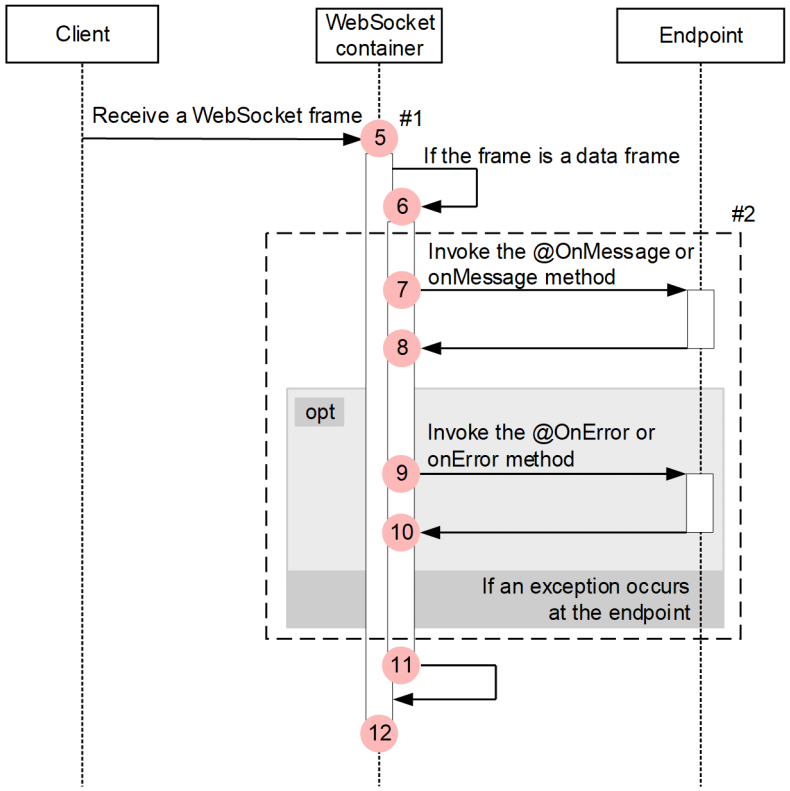
#

Corresponds to the numbers in Figure 8-100.

8.26.2 When a message is received

The following figure shows the trace collection points in WebSocket.

Figure 8–101: Trace collection points of WebSocket (when a message (data frame) is received)



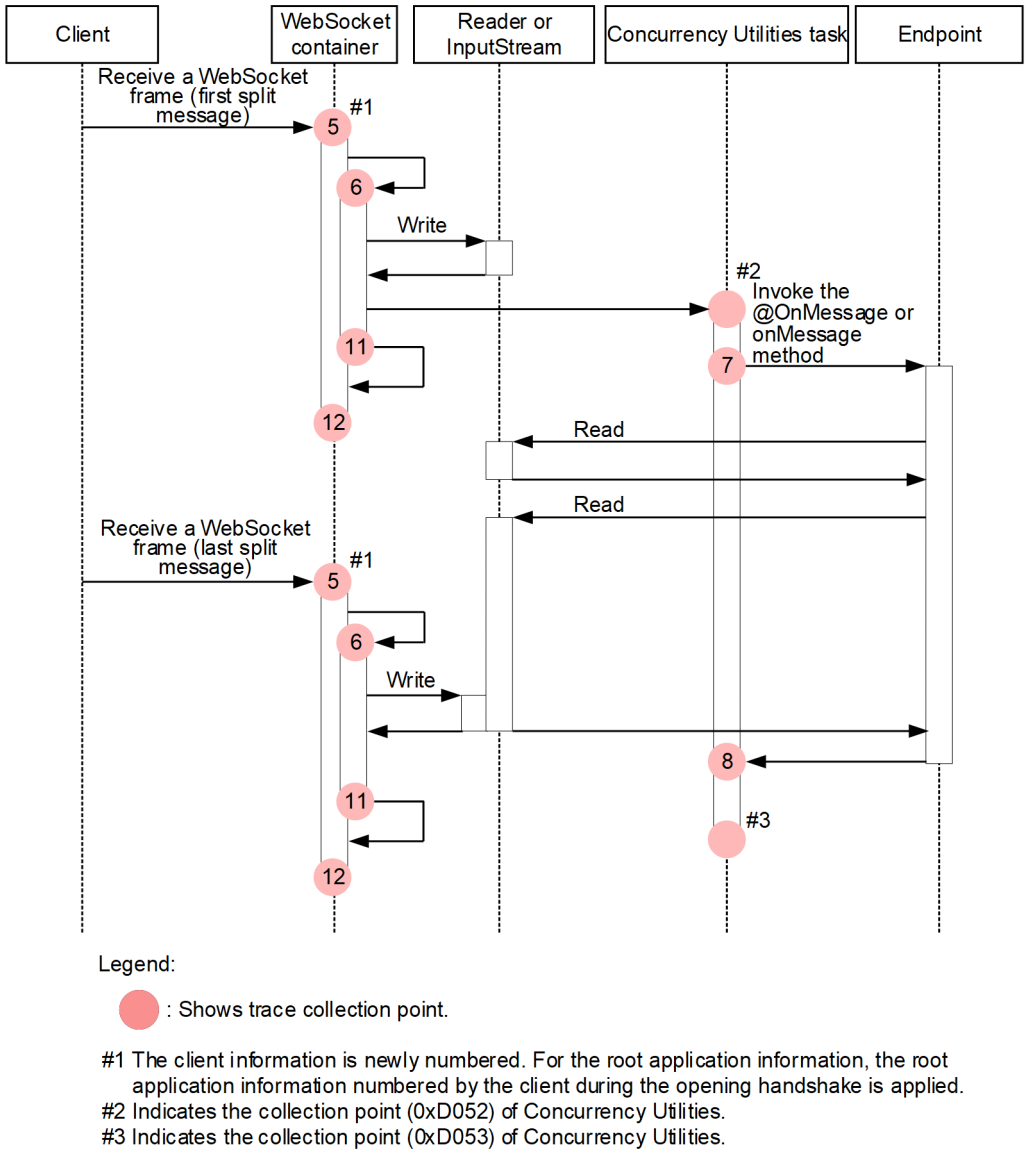
Legend:

: Shows trace collection point.

#1 The client information is newly numbered. For the root application information, the root application information numbered by the client during the opening handshake is applied.

#2 If a method with @OnMessage or the implementation class of javax.websocket.MessageHandler.Whole is used at the endpoint and a split message is received, the collection points in the frame are called only when the last frame is received.

Figure 8–102: Trace collection points of WebSocket (when a split message is received by a message handler with a Reader or InputStream type argument)



Legend:

: Shows trace collection point.

#1 The client information is newly numbered. For the root application information, the root application information numbered by the client during the opening handshake is applied.
#2 Indicates the collection point (0xD052) of Concurrency Utilities.
#3 Indicates the collection point (0xD053) of Concurrency Utilities.

The following table describes the event IDs, trace levels, trace collection points, and information that can be collected.

Table 8–136: Details of the trace collection points of WebSocket (when a message is received)

Event ID	No. in the figures#	Level	Trace acquisition points	Information that you can acquire		
				Interface name	Operation name	Optional
0xE102	5	A	Immediately before the start of processing for the received WebSocket frame	READ	--	--
0xE103	12	A	Immediately after the completion of processing for the received WebSoket frame	READ	--	• In a normal state: None • In an abnormal state:

Event ID	No. in the figures[#]	Level	Trace acquisition points	Information that you can acquire		
				Interface name	Operation name	Optional
						Exception name
0xE140	6	A	Immediately before the start of processing after receiving a data frame	MESSAGE	One of the following: • `onMessage(String)` • `onMessage(ByteBuffer)` • `onPartialMessage(String)` • `onPartialMessage(ByteBuffer)`	--
0xE141	11	A	Immediately after the completion of processing after receiving a data frame	MESSAGE	One of the following: • `onMessage(String)` • `onMessage(ByteBuffer)` • `onPartialMessage(String)` • `onPartialMessage(ByteBuffer)`	• In a normal state: None • In an abnormal state: Exception name
0xE122	7	A	Immediately before invoking a method annotated with `@OnMessage`	Class name of the endpoint	Method name	--
0xE123	8	A	Immediately after the processing of a method annotated with `@OnOpen` is complete	Class name of the endpoint	Method name	• In a normal state: None • In an abnormal state: Exception name
0xE124	7	A	Immediately before invoking the `onMessage` method of the implementation class of `javax.websocket.MessageHandler.Whole`	Implementation class name of `javax.websocket.MessageHandler.Whole`	Method name	--
0xE125	8	A	Immediately after the processing of the `onMessage` method of the implementation class of `javax.websocket.MessageHandler.Whole` is complete	Implementation class name of `javax.websocket.MessageHandler.Whole`	Method name	• In a normal state: None

Event ID	No. in the figures#	Level	Trace acquisition points	Information that you can acquire		
				Interface name	Operation name	Optional
						• In an abnormal state: Exception name
0xE126	7	A	Immediately before invoking the `onMessage` method of the implementation class of `javax.websocket.MessageHandler.Partial`	Implementation class name of `javax.websocket.MessageHandler.Partial`	Method name	--
0xE127	8	A	Immediately after the processing of the `onMessage` method of the implementation class of `javax.websocket.MessageHandler.Partial` is complete	Implementation class name of `javax.websocket.MessageHandler.Partial`	Method name	• In a normal state: None • In an abnormal state: Exception name
0xE128	9	A	Immediately before invoking a method annotated with `@OnError` or the `onError` method of the `javax.websocket.Endpoint` implementation class	Class name of the endpoint	Method name	--
0xE129	10	A	Immediately after the processing of the method annotated with `@OnError` or the processing of the `onError` method of the `javax.websocket.Endpoint` implementation class is complete	Class name of the endpoint	Method name	• In a normal state: None • In an abnormal state: Exception name

Legend:

A: Standard

--: Not applicable

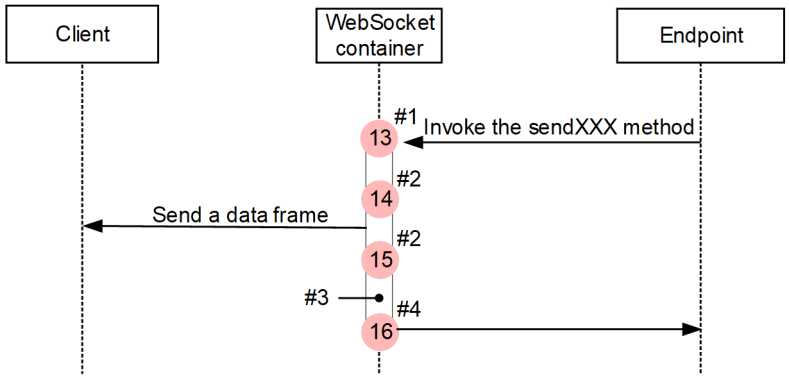
#

Corresponds to the numbers in Figure 8-101 and Figure 8-102.

8.26.3 When data is sent

The following figure shows the trace collection points in WebSocket.

Figure 8–103: Trace collection points of WebSocket (when data is sent)



Legend:

: Shows trace collection point.

#1 The client information is newly numbered. For the root application information, the root application information of the caller of the send*XXX* method is applied.
#2 For the client application information, the client application information numbered in No. 13 is applied. For the root application information, the root application information of the data frame destination is applied.
#3 The WebSocket access log is output here. For the root application information, the same value as No. 15 is applied.
#4 For the client application information, the client application information numbered in No. 13 is applied. For the root application information, the root application information of the caller of the send*XXX* method is applied.

The following table describes the event IDs, trace levels, trace collection points, and information that can be collected.

Table 8–137: Details of the trace collection points of WebSocket (when data is sent)

Event ID	No. in the figures[#]	Level	Trace acquisition points	Information that you can acquire		
				Interface name	Operation name	Optional
0xE112	14	A	Immediately before the WebSocket frame transmission processing starts	WRITE	--	--
0xE113	15	A	Immediately after the WebSocket frame transmission processing is complete	WRITE	--	• In a normal state: None • In an abnormal state: Exception name
0xE130	13	A	Immediately after the `javax.websocket.RemoteEndpoint.Async.sendXXX` method is invoked	`javax.websocket.RemoteEndpoint.Async`	Method name (argument type)	Destination session ID
0xE131	16	A	Immediately before the return of the `javax.websocket.RemoteEndpoint.Async.sendXXX` method	`javax.websocket.RemoteEndpoint.Async`	Method name (argument type)	• In a normal state: None • In an abnormal state:

Event ID	No. in the figures[#]	Level	Trace acquisition points	Information that you can acquire		
				Interface name	Operation name	Optional
						Exception name
0xE132	13	A	Immediately after the `javax.websocket.RemoteEndpoint.Basic.sendXXX` method is invoked	`javax.websocket.RemoteEndpoint.Basic`	Method name (argument type)	Destination session ID
0xE133	16	A	Immediately before the return of the `javax.websocket.RemoteEndpoint.Basic.sendXXX` method	`javax.websocket.RemoteEndpoint.Basic`	Method name (argument type)	• In a normal state: None • In an abnormal state: Exception name
0xE134	13	A	Immediately after the `javax.websocket.RemoteEndpoint.sendXXX` method is invoked	`javax.websocket.RemoteEndpoint`	Method name (argument type)	Destination session ID
0xE135	16	A	Immediately before the return of the `javax.websocket.RemoteEndpoint.sendXXX` method	`javax.websocket.RemoteEndpoint`	Method name (argument type)	• In a normal state: None • In an abnormal state: Exception name

Legend:

A: Standard

--: Not applicable

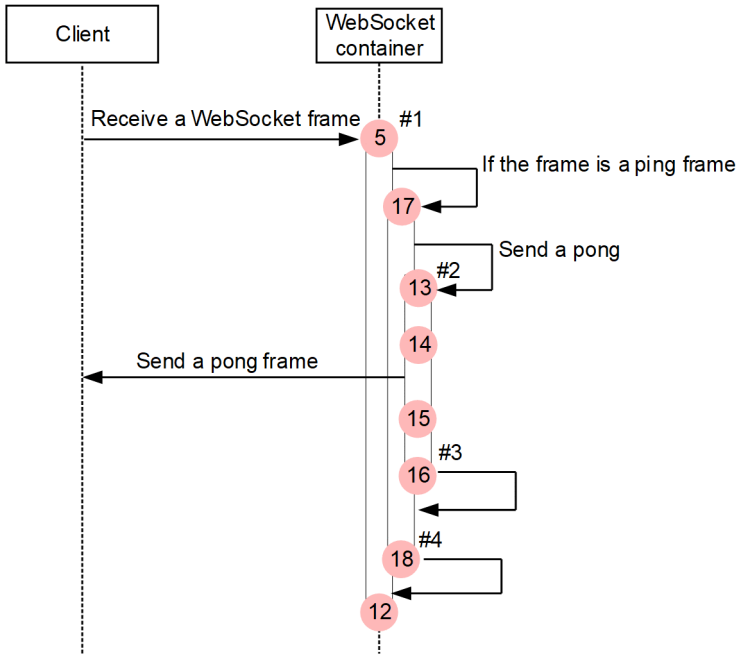
#

Corresponds to the numbers in Figure 8-103.

8.26.4 When a Ping is received

The following figure shows the trace collection points in WebSocket.

Figure 8–104: Trace collection points of WebSocket (when a Ping is received)



Legend:

: Shows trace collection point.

#1 The client information is newly numbered. For the root application information, the root application information numbered by the client during the opening handshake is applied.
#2 The client information is newly numbered. For the root application information, the same root application information as No. 5 is applied.
#3 For the client application information, the client application information numbered in No. 13 is applied. For the root application information, the same root application information as No. 5 is applied.
#4 For the client application information, the client application information numbered in No. 5 is applied. For the root application information, the same root application information as No. 5 is applied.

The following table describes the event IDs, trace levels, trace collection points, and information that can be collected.

Table 8–138: Details of the trace collection points of WebSocket (when a Ping is received)

Event ID	No. in the figures#	Level	Trace acquisition points	Information that you can acquire		
				Interface name	Operation name	Optional
0xE102	5	A	Immediately before the start of processing for the received WebSocket frame	READ	--	--
0xE103	12	A	Immediately after the completion of processing for the received WebSoket frame	READ	--	• In a normal state: None • In an abnormal state: Exception name

Event ID	No. in the figures[#]	Level	Trace acquisition points	Information that you can acquire		
				Interface name	Operation name	Optional
0xE112	14	A	Immediately before the WebSocket frame transmission processing starts	WRITE	--	--
0xE113	15	A	Immediately after the WebSocket frame transmission processing is complete	WRITE	--	• In a normal state: None • In an abnormal state: Exception name
0xE142	17	A	Immediately before the start of processing after receiving a Ping frame	PING	--	--
0xE143	18	A	Immediately after the completion of processing after receiving a Ping frame	PING	--	• In a normal state: None • In an abnormal state: Exception name
0xE130	13	A	Immediately after the `javax.websocket.RemoteEndpoint.Async.sendXXX` method is invoked	`javax.websocket.RemoteEndpoint.Async`	Method name (argument type)	Destination session ID
0xE131	16	A	Immediately before the return of the `javax.websocket.RemoteEndpoint.Async.sendXXX` method	`javax.websocket.RemoteEndpoint.Async`	Method name (argument type)	• In a normal state: None • In an abnormal state: Exception name
0xE132	13	A	Immediately after the `javax.websocket.RemoteEndpoint.Basic.sendXXX` method is invoked	`javax.websocket.RemoteEndpoint.Basic`	Method name (argument type)	Destination session ID
0xE133	16	A	Immediately before the return of the `javax.websocket.RemoteEndpoint.Basic.sendXXX` method	`javax.websocket.RemoteEndpoint.Basic`	Method name (argument type)	• In a normal state: None • In an abnormal state: Exception name
0xE134	13	A	Immediately after the `javax.websocket.RemoteEndpoint.sendXXX` method is invoked	`javax.websocket.RemoteEndpoint`	Method name (argument type)	Destination session ID

Event ID	No. in the figures#	Level	Trace acquisition points	Information that you can acquire		
				Interface name	Operation name	Optional
0xE135	16	A	Immediately before the return of the `javax.websocket.RemoteEndpoint.sendXXX` method	`javax.websocket.RemoteEndpoint`	Method name (argument type)	• In a normal state: None • In an abnormal state: Exception name

Legend:

A: Standard

--: Not applicable

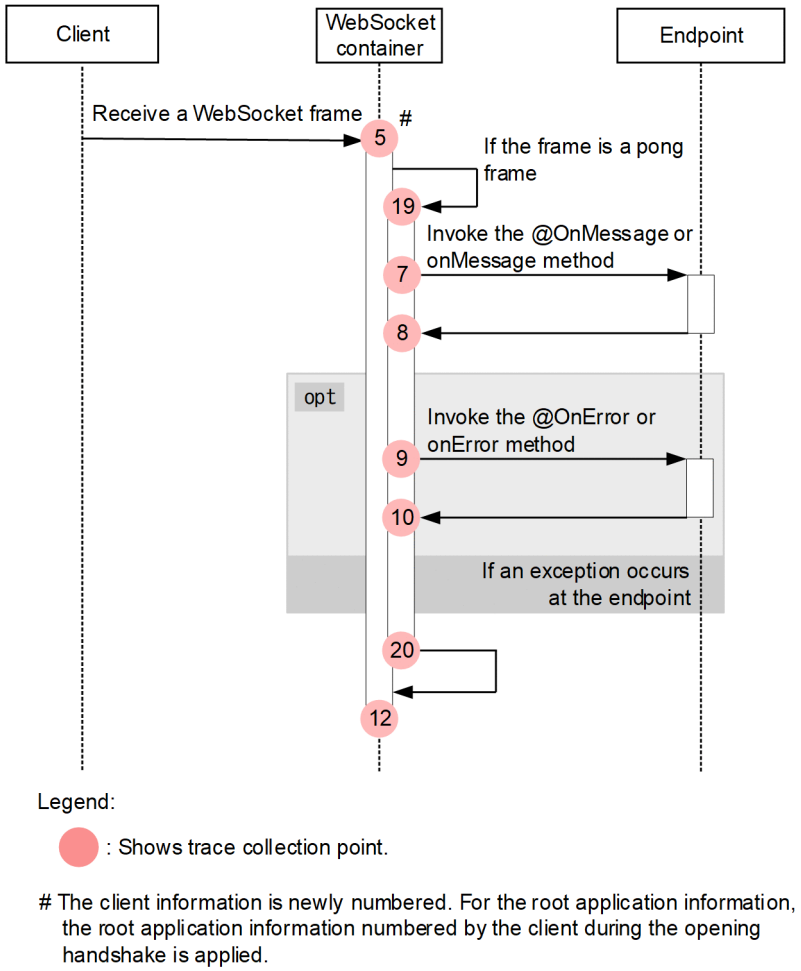
#

Corresponds to the numbers in Figure 8-104.

8.26.5 When a Pong is received

The following figure shows the trace collection points in WebSocket.

Figure 8–105: Trace collection points of WebSocket (when a Pong is received)



The following table describes the event IDs, trace levels, trace collection points, and information that can be collected.

Table 8–139: Details of the trace collection points of WebSocket (when a Pong is received)

Event ID	No. in the figures#	Level	Trace acquisition points	Information that you can acquire		
				Interface name	Operation name	Optional
0xE102	5	A	Immediately before the start of processing for the received WebSocket frame	READ	--	--
0xE103	12	A	Immediately after the completion of processing for the received WebSoket frame	READ	--	• In a normal state: None • In an abnormal state: Exception name
0xE144	19	A	Immediately before the start of processing after receiving a Pong frame	PONG	--	--
0xE145	20	A	Immediately after the completion of processing after receiving a Pong frame	PONG	--	• In a normal state:

Event ID	No. in the figures[#]	Level	Trace acquisition points	Information that you can acquire		
				Interface name	Operation name	Optional
						None • In an abnormal state: Exception name
0xE122	7	A	Immediately before invoking a method annotated with `@OnMessage`	Class name of the endpoint	Method name	--
0xE123	8	A	Immediately after the processing of a method annotated with `@OnOpen` is complete	Class name of the endpoint	Method name	• In a normal state: None • In an abnormal state: Exception name
0xE124	7	A	Immediately before invoking the `onMessage` method of the implementation class of `javax.websocket.MessageHandler.Whole`	Implementation class of `javax.websocket.MessageHandler.Whole`	Method name	--
0xE125	8	A	Immediately after the processing of the `onMessage` method of the implementation class of `javax.websocket.MessageHandler.Whole` is complete	Implementation class of `javax.websocket.MessageHandler.Whole`	Method name	• In a normal state: None • In an abnormal state: Exception name
0xE126	7	A	Immediately before invoking the `onMessage` method of the implementation class of `javax.websocket.MessageHandler.Partial`	Implementation class of `javax.websocket.MessageHandler.Partial`	Method name	--
0xE127	8	A	Immediately after the processing of the `onMessage` method of the implementation class of `javax.websocket.MessageHandler.Partial` is complete	Implementation class of `javax.websocket.MessageHandler.Partial`	Method name	• In a normal state: None • In an abnormal state: Exception name
0xE128	9	A	Immediately before invoking a method annotated with `@OnError` or the `onError` method of the `javax.websocket.Endpoint` implementation class	Class name of the endpoint	Method name	--

Event ID	No. in the figures#	Level	Trace acquisition points	Information that you can acquire		
				Interface name	Operation name	Optional
0xE129	10	A	Immediately after the processing of a method annotated with `@OnError` or of the `onError` method of the `javax.websocket.Endpoint` implementation class is complete	Class name of the endpoint	Method name	• In a normal state: None • In an abnormal state: Exception name

Legend:

A: Standard

--: Not applicable

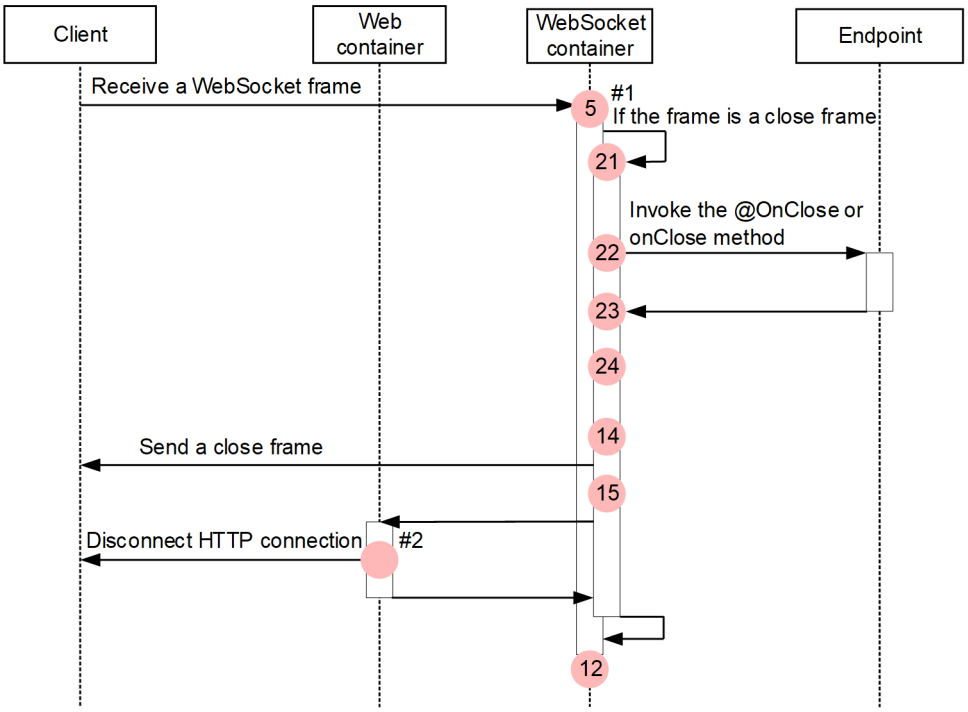
#

Corresponds to the numbers in Figure 8-105.

8.26.6 When a closing handshake request is received

The following figure shows the trace collection points in WebSocket.

Figure 8–106: Trace collection points of WebSocket (when a closing handshake request is received (with payload data))

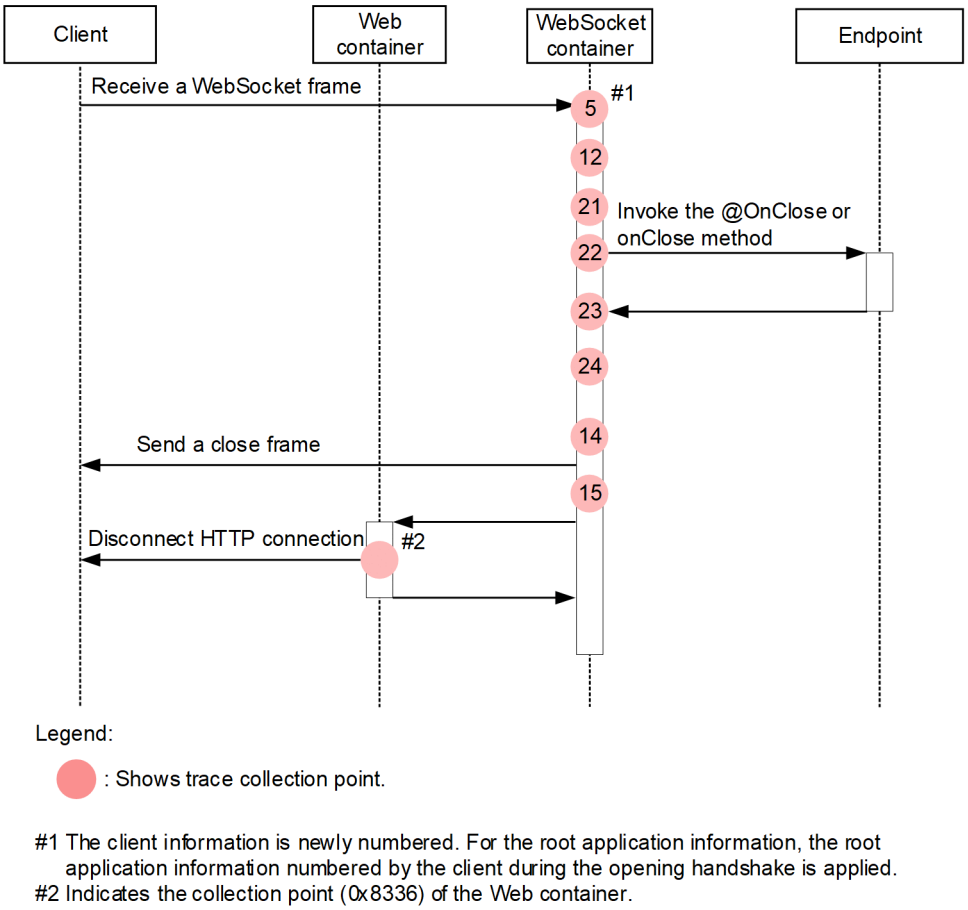


Legend:

: Shows trace collection point.

#1 The client information is newly numbered. For the root application information, the root application information numbered by the client during the opening handshake is applied.
#2 Indicates the collection point (0x8336) of the Web container.

Figure 8–107: Trace collection points of WebSocket (when a closing handshake request is received (no payload data))



#1 The client information is newly numbered. For the root application information, the root application information numbered by the client during the opening handshake is applied.
#2 Indicates the collection point (0x8336) of the Web container.

The following table describes the event IDs, trace levels, trace collection points, and information that can be collected.

Table 8–140: Details of the trace collection points of WebSocket (when a closing handshake request is received)

Event ID	No. in the figures#	Level	Operation name	Information that you can acquire		
				Interface name	Operation name	Optional
0xE102	5	A	Immediately before the start of processing for the received WebSocket frame	READ	--	--
0xE103	12	A	Immediately after the completion of processing for the received WebSoket frame	READ	--	• In a normal state: None • In an abnormal state: Exception name
0xE112	14	A	Immediately before the WebSocket frame transmission processing starts	WRITE	--	--

Event ID	No. in the figures#	Level	Operation name	Information that you can acquire		
				Interface name	Operation name	Optional
0xE113	15	A	Immediately after the WebSocket frame transmission processing is complete	WRITE	--	• In a normal state: None • In an abnormal state: Exception name
0xE146	21	A	Immediately before the start of processing after receiving or sending a closing handshake request	CLOSE	Cause of disconnection	--
0xE147	24	A	Immediately after the completion of processing after receiving or sending a closing handshake request	CLOSE	Cause of disconnection	• In a normal state: None • In an abnormal state: Exception name
0xE12A	22	A	Immediately before invoking a method annotated with `@OnClose` or the `onClose` method of the `javax.websocket.Endpoint` implementation class	Class name of the endpoint	Method name	--
0xE12B	23	A	Immediately after the processing of a method annotated with `@OnClose` or of the `onClose` method of the `javax.websocket.Endpoint` implementation class is complete	Class name of the endpoint	Method name	• In a normal state: None • In an abnormal state: Exception name

Legend:

A: Standard

--: Not applicable

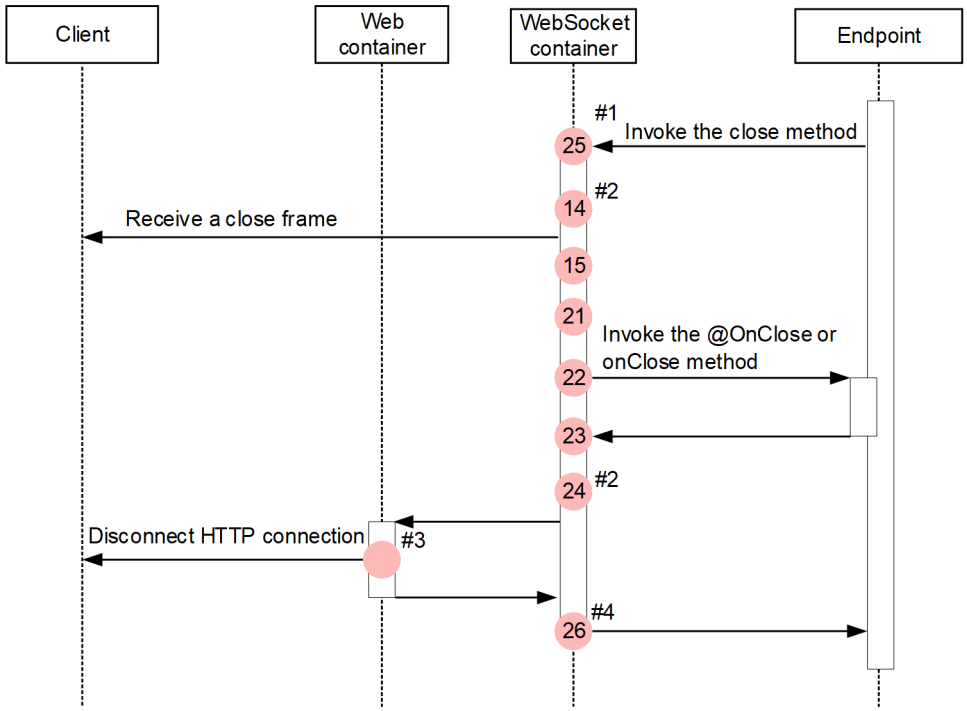
#

Corresponds to the numbers in Figure 8-106, Figure 8-107.

8.26.7 When a closing handshake request is sent

The following figure shows the trace collection points in WebSocket.

Figure 8–108: Trace collection points of WebSocket (when a closing handshake request is sent)



Legend:

: Shows trace collection point.

#1 The client information is newly numbered. For the root application information, the root application information of the caller of the send*XXX* method is applied.
#2 For the client application information, the client application information numbered in No. 25 is applied. For the root application information, the root application information of the data frame destination is applied.
#3 Indicates the collection point (0x8336) of the Web container.
#4 For the client application information, the client application information numbered in No. 25 is applied. For the root application information, the root application information of the caller of the send*XXX* method is applied.

The following table describes the event IDs, trace levels, trace collection points, and information that can be collected.

Table 8–141: Details of the trace collection points of WebSocket (when a closing handshake request is sent)

Event ID	No. in the figures#	Level	Trace acquisition points	Information that you can acquire		
				Interface name	Operation name	Optional
0xE112	14	A	Immediately before the WebSocket frame transmission processing starts	WRITE	--	--
0xE113	15	A	Immediately after the WebSocket frame transmission processing is complete	WRITE	--	• In a normal state: None • In an abnormal state: Exception name

Event ID	No. in the figures[#]	Level	Trace acquisition points	Information that you can acquire		
				Interface name	Operation name	Optional
0xE146	21	A	Immediately before the start of processing after receiving or sending a closing handshake request	CLOSE	Cause of disconnection	--
0xE147	24	A	Immediately after the completion of processing after receiving or sending a closing handshake request	CLOSE	Cause of disconnection	• In a normal state: None • In an abnormal state: Exception name
0xE12A	22	A	Immediately before invoking a method annotated with `@OnClose` or the `onClose` method of the `javax.websocket.Endpoint` implementation class	Class name of the endpoint	Method name	--
0xE12B	23	A	Immediately after the processing of a method annotated with `@OnClose` or of the `onClose` method of the `javax.websocket.Endpoint` implementation class is complete	Class name of the endpoint	Method name	• In a normal state: None • In an abnormal state: Exception name
0xE136	25	A	Immediately after the `javax.websocket.Session.close` method is invoked	`javax.websocket.Session`	Method name (argument type)	Destination session ID
0xE137	26	A	Immediately before the return of the `javax.websocket.Session.close` method	`javax.websocket.Session`	Method name (argument type)	• In a normal state: None • In an abnormal state: Exception name

Legend:

A: Standard

--: Not applicable

#

Corresponds to the numbers in Figure 8-108.

8.27 Trace collection points of Concurrency Utilities

This section describes the trace collection points of Concurrency Utilities and the trace information that can be collected.

8.27.1 Trace collection points and trace information that can be collected

The following table describes the event IDs, trace collection points, and PRF trace collection levels.

Table 8–142: Details of trace collection points in Concurrency Utilities

Event ID	No. in the figures#	Trace acquisition points	Level
0xD050	1	Entry point of the new task registration method	A
0xD051	2	Exit point of the new task registration method	A
0xD052	3	Immediately before calling a task	A
0xD053	4	Immediately after the task ends	A
0xD054	5	Entry point of the `get` method of the `Future` or `ScheduledFuture` object returned as a return value upon task registration	A
0xD055	6	Exit point of the `get` method of the `Future` or `ScheduledFuture` object returned as a return value upon task registration	A
0xD056	7	Entry point of the `cancel` method of the `Future` or `ScheduledFuture` object returned as a return value upon task registration	A
0xD057	8	Exit point of the `cancel` method of the `Future` or `ScheduledFuture` object returned as a return value upon task registration	A

Legend:

A: Standard

#

Corresponds to the numbers in Figure 8-109 through Figure 8-116.

Figure 8-109 shows the trace collection points when the following methods are called:

- `javax.enterprise.concurrent.ManagedExecutorService#execute(Runnable)`
- `javax.enterprise.concurrent.ManagedScheduledExecutorService#execute(Runnable)`

Figure 8–109: Trace collection points of Concurrency Utilities (1)

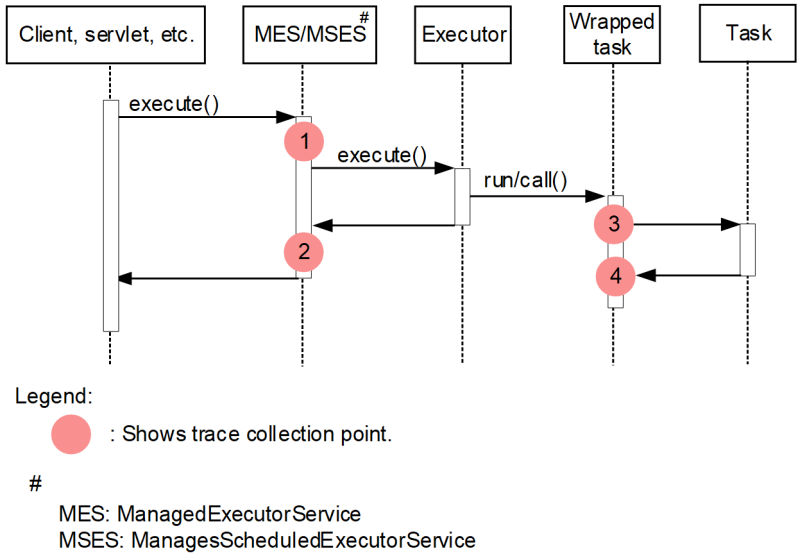


Figure 8-110 through Figure 8-114 show the trace collection points when the following methods are called. The trace collection points differ depending on the timing of calling the `get` method or the `cancel` method.

When the `get` method is called before the task finishes: Figure 8-110

When the `get` method is called after the task finishes: Figure 8-111

When the `cancel` method is called before the task execution starts: Figure 8-112

When the `cancel` method is called during task processing: Figure 8-113

When the `cancel` method is called after the task finishes: Figure 8-114

- `javax.enterprise.concurrent.ManagedExecutorService#submit(Runnable)`
- `javax.enterprise.concurrent.ManagedExecutorService#submit(Runnable, T)`
- `javax.enterprise.concurrent.ManagedExecutorService#submit(Callable<T>)`
- `javax.enterprise.concurrent.ManagedScheduledExecutorService#submit(Runnable)`
- `javax.enterprise.concurrent.ManagedScheduledExecutorService#submit(Runnable, T)`
- `javax.enterprise.concurrent.ManagedScheduledExecutorService#submit(Callable<T>)`
- `javax.enterprise.concurrent.ManagedScheduledExecutorService#schedule(Callable<V>, long, TimeUnit)`
- `javax.enterprise.concurrent.ManagedScheduledExecutorService#schedule(Callable<V>, Trigger)`
- `javax.enterprise.concurrent.ManagedScheduledExecutorService#schedule(Runnable, long, TimeUnit)`
- `javax.enterprise.concurrent.ManagedScheduledExecutorService#schedule(Runnable, Trigger)`
- `javax.enterprise.concurrent.ManagedScheduledExecutorService#scheduleAtFixedRate(Runnable, long, long, TimeUnit)`

- `javax.enterprise.concurrent.ManagedScheduledExecutorService#scheduleWithFixedDelay(Runnable, long, long, TimeUnit)`

Figure 8–110: Trace collection points of Concurrency Utilities (2) (when the get method is called before the task finishes)

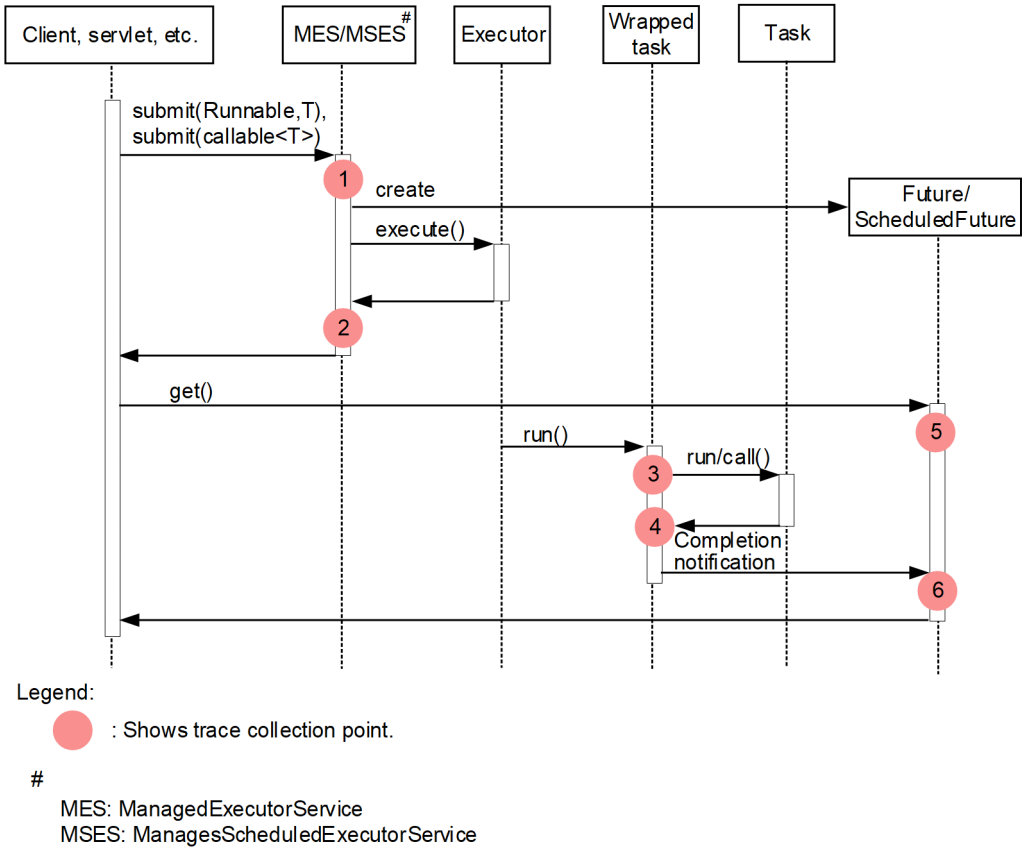
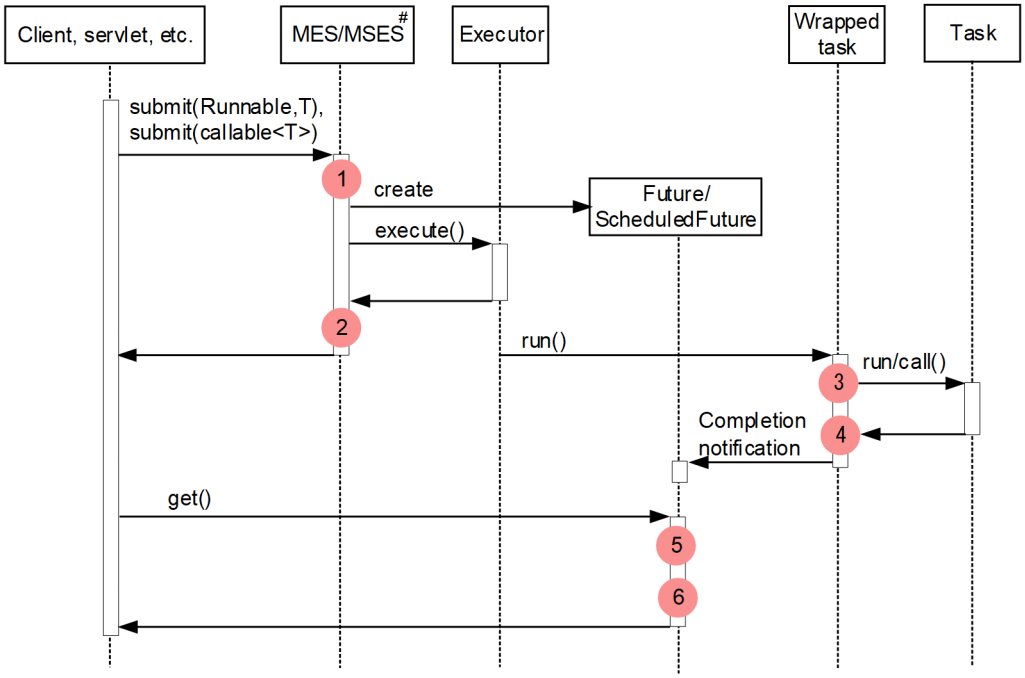


Figure 8–111: Trace collection points of Concurrency Utilities (3) (when the get method is called after the task finishes)



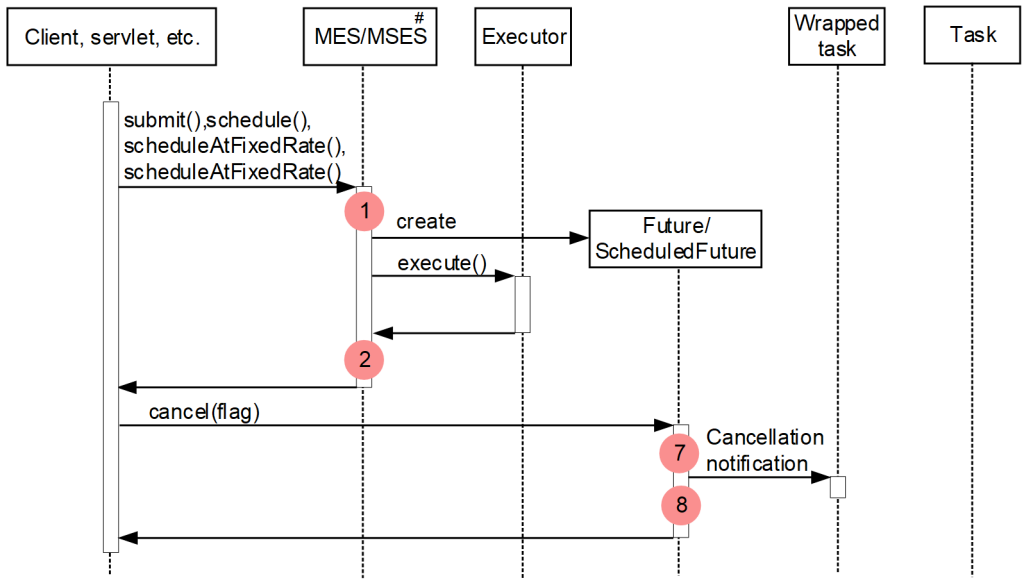
Legend:

: Shows trace collection point.

#

MES: ManagedExecutorService
MSES: ManagesScheduledExecutorService

Figure 8–112: Trace collection points of Concurrency Utilities (4) (when the cancel method is called before the task execution starts)



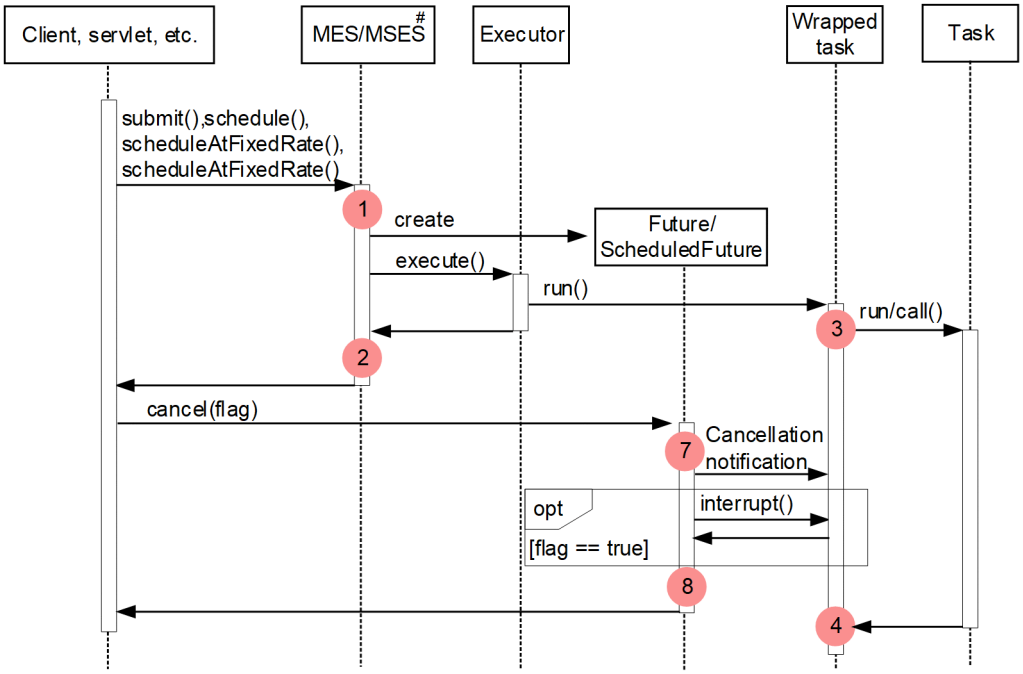
Legend:

: Shows trace collection point.

#

MES: ManagedExecutorService
MSES: ManagesScheduledExecutorService

Figure 8–113: Trace collection points of Concurrency Utilities (5) (when the cancel method is called during task processing)



Legend:

: Shows trace collection point.

#

MES: ManagedExecutorService
MSES: ManagesScheduledExecutorService

Figure 8–114: Trace collection points of Concurrency Utilities (6) (when the cancel method is called after the task finishes)

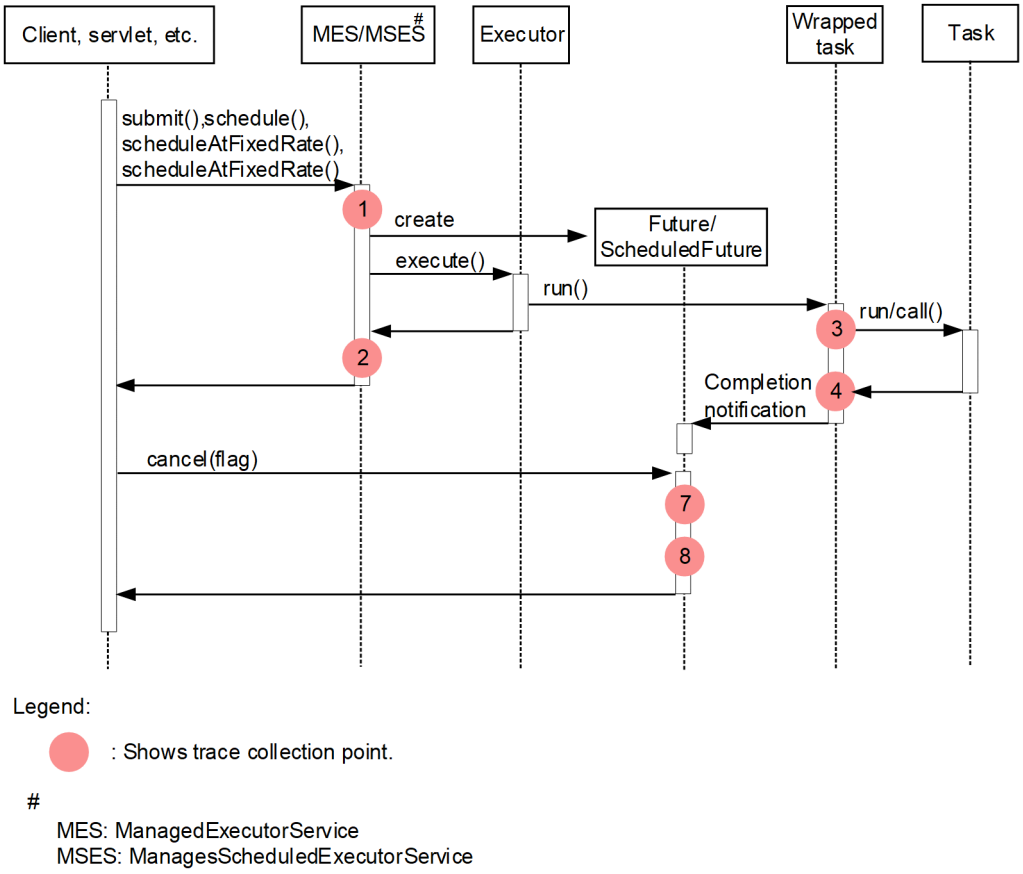


Figure 8-115 shows the trace collection points when the following methods are called:

- `javax.enterprise.concurrent.ManagedExecutorService#invokeAll(Collection<? Extends Callable<T>>)`
- `javax.enterprise.concurrent.ManagedExecutorService#invokeAll(Collection<? Extends Callable<T>>, long, TimeUnit)`
- `javax.enterprise.concurrent.ManagedScheduledExecutorService#invokeAll(Collection<? Extends Callable<T>>)`
- `javax.enterprise.concurrent.ManagedScheduledExecutorService#invokeAll(Collection<? Extends Callable<T>>, long, TimeUnit)`

Figure 8–115: Trace collection points of Concurrency Utilities (7)

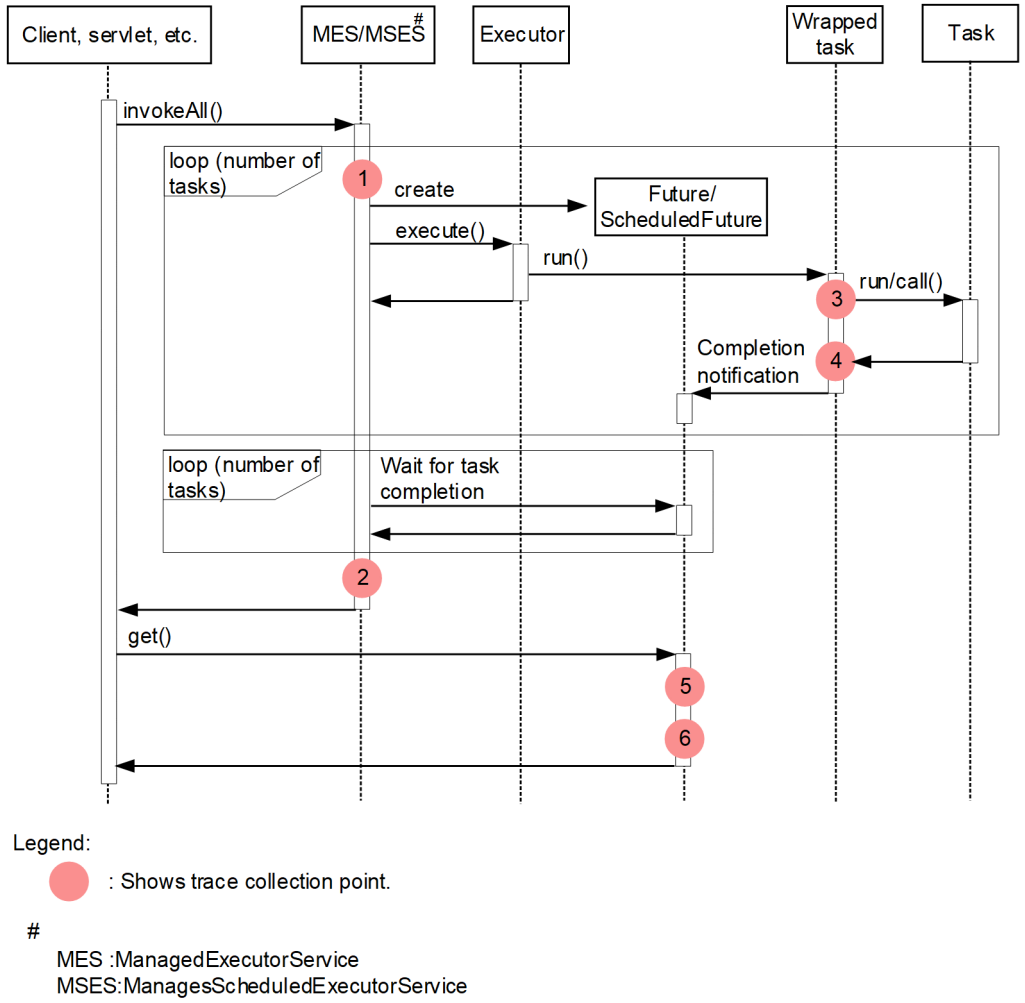
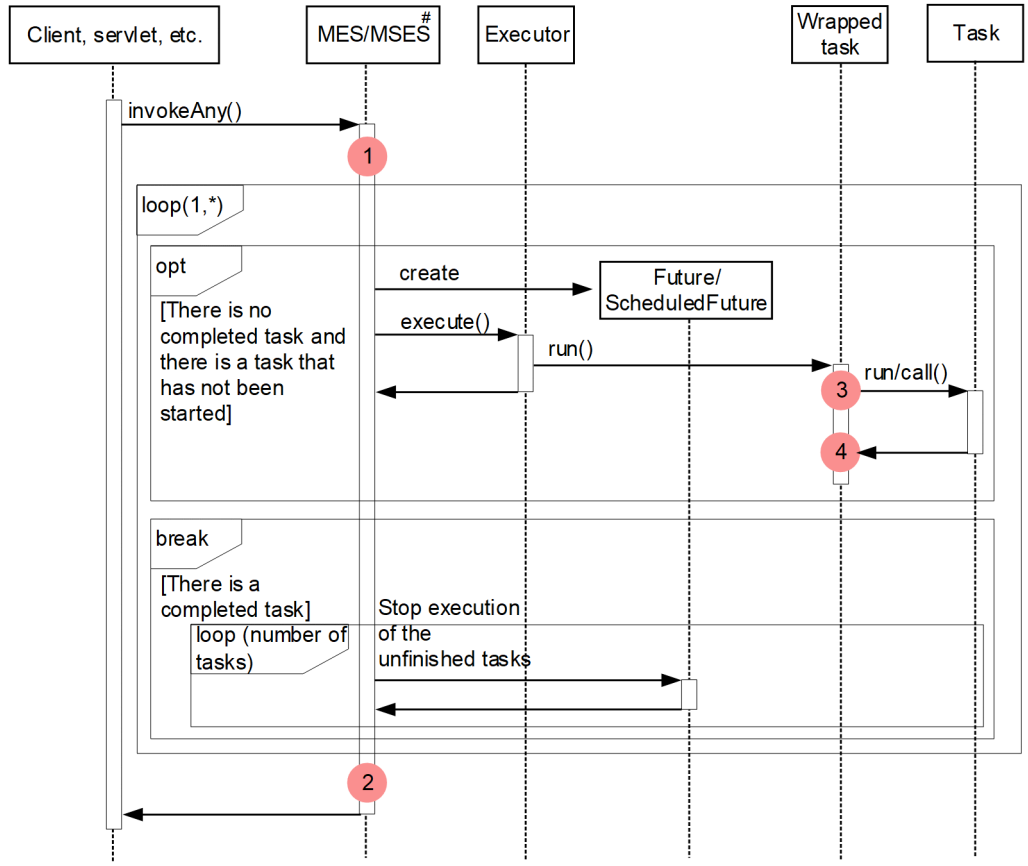


Figure 8-116 shows the trace collection points when the following methods are called:

- `javax.enterprise.concurrent.ManagedExecutorService#invokeAny(Collection<? Extends Callable<T>>)`
- `javax.enterprise.concurrent.ManagedExecutorService#invokeAny(Collection<? Extends Callable<T>>, long, TimeUnit)`
- `javax.enterprise.concurrent.ManagedScheduledExecutorService#invokeAny(Collection<? Extends Callable<T>>)`
- `javax.enterprise.concurrent.ManagedScheduledExecutorService#invokeAny(Collection<? Extends Callable<T>>, long, TimeUnit)`

Figure 8–116: Trace collection points of Concurrency Utilities (8)



8.27.2 Trace information that can be collected

The following table describes the trace information that can be collected in Concurrency Utilities.

Table 8–143: Trace information that can be collected at the JMS Connection interface

No. in the figures#	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
1	0xD050	A	JNDI name of the registration destination `ManagedExecutorService` or `ManagedScheduledExecutorService`	Name of the invoked method (For a method for which multiple tasks can be specified#2, the number of tasks is output in parentheses.)	Class name of the new task (If there are multiple tasks, the class names are separated by commas.)
2	0xD051	A	JNDI name of the registration destination `ManagedExecutorService` or `ManagedScheduledExecutorService`	Task ID	• In a normal state: None • In an abnormal state:

No. in the figures[#]	Event ID	Level	Information that you can acquire		
			Interface name	Operation name	Optional
					Exception name
3	0xD052	A	JNDI name of the calling `ManagedExecutorService` or `ManagedScheduledExecutorService`	• For a single task Task ID • For multiple tasks Task ID-List serial number	--
4	0xD053	A	JNDI name of the calling `ManagedExecutorService` or `ManagedScheduledExecutorService`	• For a single task Task ID • For multiple tasks Task ID-List serial number	• In a normal state: None • In an abnormal state: Exception name
5	0xD054	A	JNDI name of the registration destination `ManagedExecutorService` or `ManagedScheduledExecutorService`	• For a single task Task ID • For multiple tasks Task ID-List number	--
6	0xD055	A	JNDI name of the registration destination `ManagedExecutorService` or `ManagedScheduledExecutorService`	• For a single task Task ID • For multiple tasks Task ID-List number	• In a normal state: None • In an abnormal state: Exception name
7	0xD056	A	JNDI name of the registration destination `ManagedExecutorService` or `ManagedScheduledExecutorService`	• For a single task Task ID • For multiple tasks Task ID-List number	--
8	0xD057	A	JNDI name of the registration destination `ManagedExecutorService` or `ManagedScheduledExecutorService`	• In a normal state: Return value of the `cancel` method • In an abnormal state: None	• In a normal state: None • In an abnormal state: Exception name

Legend:

A: Standard

--: Not applicable

#1

Corresponds to the numbers in Figure 8-109 through Figure 8-116.

#2

A method for which multiple tasks can be specified is one that takes a Collection object as its first argument. The size of the Collection object that is the actual argument is output in parentheses.

Example: When a Collection object of three elements is passed by the `invokeAny` method

```
invokeAny(3)
```

If the Collection object is null, `null` is output in parentheses.

Example: When the Collection object is null

```
invokeAny(null)
```

9 Product JavaVM Functionality

The product JavaVM is JavaVM that is provided in Application Server. You can use the product JavaVM for acquiring the data at error time and the information used for tuning. This chapter describes about the functionality provided by the product JavaVM. For details about the Explicit Memory Management functionality provided by the product JavaVM, see *7. Suppression of Full GC by Using the Explicit Memory Management Functionality* in the *uCosminexus Application Server Expansion Guide*.

9.1 Organization of this chapter

The product JavaVM is provided by the component software, Cosminexus Developer's Kit for Java. The process of the J2EE server or the batch server running in the Application Server is executed on the JavaVM. This section explains the functionality of the product JavaVM.

For an overview of the product JavaVM functionality, see *9.2 Overview of the product JavaVM functionality*.

The following table lists the product JavaVM functionality provided in the Application Server and corresponding reference section.

Table 9–1: product JavaVM functionality provided in the Application Server

Functionality name	Reference location
Class-wise statistical functionality	*9.3*
Instance statistical functionality#	*9.4*
STATIC member statistical functionality#	*9.5*
Reference-related information output functionality#	*9.6*
Pre-statistical GC selection functionality#	*9.7*
Unused objects statistical functionality in the Tenured area#	*9.8*
Base object list output functionality for Tenured augmentation factors#	*9.9*
Class-wise statistical information analysis functionality	*9.10*
Tenuring distribution information output functionality of the Survivor area	*9.11*
`hndlwrap` functionality	*9.12*
Functionality to set the upper limit of allocation size of C heap during JIT compilation	*9.13*
Functionality to set the upper limit of number of threads	*9.14*
Notes on using the product JavaVM functionality	*9.15*

#

One of the class-wise statistical functionality that outputs the class-wise statistical information.

9.2 Overview of the product JavaVM functionality

The process of the J2EE server or the batch server running in the Application Server is executed on the JavaVM. The functionality provided by the product JavaVM is as follows:

- Explicit Memory Management functionality
- Class-wise statistical functionality
  - Instance statistical functionality
  - STATIC member statistical functionality
  - Reference-related information output functionality
  - Pre-statistical GC selection functionality
  - Unused objects statistical functionality in the Tenured area
  - Base object list output functionality for Tenured augmentation factors
- Class-wise statistical information analysis functionality#
- Tenuring distribution output functionality of the Survivor area
- `hndlwrap` functionality
- Functionality to set the upper limit of allocation size of C heap during JIT compilation
- Functionality to set the upper limit of the number of threads

#

If using the class-wise statistical information analysis functionality, you can output the class-wise statistical information to the extended thread dump file in the CSV format.

Tip

When you execute the unused objects statistical functionality in the Tenured of the class-wise statistical functionality, the following functionality are disabled:

- Instance statistical functionality
- STATIC member statistical functionality
- Pre-statistical GC selection functionality

In the product JavaVM, the output contents of the log are expanded so that the output contents can be used for the cause analysis of the error occurrence and for checking the system state. This log is output to the Hitachi JavaVM log file, and therefore, you can acquire many troubleshoot information than the standard JavaVM. You can improve the system availability by implementing an appropriate tuning using this log (extended `verbose` information). For details about the Hitachi JavaVM log files, see *5.7 JavaVM log (JavaVM log file)*. For details about the tuning of the Java VM, see *7. JavaVM Memory Tuning* in the *uCosminexus Application Server System Design Guide*.

Hereafter, the following section describes each functionality of the product JavaVM. For details about the Explicit Memory Management functionality, see *7. Suppression of Full GC by Using the Explicit Memory Management Functionality* in the *uCosminexus Application Server Expansion Guide*.

9.3 Class-wise statistical functionality

This section describes the class-wise statistical functionality.

You can use the class-wise statistical functionality to output the instance information of the reference-related class for each class, to the extended thread dump.

The following table describes the organization of this section.

Table 9–2: Organization of this section (class-wise statistical functionality)

Category	Title	Reference
Explanation	Overview of the class-wise statistical functionality	*9.3.1*
	Functionality that requires the class-wise statistical functionality	*9.3.2*
Operation	Outputting statistic information for each class	*9.3.3*
Notes	Precautions to output the class-wise statistical information	*9.3.4*

Note:

The function-specific description is not available for "Implementation" and "Settings".

9.3.1 Overview of the class-wise statistical functionality

You can output the size of all instances those are under the members of each class that has instances, to the extended thread dump, as the statistical information for each class. This statistical information is called as *class-wise statistical information*. You can output the class-wise statistical information multiple times for investigating the change in the Java object by GC, the status of the Java object having short life, change in the size of each class, and the Java object parent-child relation. You can use this information for measuring the memory used for one transaction and for investigating the memory leak.

The class-wise statistical information is output based on the following functionality:

- Instance statistical functionality
- STATIC member statistical functionality
- Reference-related information output functionality
- Pre-statistical GC selection functionality
- Unused objects statistical functionality in the Tenured area
- Base object list output functionality for Tenured augmentation factors

For details about each functionality, see *9.3.2 Functionality that requires the class-wise statistical functionality*.

In the class-wise statistical functionality, the instances in the Java heap (a combination of the Eden area, the Survivor area, and the Tenured area) are subject to statistics. Additionally, when using Explicit Memory Management functionality, you can also target the instances that exists the in Explicit heap for statistics. For details about how to output a class-wise statistical information, see *9.3.3 Outputting Statistic Information for Each Class*.

Reference note

The extended thread dump is set such as to be output by default. For details about the settings for acquiring the extended thread dump, see *3.3.17(1) Settings for Acquiring Thread Dumps of JavaVM*. For details about an output information, see *5.5 JavaVM Thread Dump*.

Also, you can output the class-wise statistical information in the CSV format. For details about how to output the class-wise statistical information in the CSV format, see *9.10 Class-wise statistical information analysis functionality*.

9.3.2 Functionality that requires the class-wise statistical functionality

The following table lists the types and the overview of the functionality that requires the class-wise statistical functionality.

Table 9–3: Types and overview of the functionality that requires the class-wise statistical functionality

Types	Overview	Reference
Instance statistical functionality	Outputs the total size of instances that have instances of class as members, for each class.	*9.4*
STATIC member statistical functionality	Outputs the total size of the instances of the static members for each class.	*9.5*
Reference-related information output functionality	Outputs the reference relation of the class that has a specified class (instance) as member.	*9.6*
Pre-statistical GC selection functionality	You can select whether to execute the GC before a class-wise statistical information is output. Use the options to specify this functionality when executing the `jheapprof` command. In default setting, the Full GC is executed before a class-wise statistical information is output.	*9.7*
Unused objects statistical functionality in the Tenured area	Identifies the unused objects in the Tenured area.	*9.8*
Base object list output functionality for Tenured augmentation factors	Outputs the information about objects acting as the base of the unused objects that are identified using the unused object statistical functionality in the Tenured area.	*9.9*

Among the above functionalities, an instance statistical functionality, the STATIC member statistical functionality, reference-related information output functionality, and the base object list output functionality for Tenured augmentation factors are enabled when executing the class-wise statistical functionality.

You can set the pre-statistical GC selection functionality when executing the class-wise statistical functionality. Select this functionality as per the purpose of investigating the class-wise statistical information. For details, see *9.7.2 Guidelines for selecting the GC*.

9.3.3 Outputting Statistic Information for Each Class

This subsection describes how to output the class-wise statistical information.

Use the jheapprof commands to output the class-wise statistical information to the extended thread dump. Specify the Java process, where you want to output the class-wise statistical information, and the class, where you want to output the reference-related information, and then execute the jheapprof commands.

You can specify the following when executing the jheapprof command:

- Specify whether to output the information of an Explicit heap as class-wise statistical information.
- Specify whether to execute the GC before acquiring the class-wise statistical information.

The execution format and the execution example of the jheapprof command and each specification method are described as follows:

(1) Execution format and execution example of the jheapprof command

The execution format and example of the jheapprof command are described below: For details about the jheapprof commands, see *jheapprof(Output of extended thread dump containing Hitachi class-wise statistical information)* in the *uCosminexus Application Server Command Reference Guide*.

Execution format

In Windows

```
jheapprof [-f|-i] [-explicit|-noexplicit] [-class class-name] [-fullgc|
-copygc|-nogc] [-garbage|-nogarbage] [-rootobjectinfo|-norootobjectinfo
] [-rootobjectinfost size] -p process-ID
```

In UNIX

```
jheapprof [-f|-i] [-explicit|-noexplicit] [-class class-name] [-fullgc|
-copygc|-nogc] [-garbage|-nogarbage] [-rootobjectinfo|-norootobjectinfo
] [-rootobjectinfost size] [-force] -p process-ID
```

Execution example

Here, the class-wise statistical information of Java process with process ID 2463 is output.

1. In the -p option, specify the process ID of the Java process where you want to output the class-wise statistical information, and then execute the jheapprof command.

```
% jheapprof -p 2463
```

When the -f option is being omitted in the jheapprof command, the following confirmation message is displayed:

In Windows

The confirmation message whether to output an extended thread dump with Hitachi class-wise statistical information is displayed in the following format:

```
Force VM to output HitachiJavaHeapProfile: ? (y/n)
```

In UNIX

The confirmation message of process ID is displayed in the following format:

```
send SIGQUIT to 2463: ? (y/n)
```

2. Enter y.

An extended thread dump with Hitachi class-wise statistics is output. The following message is output in the running java program:

```
Writing Java core to javacore2463.030806215140.txt... OK
```

The running java program creates an extended thread dump with Hitachi class-wise statistics (`javacore.`*process-ID*`.`*date-time*`.txt`) in the current directory and continues the program.

(2) When the information of an Explicit heap is output to the class-wise statistical information

If the following conditions are satisfied, you can output the information of an Explicit heap to the class-wise statistical information:

- `-XX:+HitachiUseExplicitMemory` is specified in the JavaVM start option.
- Explicit heap is used for implementing the application, or setting the execution environment (J2EE server).

Specify the `-explicit` option in the `jheapprof` command, and then execute the command to output the information of an Explicit heap to the class-wise statistical information.

For details about the Explicit Memory Management functionality, see *7. Suppression of Full GC by Using the Explicit Memory Management Functionality* in the *uCosminexus Application Server Expansion Guide*.

(3) When specifying whether to execute the GC

You can select whether to execute the GC before the class-wise statistical information is output. This functionality is called as *pre-statistical GC selection functionality*. Specify any of the following options in the `jheapprof` command, if you want to execute the GC before the class-wise statistical information is output:

- `-fullgc`
  Executes the Full GC, and then outputs the class-wise statistical information.
- `-copygc`
  Executes the copy GC, and then outputs the class-wise statistical information.
- `-nogc`
  Outputs the class-wise statistical information without executing the GC.

For details about the pre-statistical GC selection functionality, see *9.7 Pre-statistical GC selection functionality*. Note that you cannot execute the pre-statistical GC selection functionality, when you execute the unused objects statistical functionality in the Tenured area.

9.3.4 Precautions to output the class-wise statistical information

The precautions to be taken to output the class-wise statistical information is described as follows:

- For running Java process, if you execute the `jheapprof` command in which the `-copygc` option is set, you can try to execute the pre-statistical copy GC. In this case, if a free space of the Tenured area is insufficient, the copy GC might not be executed.
  When you cannot execute the copy GC, the extended `verbosegc` information when GC occurred is not output, even if you specify `-XX:+HitachiVerboseGC` in the JavaVM start option. Furthermore, an extended thread dump that contains the class-wise statistical information is output simultaneously when executing the GC.

- For Java process where `-XX:+PrintGCDetails` is specified when starting the JavaVM, you can execute the `jheapprof` command in which `-copygc` option is set, for executing the pre-statistical copy GC. In this case, a 'Full GC' is output in the GC that is output by specifying `-XX:+PrintGCDetails`.

9.4 Instance statistical functionality

This section describes the instance statistical functionality.

The instance statistical functionality is the functionality that outputs the class-wise statistical information. You can output an instance count of the class and the total size of the instances for each class.

The instance statistical functionality uses the `jheapprof` commands for output. For details about the execution format and the execution examples of the `jheapprof` commands, see *9.3.3 Outputting Statistic Information for Each Class*.

The following table describes the organization of this section.

Table 9–4: Organization of this section (Instance statistical functionality)

Category	Title	Reference
Description	Overview of the instance statistical functionality	*9.4.1*
	Class-wise statistical information output by the instance statistical functionality	*9.4.2*

Note:

The function-specific explanation is not available for "Implementation", "Settings", "Operations", and "Notes".

9.4.1 Overview of the instance statistical functionality

The instance statistical functionality is used to check memory leaks in applications.

The instance statistical functionality investigates the reference relation of instances, such as relation with instances of class A->member variable of class A(the class is classB)->instances of classB->..., and recurrently adds the size of the instances without referencing to other instances in the class that has the instances as the member. In other words, the instance statistical functionality outputs the total size of the instances referenced by the class instances.

To check the cause of memory leak, execute the instance statistical functionality as follows, before and after the application processing for which you want to check the memory leak, then take the difference between the number of instances and the total size of the instances in points 1 and 3, check the amount of the increase in the numbers, and then identify the classes that are causing the memory leak.

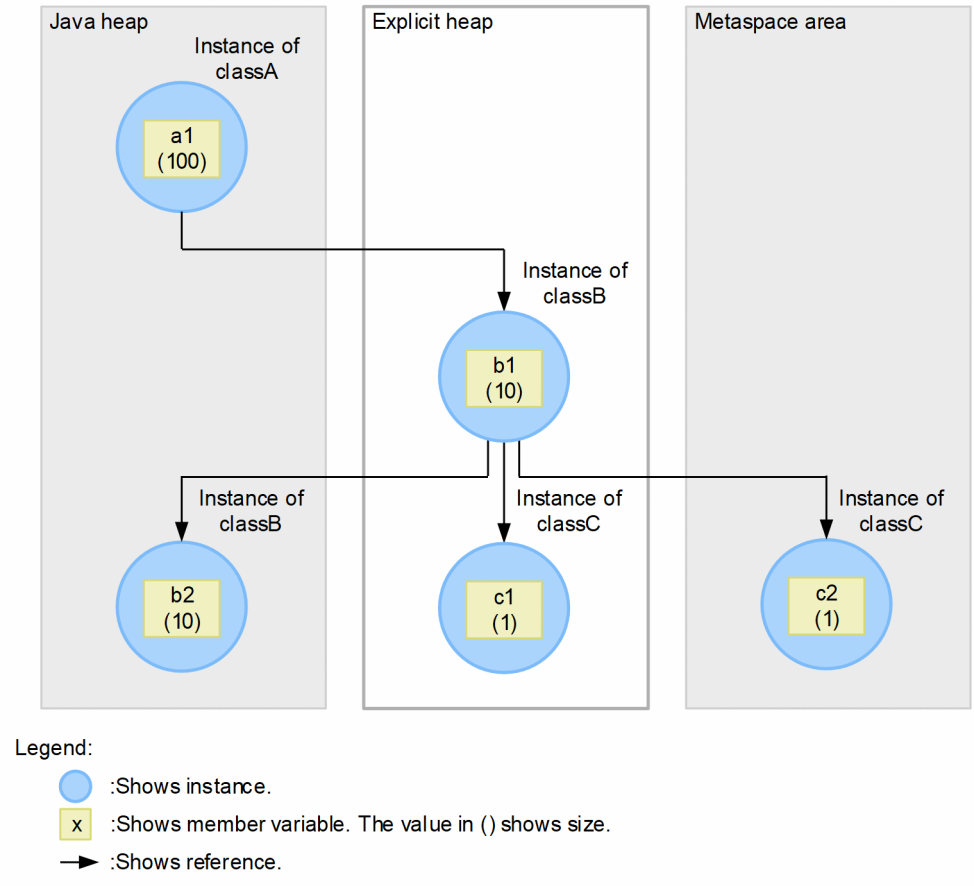
1. Execute the instance statistical functionality.
2. Execute the application processing for which you want to check the memory leak.
3. Execute the instance statistical functionality.

Note that because the instance statistical functionality recursively adds the size of all the instances referenced by each class instance, you cannot check the size of only the instances of each class (the size excludes the size of the instances being referenced).

In the instance statistical functionality, when using the Explicit Memory Management functionality, the output result of the class-wise statistical information changes according to whether to set the instances of the Explicit heap as a target for statistics. Furthermore, specify the `-explicit` option in the `jheapprof` command, and then execute the same command to set the instances of an Explicit heap as a target for statistics.

The following figure shows an example of an instance structure that describes the output result of each statistical target.

Figure 9–1: Example of instance structure



When the statistical target contains an Explicit heap, the instances a1, b1, b2, c1, and c2 becomes the target for the class-wise statistical information. When the statistical target does not contains an Explicit heap, the instances a1, b2, and c2 becomes the target for the class-wise statistical information.

The following table lists the instance count and the total size of instances for each statistical target.

Table 9–5: Instance count and total size of instances for each statistical target

Arguments of the jheapprof command	Statistical target	Class A		Class B		Class C	
		Instance count	Total size#	Instance count	Total size#	Instance count#	Total size#
-explicit	• Java heap • Explicit heap	1	122	2	22	2	2
-noexplicit	• Java heap	1	111	1	11	1	1

#
The total size indicates the total size of the instances. The unit is bytes.

The formula for calculating the total size of the instances of each class is as follows:

- When statistical target contains an Explicit heap
  - In class A: a1+b1+b2+c1+c2
  - In class B: b1+b2+c1+c2
  - In class C: c1+c2

- When statistical target does not contains an Explicit heap
  - In class A: a1+b2+c2
  - In class B: b2+c2
    (Though the instance b1 is not targeted, the instance b2 and c2 under b1 are targeted, and therefore, the size of those instances that exist in the reference relation is added to the class B)
  - In class C: c2

In the instance statistical functionality, investigate for the reference relation of the object referred in the following order from the object that is the base. The base objects are the objects those are not investigated in other reference relation. 1., 2., and 3. indicates the priority order for investigating reference relation.

1. Ascending order of the address within the Java heap
2. Ascending order of the address within an Explicit management heap

When the reference destination objects are investigated, return up to the branch point, and investigate for the reference relation.

Moreover, when the reference destination objects are the objects that is the base of other reference relation, handle it as a reference destination object. Investigate for the reference relation until all the objects that are the base are lost.

When using instance statistical functionality, instance count of each class is output for number of instances. The following contents are output for the total size of instances:

- The size of the object that is the base is added to the corresponding class. The size of the reference destination object is added to the corresponding class, and also added to the object that is the base and to their corresponding class of all objects that exist in the reference relation extended up to the corresponding class.

9.4.2 Class-wise statistical information output by the instance statistical functionality

This subsection describes about the output format, the output items, and the output examples of the class-wise statistical information output by the instance statistical functionality.

(1) Output format and output items

The output format of the class-wise statistical information output by the instance statistical functionality is as follows:

- **Output format**

```
Java Heap Profile
-----------------------
_________________Size__Instances__Class_________________
 <total_size> <Instance_count> <class_name>
 <total_size> <Instance_count> <class_name>
...
```

- **Output items**
  The items listed in the output format are described as follows.

Table 9–6: Output items (instance statistical functionality)

Output items	Meaning
<total_size>	The total size of the instance is output in byte unit.
<Instance_count>	The number of instance is output.
<class_name>	The class name is output.

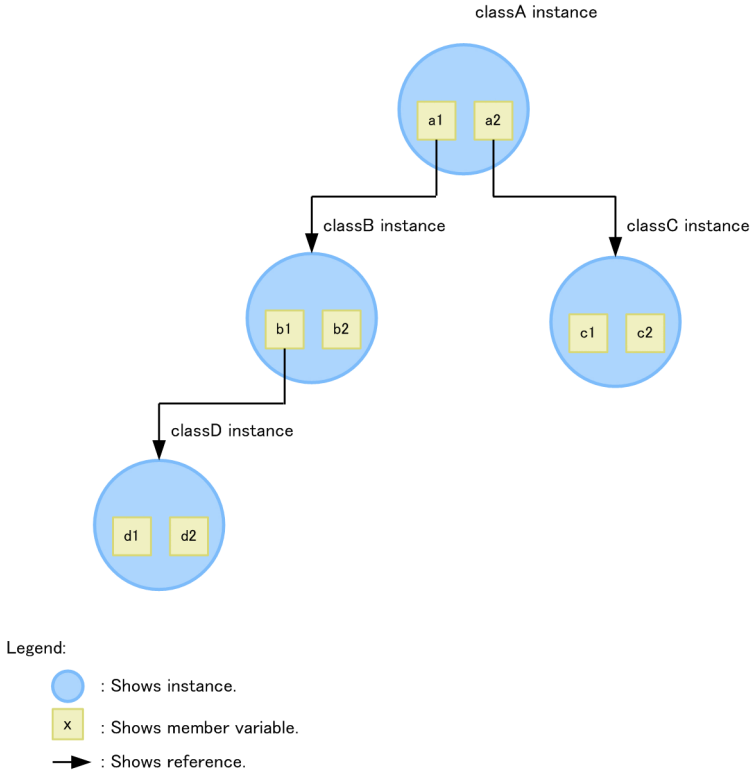
(2) Example of output

The output example of the class-wise statistical information output by the instance statistical functionality is described as the example of the following source.

```
public class instance {
 public static void main(String args[]) {
   classA cls_a = new classA();
   try {
     Thread.sleep(20000);
   } catch (Exception e) {}
 }
}
class classA {
 classB a1;
 classC a2;
 classA() {
   a1 = new classB();
   a2 = new classC();
 }
}
class classB {
 classD b1;
 String b2;
 classB() {
   b1 = new classD();
   b2 = null;
 }
}
class classC {
 String c1, c2;
 classC() {
   c1 = null;
   c2 = null;
 }
}
class classD {
 String d1, d2;
 classD() {
   d1 = null;
   d2 = null;
 }
}
```

The following figure shows the instance structure for above source.

Figure 9–2: Instance structure (instance statistical functionality)

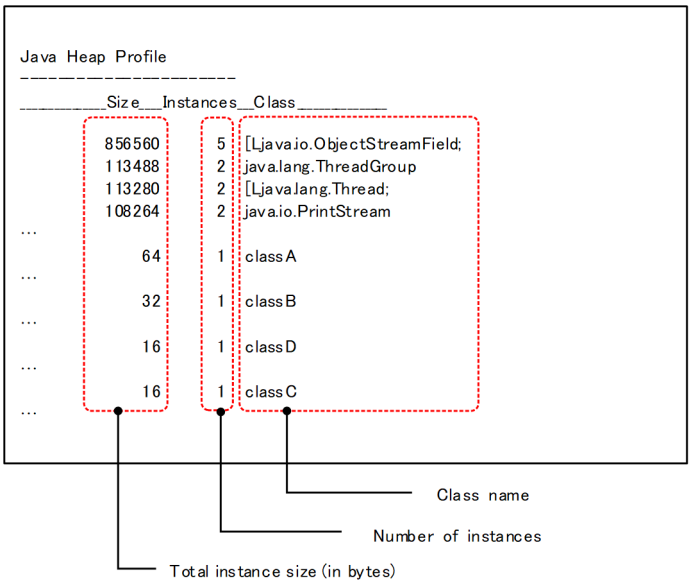


When using the above instance structure, add the following size for each class in the instance statistical functionality:

- Size of class A: a1+a2+ b1+b2+ c1+c2+ d1+d2
- Size of class B: b1+b2+ d1+d2
- Size of class C: c1+c2
- Size of class D: d1+d2

The following figure shows the output result of the instance statistical functionality.

Figure 9–3: Output result (instance statistical functionality)



9.5 STATIC member statistical functionality

This section describes the STATIC member statistical functionality.

The STATIC member statistical functionality is the functionality that outputs the class-wise statistical functionality. You can output the total size of instances of static members for each class.

The STATIC member statistical functionality uses the `jheapprof` command for output. For details about the execution format and the execution examples of the `jheapprof` commands, see *9.3.3 Outputting Statistic Information for Each Class*.

The following table describes the organization of this section.

Table 9–7: Organization of this section (STATIC member statistical functionality)

Category	Title	Reference
Description	Overview of the STATIC member statistical functionality	*9.5.1*
	Class-wise statistical information output by the STATIC member statistical functionality	*9.5.2*

Note:

The function-specific explanation is not available for "Implementation", "Settings", "Operations", and "Notes".

9.5.1 Overview of the STATIC member statistical functionality

Like the instance statistical functionality, the STATIC member statistical functionality is used to check the memory leak in applications.

The point of difference with the instance statistical functionality is that the instance statistical functionality recursively adds the size of the instances referenced from the non-static fields of the instance retrieved first, while the STATIC member statistical functionality recursively adds the size of the instances referenced from the static fields (static fields of the class) of the instance retrieved first. This enables you to acquire the total size of the instances of the static member for each class. However, apart from the instances retrieved first, both instance statistical functionality and STATIC member statistical functionality check the reference relationship based on the non-static members of the instances.

For details about the difference between the instance statistical functionality and the STATIC member statistical functionality, see *9.5.2(2) Output examples*.

To check the cause of a memory leak, execute the STATIC member statistical functionality as follows, before and after the application processing for which you want to check the memory leak, then take the difference between the number of instances and the total size of the instances in points 1 and 3, check the amount of increase in the numbers, and then identify the classes that are causing the memory leak.

1. Execute the STATIC member statistical functionality.
2. Execute the application processing for which you want to check the memory leak.
3. Execute the STATIC member statistical functionality.

Note that because the STATIC member statistical functionality recursively adds the size of the instances referenced by the static fields of each class, you cannot check the total size of only the instances of each class (the size excludes the size of the instances being referenced).

In the STATIC member statistical functionality, investigate for the reference relation from the object that is the base. The base objects are the object those are not investigated in other reference relation and refers the STATIC member of all classes of JavaVM.

When the reference destination objects are investigated, return up to the branch point, and investigate for the reference relation. Investigate for the reference relation until all the objects that are the base are lost.

When using the STATIC member statistical functionality, the following contents are output for instance count and for the total size of instances:

- Add the size of all objects that exist in the reference relation and number of objects in the object that is the base. Add this value in the class having STATIC member that is referred by the object that is the base, as a statistical value.

9.5.2 Class-wise statistical information output by the STATIC member statistical functionality

This subsection describes about the output format, the output items, and the output examples of the class-wise statistical information output by the STATIC member statistical functionality.

(1) Output format and output items

The output format of the class-wise statistical information output by the STATIC member statistical functionality is as follows:

- **Output format**

```
Java Heap Dump Static Profile
-----------------------
_________________Size__Instances__Class_________________
 <total_size> <Instance_count> <class_name>
 <total_size> <Instance_count> <class_name>
...
```

- **Output items**

  The items listed in the output format are described as follows.

  Table 9–8: Output items (STATIC member statistical functionality)

Output items	Meaning
<total_size>	The total size of the instance is output in byte unit.
<Instance_count>	The number of instance is output.
<class_name>	The class name is output.

(2) Output examples

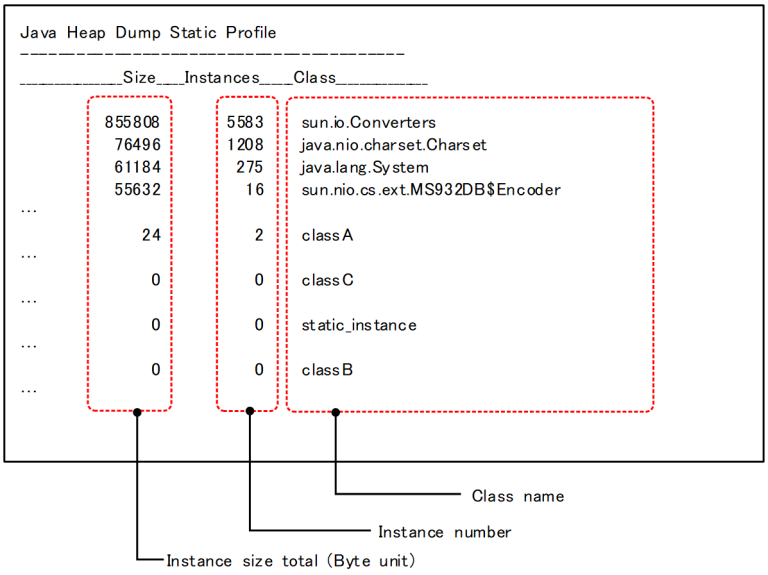
The output example of the class-wise statistical information output by the STATIC member statistical functionality, is described as an example of the following source:

```
public class static_instance {
  public static void main(String args[]) {
    classA cls_a;
    classB cls_b;
```

```
    classC cls_c;

    cls_a = new classA();
    cls_b = new classB();
    cls_c = new classC();
    cls_b.cls_c = cls_c;
    cls_a.cls_b = cls_b;

    try {
      Thread.sleep(20000);
    } catch (Exception e) {}
 }
}
class classA {
 static classB cls_b;
}
class classB {
 classC cls_c;
}
class classC {
}
```

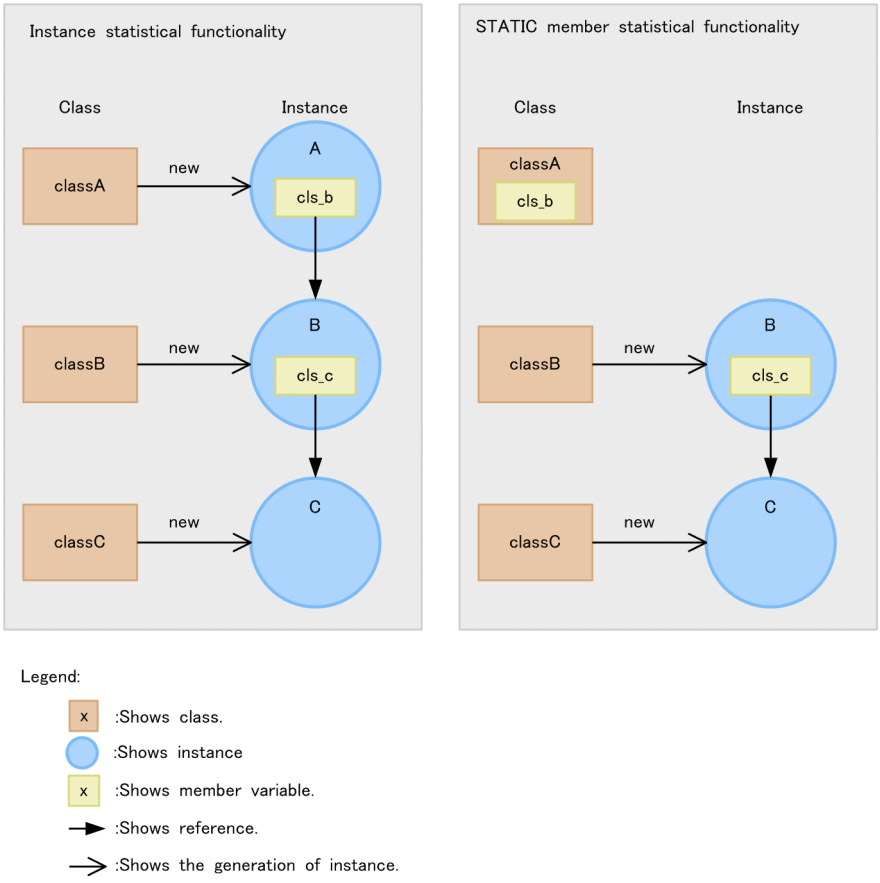
The following figure shows the output result of the STATIC member statistical functionality.

Figure 9–4: Output result (STATIC member statistical functionality)



Moreover, there is a difference in the reference relation for the above source in the instance statistical functionality and in the STATIC member statistical functionality and the following figure shows this difference.

Figure 9–5: Difference in reference relation in the instance statistical functionality and in the STATIC member statistical functionality



The reference relation of the respective functionality is as follows:

- Reference relation of the instance statistical functionality
  Instance variable `cls_b` of Instance A ->Instance variable `cls_c` of instance B -> Instance C
- Reference relation of the STATIC member statistical functionality
  Class variable `cls_b` of class A ->Instance variable `cls_c` of instance B -> Instance C

9.6 Reference-related information output functionality

This section describes about the reference-related information output functionality.

You can output the reference relation of the instances for the specified class in a sequence from the beginning, using the reference-related information output functionality.

The following table describes the organization of this section.

Table 9–9: Organization of this section (reference-related information output functionality)

Category	Title	Reference
Description	Overview of the reference-related information output functionality	*9.6.1*
	Class-wise statistical information output by the reference-related information output functionality	*9.6.2*
	Class-wise statistical information output by the static field-based reference relationship output functionality	*9.6.3*
Notes	Notes for the output of the static field-based reference relationships	*9.6.4*

Note:
The function-specific explanation is not available for "Implementation", "Settings", and "Operations".

9.6.1 Overview of the reference-related information output functionality

The class from which an instance of the class that is specified in the `-class` option of the `jheapprof` command is to be referenced, is output in the sequence beginning from the reference relation of the instance.

If there are many instances of the specified class, all the corresponding instances are output. Even if there are many instances having same name, you can identify the instances separately since the following information is output after the instance name:

- Address of the instance
- Area name to which instance belongs

In the reference-related information output functionality, investigate for the reference relation of object referred in the following order from the object that is the base. The base objects are the objects those are not investigated in other reference relation. 1., 2., and 3. indicates the priority order for investigating reference relation.

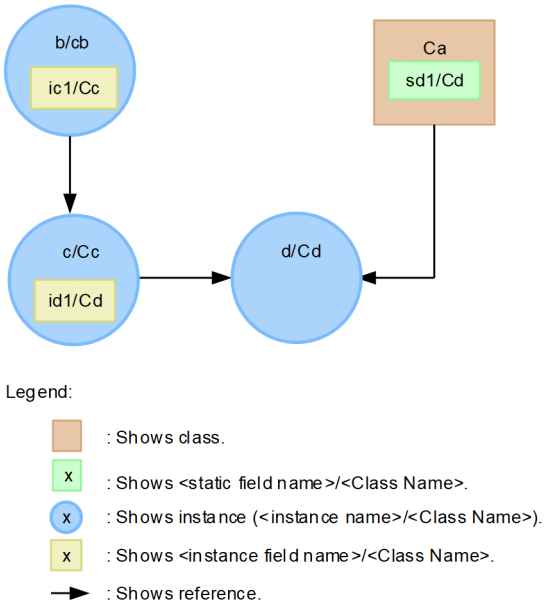
1. Ascending order of the address within the Java heap
2. Ascending order of the address within an Explicit management heap

When the reference destination object is the class specified in the `-class` option, the reference relation extended from the base object up to the object of the class specified in the `-class` option, is output in the reference-related information. Moreover, when the reference destination objects are investigated, return up to the branch point, and investigate for the reference relation. Moreover, when the reference destination objects are the objects that is the base of other reference relation, handle it as a reference destination object. Investigate for the reference relation until all the objects that are the base are lost.

Also, the static field-based reference relationship output functionality is enabled when you specify the `-staticroot` option with the `jheapprof` command. The reference relationship output functionality is a prerequisite for this functionality. With the static field-based reference relationship output functionality, the static field-based reference

relationship is additionally output to the reference relationship output by the reference relationship output functionality. This output information is used to ascertain the cause of the memory leak using the reference relationships based on static fields.

The following figure shows an example of a reference relationship with a memory leak:



If you execute the static field-based reference relationship output functionality for the reference relationship instance `d/Cd` in the figure, the following reference relationship is output:

```
Reference of class Cd
---------------------
Cb(0x088065f0)[Tenured]
  Cc(0x08806668)[Tenured]
    Cd(0x088066e0)[Tenured]
------

Reference of class Cd from static field
---------------------------------------
static field Ca.sd1
  Cd(0x088066e0)[Tenured]
------
```

Reference relation information output by the reference relation information output functionality

Reference relation information output by the reference relation information output functionality based on the static field

Based on this information, the references of the following fields will be cleared as a measure against the leak. Due to this, the instance `d/Cd` is collected by GC, so the memory leak can be resolved.

- Instance field `id1` of the instance `c/Cc`
- Static field `sd1` of the class `Ca`

For details on the reference relationships of the reference relationship output functionality, see *9.6.2 Class-wise statistical information output by the reference-related information output functionality*, and for the static field-based reference relationships, see *9.6.3 Class-wise statistical information output by the static field-based reference relationship output functionality*.

9.6.2 Class-wise statistical information output by the reference-related information output functionality

This subsection describes about the output format, the output items, and the output examples of the class-wise statistical information output by the reference-related information output functionality.

(1) Output format and output items

The output format of the class-wise statistical information output by the reference-related information output functionality is as follows:

- **Output format**

```
Reference of class option-specified-class-name
-------------------------------------------#
class-name(address)[area-name]
  class-name(address)[area-name]
    option-specified-class-name(address)[area-name]
------
class-name(address)[area-name]
  java.lang.ref.Finalizerrepetition-count times
    class-name(address)[area-name]
      class-name(address)[area-name]
        option-specified-class-name(address)[area-name]
------
...
```

#

'-(hyphen)' for the number in which 19 is added is output to the string length of the *option-specified-class-name*.

- **Output items**

The each items listed in the output format are described as follows:

Table 9–10: Output items (reference-related information output functionality)

Output items	Meaning
class-name	The class name to be referenced by an instance of the class that is specified in the `-class` option of the `jheapprof` command is output.
address	The address of the instance is output.
area-name	The area where instance belongs is output. • Eden: Indicates Eden area. • Survivor: Indicates Survivor area. • Tenured: Indicates Tenured area. • EM(eid=<id>): Indicates Explicit memory block.
option-specified-class-name	The class name specified in the `-class` option of the `jheapprof` command is output.
`java.lang.ref.Finalizer`	The object of `java.lang.ref.Finalizer` created in the class having the `finalize()` method is output all together.
repetition-count	The count where the reference of the Finalizer instance is continuous is output.

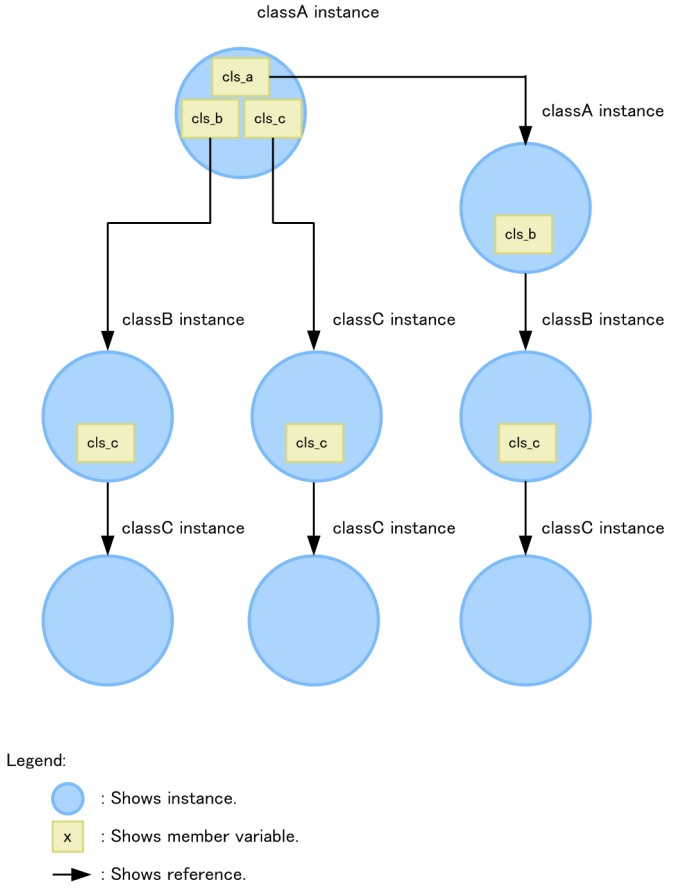
(2) Example of output

The output example of the class-wise statistical information output by the reference-related information output functionality, is described as the example of the following source:

```
public class instance2 {
  public static void main(String args[]) {
    classA cls_a1 = new classA(); classA cls_a2 = new classA();
    classB cls_b1 = new classB(); classB cls_b2 = new classB();
    classC cls_c1 = new classC(); classC cls_c2 = new classC();
    classC cls_c3 = new classC(); classC cls_c4 = new classC();
    cls_a1.cls_a = cls_a2; cls_a1.cls_b = cls_b1;
    cls_a1.cls_c = cls_c1; cls_a2.cls_b = cls_b2;
    cls_b1.cls_c = cls_c2; cls_b2.cls_c = cls_c3;
    cls_c1.cls_c = cls_c4;
    try {
      Thread.sleep(20000);
    } catch (Exception e) {}
  }
}
class classA {
  classA cls_a;
  classB cls_b;
  classC cls_c;

  classA() {
    classB cls_b;
  }
}
class classB {
  classC cls_c;
}
class classC {
  classC cls_c;
}
```

The following figure shows the instance structure.

Figure 9–6: Instance structure (reference-related information output functionality)



The following figure shows the output result of the reference-related information output functionality. In such case, specify an argument `-class` *class-name* in the `jheapprof` command, and then execute the command.

Figure 9–7: Output result (reference-related information output functionality)

```
Reference of class classC
----------
classA(0x10766840)[Eden]
 classB(0x10766998)[Eden]
  classC(0x10766a88)[Survivor]
------                                  (1)Displays reference of classA -->classB --> classC
classA(0x10766840)[Eden]
 classC(0x10766920)[Survivor]
------                                  (2)Displays first classA --> classC within reference
                                        relation of classA --> classC --> classC
classA(0x10766840)[Eden]
 classC(0x10766920)[Survivor]
 classC(0x10766ab8)[EM(eid=1)]
------                                  (3)Displays reference of classA --> classC --> classC
classA(0x10766840)[Eden]
 classA(0x10766858)[Tenured]
  classB(0x10766968)[Eden]
   classC(0x10766a28)[Survivor]
------                                  (4)Displays reference of classA --> classA --> classB -->
                                        classC
```

Note 1: (1) to (4) output order might change the processing order when searching the instance reference relationship.
Note 2: xxxxxxxxxx of (xxxxxxxxxx) shows the address on the instance memory.
Also, yy...yy of [yy...yy] shows the domain that the instance belongs to.

The address of all class A has same addresses (0x10766840). Therefore, it is understood that instances of all class A are same. On the other hand, the class B in (1) and (4) has different address and therefore, has different instances.

Note that the placement for the instances on the memory is changed due to the GC occurrence. Therefore, whenever the address and the area name is output, they might change every time.

9.6.3 Class-wise statistical information output by the static field-based reference relationship output functionality

This subsection describes the output format, output items, and examples of output of the class-wise statistical information output by the static field-based reference relationship output functionality.

(1) Output format and output items

- **Output format**

  The output format of the class-wise statistical information output by the static field-based reference relationship output functionality is as follows:

```
Reference of class option-specified-class-name from static field
-------------------------------------------------------------#1
static field static-field-declaring-class-name#2.static-field-name#2
class-name(address)[area-name]
  class-name(address)[area-name]
    option-specified-class-name(address)[area-name]
------
...
```

  #1:

  The number of "-" (hyphens) that are output is equal to 37 added to the number of characters in *option-specified-class-name*.

  #2:

  Indicates the base of the reference relationship.

- **Output items**

  The following table describes the items shown in the output format.

  Table 9–11: Output items (static field-based reference relationship output functionality)

Output items	Meaning
static-field-declaring-class-name	The class name declaring the static field that forms the base is output.
static-field-name	The name of the static field that forms the base is output.
class-name	The class name of the instance that references the instances of the classes specified in the `-class` option of the `jheapprof` command is output.
address	The address of the instance is output.
area-name	The area to which the instance belongs is output. • Eden: Indicates the Eden area. • Survivor: Indicates the Survivor area. • Tenured: Indicates the Tenured area. • EM(eid=<id>): Indicates the Explicit memory block.
option-specified-class-name	The class name specified in the `-class` option of the `jheapprof` command is output.

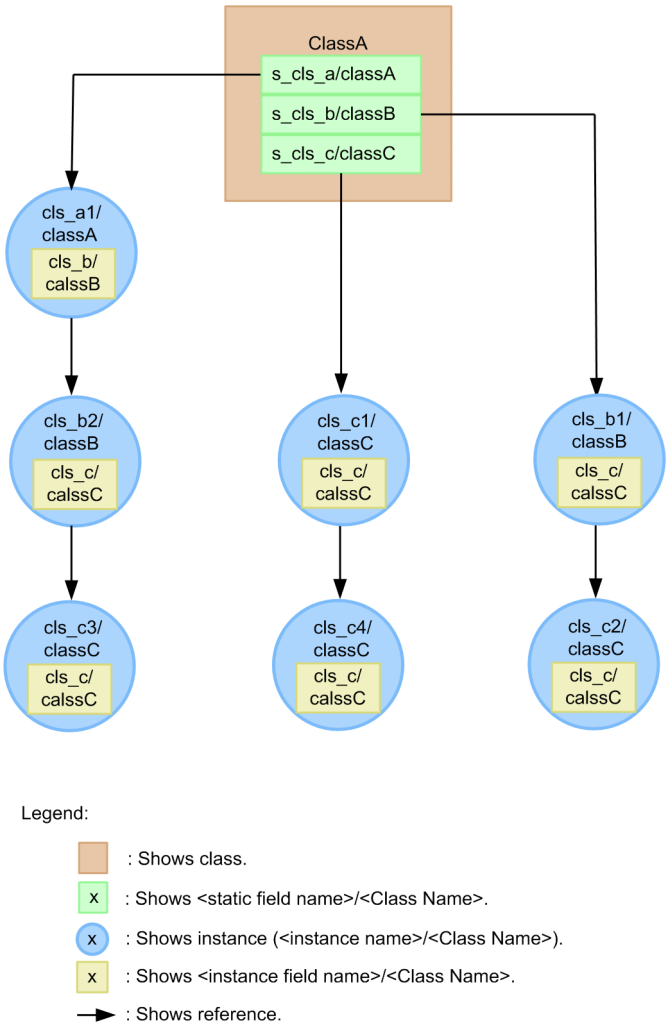
(2) Examples of output

This subsection gives an example output of the class-wise statistical information output by the static field-based reference relationship output functionality using the following source as an example:

```
import JP.co.Hitachi.soft.jvm.MemoryArea.*;
public class static_reference {
  public static void main(String args[]) {
    try {
      classA cls_a1 = new classA();
      classB cls_b1 = new classB();
      classB cls_b2 = new classB();
      classC cls_c1 = new classC();
      classC cls_c2 = new classC();
      classC cls_c3 = new classC();
      BasicExplicitMemory emem = new BasicExplicitMemory();
      classC cls_c4 = (classC)emem.newInstance(classC.class);
      cls_a1.s_cls_a = cls_a1;
      cls_a1.s_cls_b = cls_b1;
      cls_a1.s_cls_c = cls_c1;
      cls_a1.cls_b = cls_b2;
      cls_b1.cls_c = cls_c2;
      cls_b2.cls_c = cls_c3;
      cls_c1.cls_c = cls_c4;
      Thread.sleep(20000);
    } catch (Exception e) {e.printStackTrace();}
  }
}
class classA {
  static classA s_cls_a;
  static classB s_cls_b;
  static classC s_cls_c;
  classB cls_b;
}
class classB {
  classC cls_c;
}
class classC {
  classC cls_c;
  public classC(){
  }
}
```

The following figure shows the structure of the instances.

Figure 9–8: Structure of the instances (static field-based reference relationship output functionality)



The following figure shows the output result of the static field-based reference relationship output functionality. In this case, the `jheapprof` command is executed specifying the argument `-class` *class-name* `-staticroot`.

Figure 9–9: Output result (static field-based reference relationship output functionality)

```
Reference of class classC from static field
------------------------------------------
static field classA.s_cls_a
  classA(0x10511d08)[Tenured]
    classB(0x10511d78)[Eden]
      classC(0x10511df0)[Survivor]
------
static field classA.s_cls_b
  classB(0x10511d68)[Tenured]
    classC(0x10511de0)[Tenured]
------
static field classA.s_cls_c
  classC(0x10511dd0)[Tenured]
    classC(0x03bc0000)[EM(eid=1)]
------
static field classA.s_cls_c
  classC(0x10511dd0)[Tenured]
------
```

(1) Shows the reference of the static field s_cls_a →classA→classB→classC of classA

(2) Shows the reference of the static field s_cls_b →classB→classC of classA

(3) Shows the reference of the static field s_cls_c →classC→classC of classA

(4) Shows the portion of the static field s_cls_c→classC of the first classA in the reference relation of the static field s_cls_c →classC→classC of classA

Note 1: The processing order for the output sequence in (1)-(4) sometimes differs depending on the status when checking the reference relation of the instances.

Note 2: xxxxxxxxx of (xxxxxxxxx) shows the address in the instance memory. Also, yy...yy of [yy...yy] shows the area to which an instance belongs to.

9.6.4 Notes for the output of the static field-based reference relationships

If the static field-based reference relationship output functionality is enabled, the execution time of the `jheapprof` command is longer by the time taken for executing the reference relationship output functionality, as compared to when the static field-based reference relationship output functionality is disabled.

9.7 Pre-statistical GC selection functionality

This section describes about the pre-statistical GC selection functionality.

You can use the pre-statistical GC selection functionality for selecting the processing that is to be executed before the class-wise statistical information is output.

The following table describes the organization of this section.

Table 9–12: Organization of this section (Pre-statistical GC selection functionality)

Category	Title	Reference
Description	Overview of the pre-statistical GC selection functionality	*9.7.1*
	Guidelines for selecting the GC	*9.7.2*

Note:
The function-specific explanation is not available for "Implementation", "Settings", "Operations", and "Notes".

9.7.1 Overview of the pre-statistical GC selection functionality

You can execute the class-wise statistical functionality to output the class-wise statistical information to the extended thread dump. In the pre-statistical GC selection functionality, you can select the processing that is to be executed before the class-wise statistical information is output. By selecting this processing that is to be executed, you can acquire the various changes in the appearance of the Java object in the class-wise statistical information, depending on the purpose of investigation.

When using the pre-statistical GC selection functionality, specify the processing to be executed by an argument of the `jheapprof` command. The following table lists the process and the arguments of the `jheapprof` commands that can be implemented before executing the class-wise statistical functionality.

Table 9–13: Process and the arguments of the jheapprof commands that can be implemented before executing the class-wise statistical functionality

Type of process	Process contents	Argument of the jheapprof command
Execution of a Full GC	Collects the used object for an entire JavaVM specific area even including Tenured area.	-fullgc
Execution of a copy GC	Collects the used object only for the Eden area and for the Survivor area	-copygc
No execution of any GC	Does not collect the used object though available.	-nogc

Note that if you execute the class-wise statistical functionality when specifying `-XX:+HitachiVerboseGC` and `-XX:+HitachiVerboseGCPrintCause` in the JavaVM start option, the following information is output to extended `verbosegc` information:

- Type of GC
- Cause for GC occurrence in the extended thread dump

The above information differs with the information output by an argument specified in the `jheapprof` command. The following table lists the relation between the arguments of the `jheapprof` command and the output information.

Table 9–14: Relation between the arguments of the jheapprof command and the output information

Argument of the jheapprof command	Type of GC	Cause for GC occurrence
-fullgc	Full GC	JHeapProf Command
-copygc	GC	JHeapProf Command

9.7.2 Guidelines for selecting the GC

The processing that is to be specified using the pre-statistical GC selection functionality differs depending on the purpose of investigating the class-wise statistical information to be output.

The guidelines for selecting the processing depending on the purpose of investigation is described here.

Specify the processing that can be selected using the pre-statistical GC selection functionality by the argument of the `jheapprof` command. Depending on the object that is to be investigated or depending on the class-wise statistical information used in the type of investigation, you can select the processing.

The following table lists the guidelines for selecting the processing.

Table 9–15: Guidelines for selecting the processing

Processing (argument of the jheapprof command)	Investigation target	Example of how to investigate the class-wise statistical information
Execution of a Full GC (-fullgc)	Changing the object by Full GC	You can identify the object that causes the memory leak of the Java heap, since the used object is collected by Full GC.
Execution of a copy GC (-copygc)	Changing the object by copy GC	With the copy GC, you can identify the object with a long life that migrates to the Tenured area. From this information, you can identify the object that causes increase in the occurrence count of the Full GC.
No execution of any GC (-nogc)#	Object with short life is being collected by executing GC	Information on all objects including the used object is output. You can identify the object with a high memory sharing that becomes the cause (super-large object) when the GC collection occurs frequently.

#

When `-nogc` is specified in the argument of the `jheapprof` command, the information on all short-lived objects is output, and therefore, the log output volume increases.

9.8 Unused objects statistical functionality in the Tenured area

This section describes the unused objects statistical functionality in the Tenured area.

The unused objects statistical functionality in the Tenured area is the functionality used to output the class-wise statistical information. You can use this functionality in the Tenured area to identify the unused objects of the Tenured area.

To use the statistical functionality for the unused objects of the Tenured area, specify `-garbage` in the arguments of the `jheapprof` command. For details on the `jheapprof` command, see *jheapprof(Output of extended thread dump containing Hitachi class-wise statistical information)* in the *uCosminexus Application Server Command Reference Guide*.

The following table describes the organization of this section.

Table 9–16: Organization of this section (Unused objects statistical functionality in the Tenured area)

Category	Title	Reference
Description	Overview of the unused objects statistical functionality in the Tenured area	*9.8.1*
	Class-wise statistical information output by the unused objects statistical functionality in the Tenured area	*9.8.2*
Notes	Notes for executing the unused objects statistical functionality in the Tenured area	*9.8.3*

Note:
The function-specific explanation is not available for "Implementation", "Settings", and "Operations".

9.8.1 Overview of the unused objects statistical functionality in the Tenured area

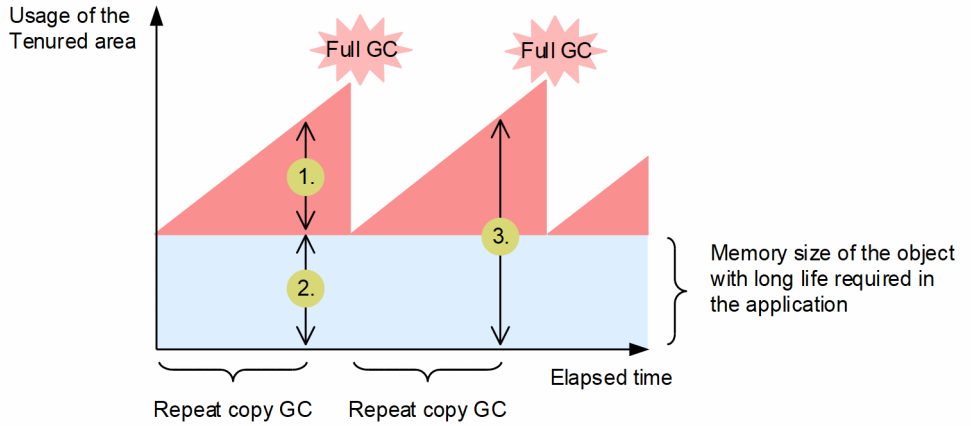
You can use the unused objects statistical functionality in the Tenured area to identify only the unused objects that are accumulated in the Tenured area, and output them to the thread dump file. This subsection describes the mechanism of the unused objects statistical functionality in the Tenured area.

(1) Output of the size of unused objects

Objects with a long life accumulate in the Tenured area based on iteration of the copy GC. Among the accumulated objects with a long life, the objects that lose their usage as time lapses, remain in the Tenured area as unused objects. After this, a Full GC occurs when the memory becomes full. You can check the usage of the Tenured area from the time of occurrence of a copy GC until the time of occurrence of a Full GC by using the unused objects statistical functionality in the Tenured area and the instance statistical functionality.

The following figure shows the contents that you can identify using the unused objects statistical functionality in the Tenured area and the instance statistical functionality.

Figure 9–10: Contents that can be identified using the unused objects statistical functionality in the Tenured area and the instance statistical functionality



When the instance statistical functionality is executed without executing the pre-statistical GC, the size of the unused object shown in step 3 in Figure 9-10 is output. This size is same as the memory usage status in the Tenured area that includes the size of unused objects of the Tenured area corresponding to step 1 of Figure 9-10 and the objects in use in the Tenured area corresponding to step 2 of Figure 9-10.

On the other hand, when you execute the unused objects statistical functionality in the Tenured area, you can output the memory usage status (corresponding to step 1 of Figure 9-10) in the Tenured area that excludes the objects in use shown in step 2 of Figure 9-10. You can use the unused objects statistical functionality in the Tenured area to identify the unused objects that act as the augmentation factors for the Tenured area, which enables you to inhibit the Full GC.

(2) Checking the reference relationship of unused objects

In the unused objects statistical functionality of the Tenured area, search the base objects in the ascending order of their address in the Tenured area. Even among the searched objects, the objects that are not investigated in another reference relationship become the base objects.

When you have already investigated the referenced objects, return to the branching point and investigate the reference relationship. Also, if a referenced object is the base object of another reference relationship, then handle as a referenced object. Investigate a reference relationship until all base objects are eliminated.

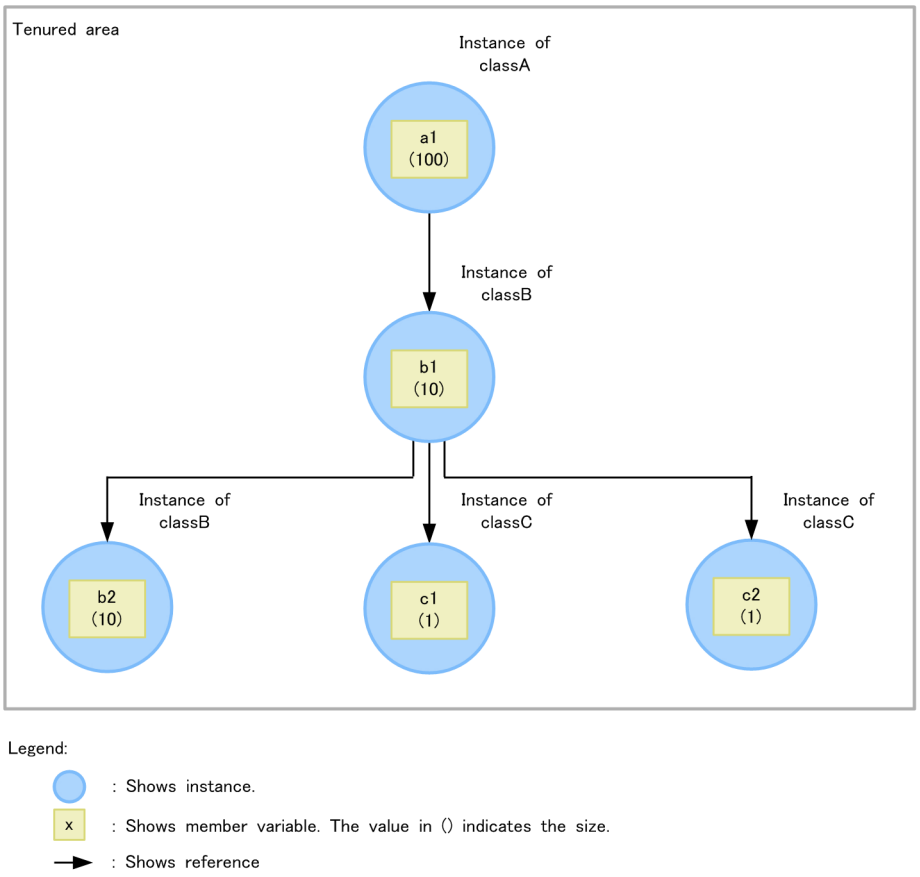
For the unused objects statistical functionality in the Tenured area, the total of the number of instances and instance size is output. Add the corresponding classes for number of instances. The following contents are output to the total size of instances:

- The size of the base objects is added to the corresponding class. The size of the referenced objects is added to the corresponding class, and then added to the corresponding classes of the base objects and all objects that exist in the reference relationship up to the corresponding class.

Note that when you execute the unused objects statistical functionality in the Tenured area, the instance statistical functionality, STATIC member statistical functionality, and pre-statistical GC selection functionality become disabled.

The following figure shows an example of the reference relationship based on the unused objects in the Tenured area.

Figure 9–11: Example of the reference relationship based on the unused objects in the Tenured area



The reference relationship shown in Figure 9-11 is described as follows:

Number of instances

- classA: One, because of the existence of a1
- classB: Two, because of the existence of b1 and b2
- classC: Two, because of the existence of c1 and c2

Total instance size

- classA: 122, when the size of the instance of classA (a1) and the size of the referenced instances (b1, b2, c1, c2) are added
- classB: 22, when the size of the instances of classB (b1, b2) and the size of the referenced instances (c1, c2) are added
- classC: 2, when the instances of classC (c1, c2) are added

The following is an output example, when the reference-related information shown in Figure 9-11 is output by executing the unused objects statistical functionality in the Tenured area:

```
Garbage Profile
---------------
__________________Size__Instances__Class_________________
                   122           1 A
                    22           2 B
                     2           2 C
```

9.8.2 Class-wise statistical information output by the unused objects statistical functionality in the Tenured area

This subsection describes the output format, output items, and output example of the class-wise statistical information output by the unused objects statistical functionality in the Tenured area.

(1) Output format and output items

The output format of the class-wise statistical information output by the unused objects statistical functionality in the Tenured area is as follows:

- **Output format**

```
Garbage Profile
-----------------------
__________________Size__Instances__Class_________________
                   size      number             class-name
                   size      number             class-name...
```

- **Output items**

  Each item listed in the output format is described as follows.

  Table 9–17: Output items (Unused objects statistical functionality in the Tenured area)

Output item	Meaning
Size	The total size of instances is output in bytes.
Number	The number of instances is output.
Class-name	The class name is output.

(2) Output example

The following is an output example of the class-wise statistical information output by the unused objects statistical functionality in the Tenured area:

```
Garbage Profile
---------------
_________________Size__Instances__Class_________________
               35234568       10648 java.util.HashMap
                5678900       10668 [Ljava.util.HashMap$Entry;
                4456788        7436 java.util.HashMap$Entry
                4321000         200 java.util.WeakHashMap
                1234568         190 [Ljava.util.WeakHashMap$Entry
                 454400           4 java.util.WeakHashMap$Entry
                      0           0 java.lang.Class
...
```

9.8.3 Notes for executing the unused objects statistical functionality in the Tenured area

This subsection describes the notes for executing the unused objects statistical functionality in the Tenured area.

(1) Notes when the unused objects statistical functionality in the Tenured area is executed immediately after a Full GC

If you execute the unused objects statistical functionality in the Tenured area immediately after a Full GC, the statistical processing is executed when the unused objects in the Tenured area for which the statistics are to be collected have been collected. Therefore, the total instance size within the class-wise statistical information and the number of instances becomes less, and the unused objects cannot be identified effectively. To effectively identify the unused objects, execute the unused objects statistical functionality in the Tenured area. The following describes the execution of the unused objects statistical functionality in the Tenured area separately for the case where the timing of occurrence of a Full GC is known and for the case where it is not known.

(a) When the timing of occurrence of a Full GC is known

If you execute the unused objects statistical functionality in the Tenured area immediately before a Full GC, the statistical processing is executed when the number of unused objects in the Tenured area for which the statistics are to be collected is large. Therefore, the total instance size within the class-wise statistical information and the number of instances becomes large, and the unused objects can be identified effectively.

(b) When the timing of occurrence of a Full GC is not known

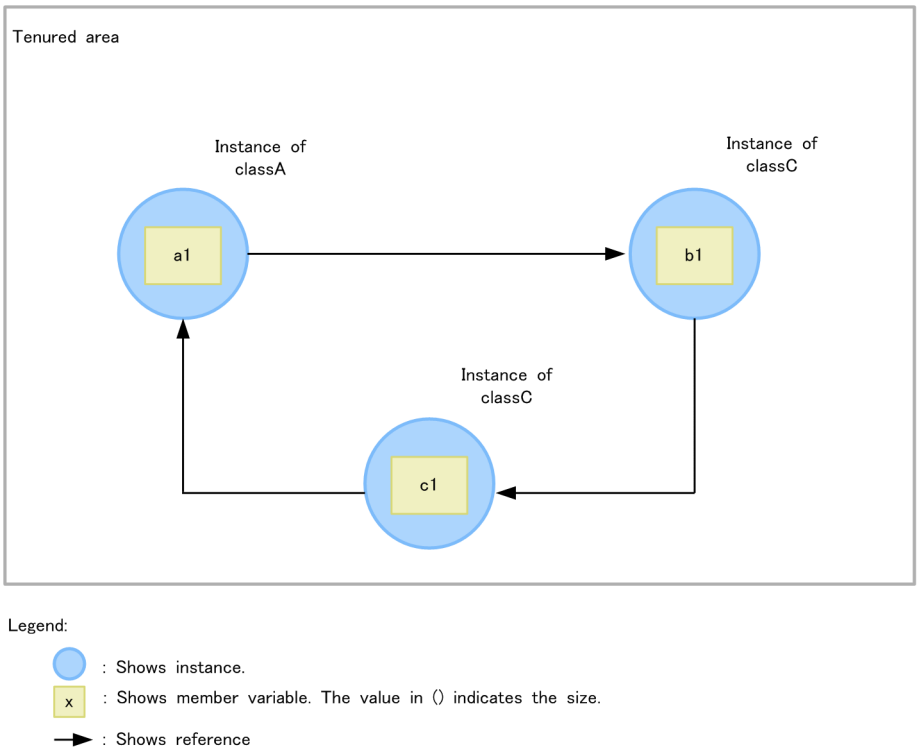
To increase the total instance size within the class-wise statistical information, the number of instances, and effectively identify the unused objects:

1. To know the timing of occurrence of Full GC, set the `-XX:+HitachiVerboseGC` option to output the extended `verbosegc` information at JavaVM startup. By specifying the option, you can acquire the GC information.
2. Execute the unused objects statistical functionality in the Tenured area at a fixed interval in JavaVM. As a result, you can acquire the GC information and multiple class-wise statistical information.
3. You can acquire the date and time of a Full GC from the extended `verbosegc` information, so select the class-wise statistical information that is close to the Full GC. The class-wise statistical information close to the Full GC is information about the statistical processing executed when the number of unused objects in the Tenured area for which statistics are to be collected is large.

(2) Notes on statistical results

The following is an example of reference relationship showing the notes on statistical results.

Figure 9–12: Example of reference relationship (Notes on statistical results)

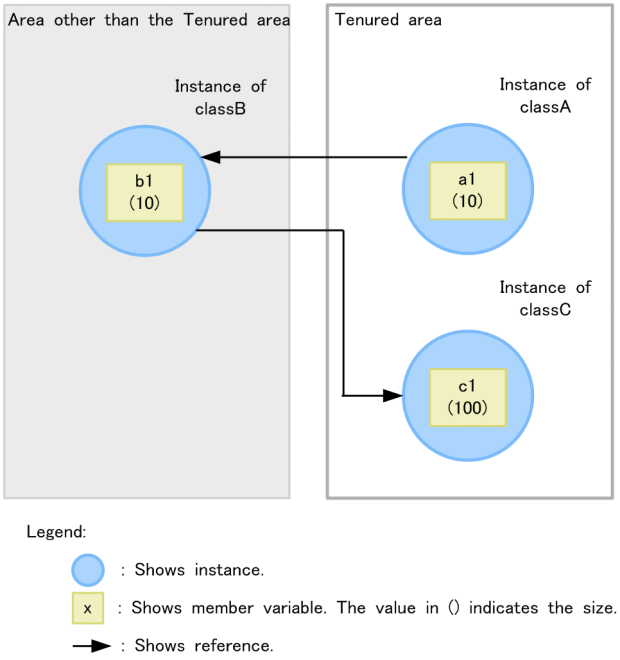


In this figure, when you assume `a1` as the lowest address, the statistical processing is executed with the reference relationship `a1`→ `b1`→ `c1` in which `a1` is the base object. At this point, if you assume `b1` or `c1` as the base object, expected results will not be output to the statistical results of the unused objects statistical functionality in the Tenured area and the base object list output functionality for Tenured augmentation factors.

(3) Notes on objects for which the statistics are not to be collected

The following figure shows the objects for which the statistics are not to be collected for a case in which the objects (`a1` and `c1`) of the Tenured area have a reference relationship with the object (`b1`) of other than the Tenured area.

Figure 9–13: Example of reference relationship (Objects for which the statistics are not to be collected)

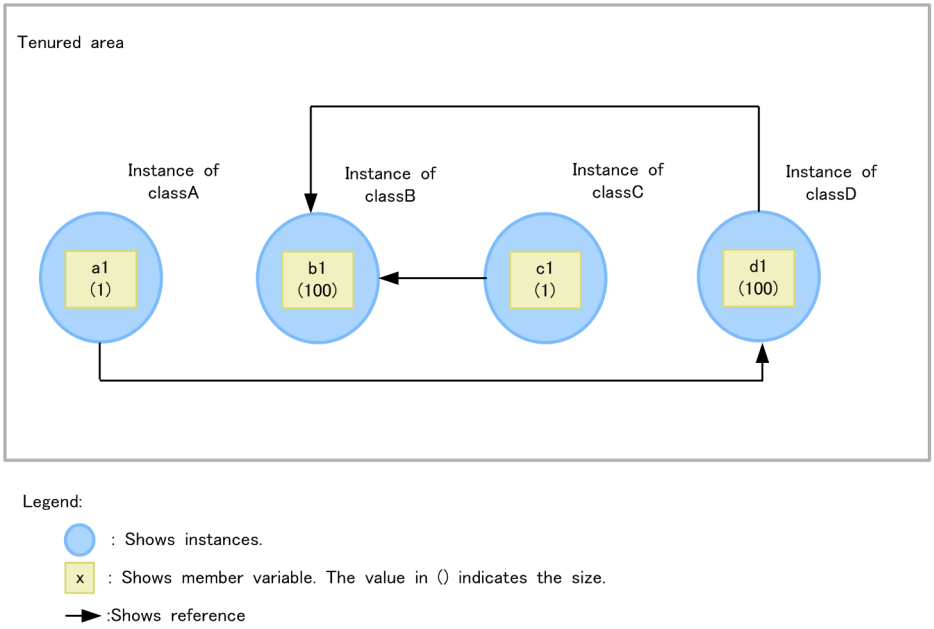


In this figure, the statistics are not collected for the object (`b1`) that belongs to other than the Tenured area. However, the size of the referenced object (`c1`) of the Tenured area is added to the total instance size of the class B.

(4) Notes for reference relationship from multiple objects

The following figure shows a reference relationship from multiple objects for a case having a reference relationship in which one object (`b1`) is referenced from multiple reference sources (`c1` and `d1`).

Figure 9–14: Example of reference relationship (For a reference relationship from multiple objects)



In this figure, if you execute the unused objects statistical functionality in the Tenured area, the statistical processing is executed from the base object having the lowest address among the base objects (`c1` and `a1`) of the reference relationship to which the reference source belongs. Therefore, if `a1` is the object with the lowest address and `c1` is the base object, the

expected results will not be output in the statistical results of the unused objects statistical functionality in the Tenured area and the base object list output functionality for Tenured augmentation factors.

(5) Notes for increasing the statistical value

If executing the unused objects statistical functionality in the Tenured area and the base object list output functionality for Tenured augmentation factors in a Java process in which the `-XX:+HitachiAutoExplicitMemory` option is specified, the following phenomena occurs:

- In the class-wise statistical information output by the unused objects statistical functionality in the Tenured area, the total instance size of the `float` type array type (information in which `[F` is output in the class name) and the statistical value of the number of instances becomes larger than the original statistical value. An output example of the class-wise statistical information is as follows:

```
Garbage Profile
---------------
__________________Size__Instances__Class__________________
             43861400     473859 [F
                    0          0 java.util.Collections$EmptyMap
                    0          0 sun.security.util.Debug
                    0          0 java.nio.ByteOrder
```

- If you output a `float` type array type (information in which `[F` is output in the class name) in the base object list that is output by the base object list output functionality for Tenured augmentation factors, the total statistical value of the instance size becomes larger than the original statistical value. An output example of the base object list for Tenured augmentation factors is as follows:

```
Garbage Profile Root Object Information
---------------------------------------
*, [F # 43861400
```

If you are using versions earlier than 08-70, specify the `-XX:-HitachiExplicitMemoryPartialTenuredAreaCollection` option to prevent occurrence of this phenomenon.

9.9 Base object list output functionality for Tenured augmentation factors

This section describes the base object list output functionality for Tenured augmentation factors.

The base object list output functionality for Tenured augmentation factors is used to output the class-wise statistical information. You can use the base object list output functionality for Tenured augmentation factors to output the base object information of the unused objects that are identified using the unused objects statistical functionality in the Tenured area.

To use the base object list output functionality for Tenured augmentation factors, specify `-rootobjectinfo` in the arguments of the `jheapprof` command. For details on the `jheapprof` command, see *jheapprof(Output of extended thread dump containing Hitachi class-wise statistical information)* in the *uCosminexus Application Server Command Reference Guide*.

The following table describes the organization of this section.

Table 9–18: Organization of this section (Base object list output functionality for Tenured augmentation factors)

Category	Title	Reference
Description	An overview of the base object list output functionality for Tenured augmentation factors	*9.9.1*
	Class-wise statistical information output by the base object list output functionality for Tenured augmentation factors	*9.9.2*

Note:

The function-specific explanation is not available for "Implementation", "Settings", "Operations", and "Notes".

9.9.1 Overview of the base object list output functionality for Tenured augmentation factors

With the base object list output functionality for Tenured augmentation factors, you can use the unused objects statistical functionality in the Tenured area to list the base object information of unused objects and output to the thread dump file.

The unused objects statistical functionality in the Tenured area is a prerequisite for the base object list output functionality for Tenured augmentation factors.

Tip

You can specify the information acquired using the base object list output functionality for Tenured augmentation factors in the automatic allocation setup file that is specified when you use the Explicit Memory Management functionality. For details about using the Explicit management heap with the automatic allocation configuration file, see *7.13.2 Using the Explicit Memory Management functionality by using the automatic placement configuration file* in the *uCosminexus Application Server Expansion Guide*.

This subsection describes the mechanism of the base object list output functionality for Tenured augmentation factors.

(1) Output of the base object list

The base object list output functionality for Tenured augmentation factors is executed in continuation after the processing of the unused objects statistical functionality in the Tenured area. For details about the processing executed in the unused objects statistical functionality in the Tenured area, see the subsection *9.8.1 Overview of the unused objects statistical functionality in the Tenured area*.

In the base object list output functionality for Tenured augmentation factors, search the objects in the ascending order of their address within the Tenured area. Among the searched objects, acquire the class information of objects that are already investigated in a reference relationship, and also the information of unused objects and save in an output options list.

The information saved in the output options list is sorted according to the total instance size, and only the class information in which the total size is more than the value specified in the `-rootobjectinfost` option of the `jheapprof` command are output.

(2) Checking the reference relationship of unused objects

The following is an example of the reference relationship based on the unused objects within the Tenured area, when you execute the base object list output functionality for Tenured augmentation factors to acquire the base objects.

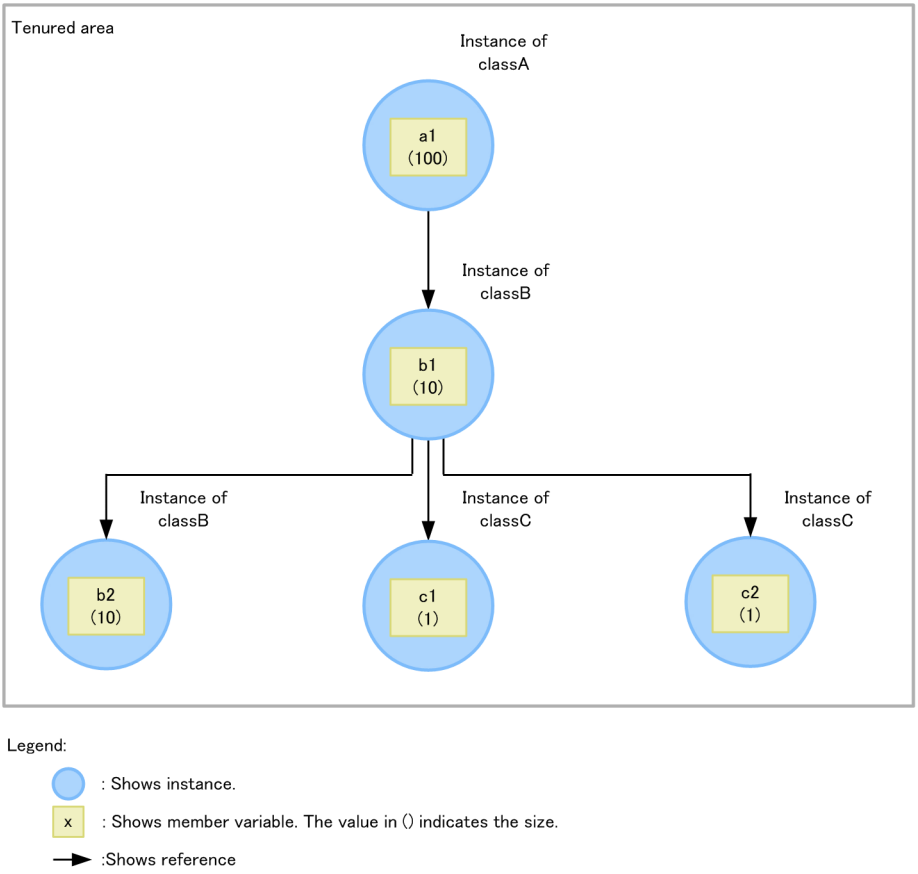
Number of instances

- classA: One, because of the existence of `a1`
- classB: Two, because of the existence of `b1` and `b2`
- classC: Two, because of the existence of `c1` and `c2`

Total instance size

- classA: 122, when the size of the instance of classA (`a1`) and the size of the referenced instances (`b1`, `b2`, `c1`, `c2`) is totaled
- classB: 22, when the size of the instances of classB (`b1`, `b2`) and the size of the referenced instances (`c1`, `c2`) is totaled
- classC: 2, when the instances of classC (`c1`, `c2`) are totaled

Figure 9–15: Example of the reference relationship based on the unused objects within the Tenured area



When you execute the unused objects statistical functionality in the Tenured area to acquire information about a reference relationship as is shown in Figure 9-15, the information acquired by the base object list output functionality for Tenured augmentation factors will be as follows:

- Base point object: `a1`
- Class of the base point object: `A`
- Total instance size of class A of the base point object: `122`

The following is an output example for the case when you execute the base object list output functionality for Tenured augmentation factors to output the information about the reference relationship as shown in Figure 9-15:

```
Garbage Profile Root Object Information
---------------------------------------
*, A # 122
```

9.9.2 Class-wise statistical information output by the base object list output functionality for Tenured augmentation factors

This subsection describes the output format and output items of the class-wise statistical information output using the base object list output functionality for Tenured augmentation factors.

- **Output format**

```
Garbage Profile Root Object Information
---------------------------------------
*, class-name # size
...
```

- **Output items**

  The following table lists each item in the output format.

  Table 9–19: Output items (Base object list output functionality for Tenured augmentation factors)

Output item	Meaning
Class-name	The class name of base objects acting as the Tenured augmentation factors is output.
Size	The total instance size acquired from the class-wise statistical information of the unused objects statistical functionality in the Tenured area is output in bytes.

9.10 Class-wise statistical information analysis functionality

This section describes the class-wise statistical information analysis functionality.

If using the class-wise statistical information analysis functionality, you can output the information acquired as class-wise statistical information in the CSV format.

The following table describes the organization of this section.

Table 9–20: Organization of this section (Class-wise statistical information analysis functionality)

Category	Title	Reference
Description	Overview of the class-wise statistical information analysis functionality	*9.10.1*
	Output example of the class-wise statistical information analysis functionality	*9.10.2*
Notes	Notes for the class-wise statistical information analysis functionality	*9.10.3*

Note:

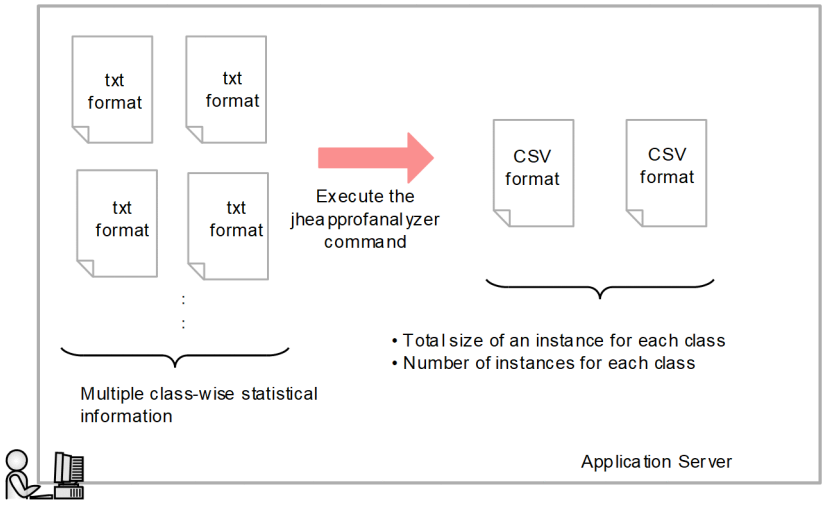
The function-specific explanation is not available for "Implementation", "Settings", and "Operations".

9.10.1 Overview of the class-wise statistical information analysis functionality

Execute the `jheapprofanalyzer` command (class-wise statistical information analysis functionality) to output the total instance size of each class and the number of instances of each class in the time series, assuming multiple extended thread dump files with class-wise statistical information as input files. The files are output in the CSV format.

The following figure shows the flow execute the class-wise statistical information analysis functionality for the CSV output of the information that is acquired as the class-wise statistical information.

Figure 9–16: Flow of CSV output by executing the class-wise statistical information analysis functionality



Using the class-wise statistical information analysis functionality, you can output the information about the instances with large total size and can check the memory usage of only such instances. To output only the instances with large total size, specify the threshold value in `-DJP.co.Hitachi.soft.jvm.tools.jheapprofanalyzer.threshold`, and then specify in the

`jheapprofanalyzer` command and execute the command. For details about the `jheapprofanalyzer` command, see *jheapprofanalyzer (CSV output of Hitachi class-wise statistical information analysis file)* in the *uCosminexus Application Server Command Reference Guide*.

9.10.2 Output example of the class-wise statistical information analysis functionality

This subsection describes the input files, output files, and output format of the class-wise statistical information analysis functionality.

(1) Input files

The extended thread dump files in which the class-wise statistical information is output are used as the input files in the class-wise statistical information analysis functionality.

(2) Output files

The files output in the class-wise statistical information analysis functionality include two types of files, such as files that output the total instance size of each class and files that output the number of instances of each class. The output files are created in the current directory with the following names.

Table 9–21: Names of output files

Type of output file	Example of output file name
Instance total size file	`JheapprofAnalyzer_size_`*nnn*`.csv`
Instance count file	`JheapprofAnalyzer_num_`*nnn*`.csv`

Legend:

nnn: The file segmentation number is output. The segmentation number is in the range of 001 to 999.

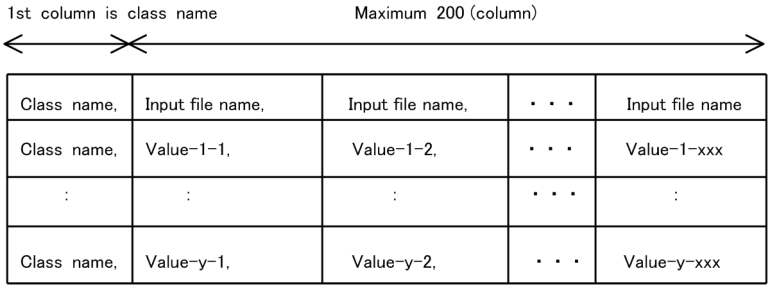
The output file is segmented when the number of columns exceeds 201. When the number of files exceeds 999, the count returns to 001 and files are re-written.

When the number of columns at which segmentation is performed exceeds 201 (1 column for class name + 200 columns for value), the output format is the same for the segmented file as well.

(3) Output format

The following figure shows the output format of the files output in the class-wise statistical information analysis file. Note that the output format of the CSV file in which the total instance size and number of instances are output is also the same.

Figure 9–17: Output format of files output in the class-wise statistical information analysis functionality



Class name,	Input file name,	Input file name,	・・・	Input file name
Class name,	Value-1-1,	Value-1-2,	・・・	Value-1-xxx
:	:	:	・・・	:
Class name,	Value-y-1,	Value-y-2,	・・・	Value-y-xxx

Legend:
Input file name: Class-wise statistical information specified for each processing
Class name: A class name output to input file
Value: Total size of instance or number of instances

Demarcate a class name and value, and a value from another value with a comma. End a line with a value (including a blank).

Class names are output in a random order. Based on the date displayed as the value in the first row of an input file, the input files are arranged side by side starting from the file with the oldest date. If input files with the same date exist, they are connected randomly and arranged side by side.

Reference note

If you execute the class-wise statistical information analysis functionality more than once, classes might be removed or added during processing. Also, 0 will be output as the value when the corresponding class does not exist. The following figure shows the class information.

Figure 9–18: Example of class information

First class-wise statistical information (A.txt)	Second class-wise statistical information (B.txt)	Third class-wise statistical information (C.txt)
ClassA 100 ClassB 100	ClassA 100 ClassB 30 ClassC 50 ClassB 0	ClassA 100 ClassC 50

In case with the above class information, the output results when 0 is set as the threshold value for `-DJP.co.Hitachi.soft.jvm.tools.jheapprofanalyzer.threshold` are as shown in Figure 9-19:

Figure 9–19: Output example of the class-wise statistical information analysis functionality

```
class name,A.txt,B.txt,C.txt
ClassA,100,100,100
ClassB,100,30,0
ClassC,0,50,50
ClassD,0,0,0
```

The maximum value of total instance size is 0 to 2^{63}-1, and the maximum value of the number of instances is 0 to 2^{31}-1. If the same class name exists in an input file, the total instance size is added. The number of instances is also added. If the respective maximum values are exceeded due to adding up, the specified maximum value

is output. Note that if the corresponding class information does not exist in all input files of a class or if the threshold value is not reached, the information of that class is not output.

9.10.3 Notes for the class-wise statistical information analysis functionality

When using the class-wise statistical information analysis functionality, do not update or delete an input file during the execution of the `jheapprofanalyzer` command. The class-wise statistical information analysis functionality opens the file twice: once when acquiring the date and once when reading the data. Therefore, the results will not be guaranteed, when you update or delete an input file during the execution of the command.

9.11 Tenuring distribution information output functionality of the Survivor area

This section describes about the tenuring distribution information output functionality of the Survivor area.

You can use the tenuring distribution information output functionality of the Survivor area for investigating the Use Status of the Survivor area when executing the copy GC. You can use this information for tuning the memory size.

The following table describes the organization of this section.

Table 9–22: Organization of this section (tenuring distribution information output functionality of the Survivor area)

Category	Title	Reference
Description	Overview of the tenuring distribution information output functionality of the Survivor area	*9.11.1*
	Output format and output example of the tenuring distribution information of the Survivor area	*9.11.2*
Settings	Settings for execution environment	*9.11.3*
Notes	Precautions when using tenuring distribution information output functionality of the Survivor area	*9.11.4*

Note:
The function-specific explanation is not available for "Implementation" and "Operations".

9.11.1 Overview of the tenuring distribution information output functionality of the Survivor area

In product JavaVM, the output contents of the log are expanded more than that of a standard JavaVM, so that you can acquire more troubleshooting information. The product JavaVM log is output to the Hitachi JavaVM log file. The tenuring distribution information output functionality of the Survivor area outputs the tenuring distribution information of the Java object of the Survivor area to the Hitachi JavaVM log file when executing the copy GC. You can use this information for investigating the Use Status of the object of the Survivor area and for tuning the memory size of the Survivor area. For details about the tuning the memory size of the Survivor area, see *7.6.1 Estimating the memory size of the Survivor area in Java heap* in the *uCosminexus Application Server System Design Guide*.

You can use the tenuring distribution information output functionality of the Survivor area to output the date and time in addition to the tenuring distribution information of the Survivor area. Moreover, since the output destination is the Hitachi JavaVM log file, you can obtain synchronization with other logs.

For details about the Hitachi-specific Java VM log file, see *4.10 JavaVM log (JavaVM log file)*. Moreover, for details about the tuning of Java VM, see *7. JavaVM Memory Tuning* in the *uCosminexus Application Server System Design Guide*.

9.11.2 Output format and output example of the tenuring distribution information of the Survivor area

The tenuring distribution information of the Survivor area is output following to the log of the copy GC when the copy GC occurs. The output format and the output example of the tenuring distribution information of the Survivor area are as follows:

Output format

```
[PTD]<date>[Desired survivor:size bytes][New threshold:value][MaxTenuringT
hreshold: max_value][age1:total_age1][age2:total_age2]...[agen:total_agen]
```

Description

- PTD: An identifier that indicates the tenuring information of the Survivor area
- Date: Date and time when GC occurred (outputs only when `-XX:+HitachiVerboseGCPrintDate` (extended `verbosegc` information date output option) is specified)#1
- size: Size of the Survivor area (unit: bytes)
- value: Tenuring threshold value of the object that migrates to the Tenured area when the GC will occur next time
- max_value: Specified value of `-XX:MaxTenuringThreshold`#2
- total_age1: Total of memory size used by the one-year-old object (unit: bytes)
- total_age2: Total of memory size used by the one-to-two-year-old object (unit: bytes)
- total_agen: Total of memory size used by the object from 1-year-old to n-year-old (unit: bytes)#3

Note:

When -XX:+HitachiCommaVerboseGC is specified, the tenuring distribution information is output in the following format:

```
PTD,date,size,value,max_value,total_age1,total_age2,...,total_agen
```

#1

The time similar to the log of the corresponding copy GC is displayed.

#2

Set the threshold value of the frequency in which the Java object is replaced in the `From` space and in the `To` space when executing the copy GC in the specified value of the `-XX:MaxTenuringThreshold`.

#3

The existing object is displayed in the order from minimum years to the maximum years. If the maximum years of the displayed object are nearer to the value of `max_value`, then the object with long life exists is indicated.

Example of output

```
[VGC]<Wed May 28 11:45:23 2008>[GC 648K->136K(1984K), 0.0013020 secs][Def
New::Eden: 512K->0K(512K)][DefNew::Survivor: 0K->0K(64K)][Tenured: 136K->1
36K(1408K)][Metaspace: 3634K(4492K, 4492K)->3634K(4492K, 4492K)][class spa
ce: 356K(388K, 388K)->356K(388K, 388K)][cause:ObjAllocFail][User: 0.000000
0 secs][Sys: 0.0000000 secs]
[PTD]<Wed May 28 11:45:23 2008>[Desired survivor:5467547 bytes][New thresh
old:30][MaxTenuringThreshold:31][age1:1357527][age2:1539661]
```

You can check the following contents in the above output example:

- The output trigger is a copy GC that occurred at 11:45:23 on Wednesday, May 28, 2008.

- The memory size of the Survivor area is 5,467,547 bytes. The object of the Survivor area is of up to two years old. The memory size used by the one-year-old object is 1,357,527 bytes, and the memory size used by the object from one-year-old to two-year-old, is 1,539,661 bytes.

9.11.3 Settings for execution environment

You must perform the following settings when using the tenuring distribution information output functionality of the Survivor area:

- J2EE server
- Batch server
- Java application

(1) Setting the J2EE server

Implement the settings of the J2EE server in the Easy Setup definition file. Specify the definition of the tenuring distribution information output functionality of the Survivor area in the JavaVM start parameter (`add.jvm.arg`) in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file. The parameter value to be specified is as follows:

- `-XX:+HitachiVerboseGCPrintTenuringDistribution`

  Output the tenuring distribution information of the Survivor area to the Hitachi JavaVM log file. The default value is `-XX:-HitachiVerboseGCPrintTenuringDistribution`.

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*.

(2) Setting the batch server

Implement the settings of the batch server in the Easy Setup definition file. Specify the definition of the tenuring distribution information output functionality of the Survivor area in the JavaVM start parameter (`add.jvm.arg`) in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file. The parameter value to be specified is as follows:

- `-XX:+HitachiVerboseGCPrintTenuringDistribution`

  Output the tenuring distribution information of the Survivor area to the Hitachi JavaVM log file. The default value is `-XX:-HitachiVerboseGCPrintTenuringDistribution`.

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*.

(3) Setting the Java application

Implement the settings of the Java application in the `usrconf.cfg` (option definition file for Java application). Specify the definition of the tenuring distribution information output functionality of the Survivor area in the `add.jvm.arg` of the `usrconf.cfg` key. The parameter value to be specified is as follows:

- `-XX:+HitachiVerboseGCPrintTenuringDistribution`

  Output the tenuring distribution information of the Survivor area to the Hitachi JavaVM log file. The default value is `-XX:-HitachiVerboseGCPrintTenuringDistribution`.

For details on `usrconf.cfg` (option definition file for Java applications), see *12.2.1 usrconf.cfg (Option definition file for Java applications)* in the *uCosminexus Application Server Definition Reference Guide*.

9.11.4 Precautions when using tenuring distribution information output functionality of the Survivor area

When using the tenuring distribution information output functionality of the Survivor area, the log when executing the copy GC increases as compared to the case when not used. We recommend that this functionality is to be used only for the tuning of the Survivor area.

9.12 hndlwrap functionality

This section describes the `hndlwrap` functionality.

The `hndlwrap` functionality can suppress the occurrence of a logoff event of JavaVM during a logoff. Note that you can use this functionality only in Windows.

The following table describes the configuration of this section.

Table 9–23: Configuration of this section (hndlwrap functionality)

Category	Title	Reference
Description	Overview of the hndlwrap functionality	*9.12.1*
Notes	Notes for using the hndlwrap functionality	*9.12.2*

Note:
The function-specific explanation is not available for "Implementation", "Settings", and "Operations".

9.12.1 Overview of the hndlwrap functionality

The `hndlwrap` functionality prevents the occurrence of a logoff event and closure of the window during a logoff.

This functionality uses Java Virtual Machine Tool Interface (JVMTI). If `hndlwrap2.dll` is loaded in the JVMTI interface, an event handler that detects the class preparation event and ignores the logoff and window closure events is installed. As a result, when you execute the `hndlwrap` functionality by specifying the `-agentlib:hndlwrap2` option, you can operate the command even after logging off. For details on the options to be specified, see *14.5 Java HotSpot VM options that can be specified in Cosminexus* in the *uCosminexus Application Server Definition Reference Guide*.

9.12.2 Notes for using the hndlwrap functionality

The notes for using the `hndlwrap` functionality are as follows:

- When you specify the `-XX:+EagerXrunInit` option, the `-Xrunhndlwrap` option becomes disabled.
- When you execute a logoff while a Java application using the `hndlwrap` functionality on a command prompt is being executed, a pop-up message indicating an error occurs and you cannot perform the logoff.
- You cannot execute the `-agentlib:hndlwrap2` option simultaneously with another JVMTI program. If you execute this option, the operation will not be guaranteed.
- You cannot specify the `-Xrunhndlwrap` option and `-agentlib:hndlwrap2` option simultaneously.

9.13 Functionality to set the upper limit of allocation size of C heap during JIT compilation

Use C heap when compiling with Just In Time method (**JIT compiler**) supported by JavaVM. If you execute JIT compiler for methods with large numbers of processing or for large methods, the C heap size allocated for compilation process becomes larger and might result into insufficient C heap. In such cases, problems like forced termination of JavaVM or abnormal end of J2EE server might occur, which might result into complete suspension of the system.

To prevent the occurrence of such problems, you can set an upper limit to the C heap size used in JIT compilation. When the upper limit is exceeded, JIT compiling will be canceled and the subsequent compilation will be executed by interpreter method. Thus, you can prevent the forced termination of JavaVM and stopping of the system.

Specify the upper limit of allocation size of C heap during JIT compilation
by using the `-XX:HitachiJITCompileMaxMemorySize` option. For details on the `-XX:HitachiJITCompileMaxMemorySize` option, see *-XX:HitachiJITCompileMaxMemorySize (Option for specifying the maximum memory allocated for JIT compilation)* in the *uCosminexus Application Server Definition Reference Guide*.

9.14 Functionality to set the upper limit of the number of threads

The C heap memory usage increases with the increase in the number of threads used in an application. If an increase in the memory usage results in insufficient C heap, problems such as forced termination of JavaVM or abnormal end of J2EE server might occur, which might result into complete suspension of the system.

To prevent the occurrence of such problems, you can set an upper limit to the available number of threads. You can prevent C heap insufficiency by figuring out in advance the upper limit of the number of threads, and based on this you can determine the memory size allocated to C heap. Note that an exception is thrown, if the number of threads generated exceeds the set upper value. You can prevent stopping of the system by catching this exception in the application and taking appropriate measures.

Specify the upper limit of the number of threads by using the `-XX:HitachiThreadLimit` option. For details on the `-XX:HitachiThreadLimit` option, see *-XX:HitachiThreadLimit (Option for specifying the maximum number of threads)* in the *uCosminexus Application Server Definition Reference Guide*.

9.15 Notes on using the product JavaVM functionality (in UNIX)

This section describes the notes for using the product JavaVM functionality.

9.15.1 Common in UNIX

This subsection describes the common notes in UNIX.

- Operations when the SIGXFSZ signal is received

  When JavaVM receives the SIGXFSZ signal, the following message is output to the standard output and the processing continues:

  ```
  Java HotSpot(TM) 64-Bit Server VM warning: File size limit exceeded.
  ```

  However, if the upper limit of the file size is exceeded, the data is not written to the file.

- Operations when the SIGXCPU signal is received

  When JavaVM receives the SIGXCPU signal, the following message is output to the standard output and the processing continues:

  ```
  Java HotSpot(TM) 64-Bit Server VM warning: CPU time limit exceeded.
  ```

- AWT (in AIX)

  In AIX, AWT of Application Server uses XToolkit (sun.awt.X11.XToolkit). Motif-based MToolkit is not supported.

9.15.2 In AIX

This subsection describes the notes in AIX.

- Estimating the paging space

  When a J2EE server and a Web container server are started, to prevent a program from stopping (`SIGKILL`) due to insufficient paging while the program is running, set up the environment variable `PSALLOC=early` that specifies the early paging space allocation for AIX in the start shell. However, if you specify `PSALLOC=early` in the `/etc/environment` file, the early paging space allocation is specified for all the processes leading to insufficient paging space. Therefore, specify the environment variable only for the shell that starts the relevant server. Also, specify the environment variable `NODISCLAIM=true` at the same time. For the points to be considered when you estimate the paging space for the early paging space allocation, see the *AIX documentation*.

- Native library search path

  When the `java` command is used

  When you use the `java` command included in Developer's Kit for Java to execute a Java program, you can specify the directory search path of the system library loaded with `System.loadLibrary()` in the environment variable `LIBPATH`, and environment variable `LD_LIBRARY_PATH`. The search order prioritizes `LD_LIBRARY_PATH` and then `LIBPATH`.

  When the `java` command is not used

  When you use the JNI to start JavaVM and execute a Java program, the directory search path of the system library loaded with `System.loadLibrary()` is the path specified only in the environment variable `LIBPATH`.

- Using the JNI to start JavaVM

Note the following points when you use the JNI to start JavaVM:

- Make sure the following AIX-specific environment variables are set up:

  For csh (C shell)

  ```
  setenv AIXTHREAD_MUTEX_DEBUG OFF
  setenv AIXTHREAD_RWLOCK_DEBUG OFF
  setenv AIXTHREAD_COND_DEBUG OFF
  ```

  For sh (standard shell) and ksh

  ```
  export AIXTHREAD_MUTEX_DEBUG=OFF
  export AIXTHREAD_RWLOCK_DEBUG=OFF
  export AIXTHREAD_COND_DEBUG=OFF
  ```

  For these environment variables, see the relevant page (`https://www.ibm.com/support/knowledgecenter/ja/ssw_aix_72/performance/thread_env_vars.html`).

- Add the following paths to the environment variable `LIBPATH` and then execute the start program:

  ```
  /opt/Cosminexus/jdk/lib
  /opt/Cosminexus/jdk/lib/server
  ```

  Note that the environment variable `LIBPATH` need not be set up when these paths are specified as the linkage options for generating the start program.

- Font path

  The fonts displayed with the GUI are searched from the system font path. If the program for changing the system font path is executed before Developer's Kit for Java starts, the GUIs might be displayed incorrectly. Furthermore, the system font path is also changed by Developer's Kit for Java. Therefore, the GUIs might be incorrectly displayed with the programs such as the ones that search the system font path after Developer's Kit for Java starts.

  Specify the following settings before and after starting Developer's Kit for Java, to return the system font path to the default status:

  ```
  % xset fp default
  ```

- Program starting time

  The server VM (Java HotSpot Server VM) implements the Java virtual machine for Developer's Kit for Java. Compared to the earlier client VM (Java HotSpot Client VM) version, the program execution speed is generally high for the server VM, but the starting of the J2EE server programs might become slow. To start the programs quickly, specify the following option. However, when you specify this option, the execution speed of the program might become slower than normal.

  ```
  -XX:CompileThreshold=3000
  ```

- Notes on migrating from the 32-bit JavaVM (Application Server Version 8) to the 64-bit JavaVM (Application Server Version 9)

  As compared to the 32-bit JavaVM, the availability of a large address space is an advantage with the 64-bit JavaVM. Also, the pointer value of objects is 32 bits in the 32-bit JavaVM and 64 bits in the 64-bit JavaVM. Therefore, when you migrate from the 32-bit JavaVM to the 64-bit JavaVM, you must review the applications, memory tuning, and the environment. Note the following and then execute the migration process:

  - Increased object access cost

    The pointer value of the object changes from 32 bit to 64 bit, so the access to the object doubles at the maximum. The execution performance of applications that frequently access the objects might be affected with the 64-bit JavaVM.

  - Increased Java heap memory size

The pointer value of the object changes from 32 bit to 64 bit, so the size per object doubles at the maximum. As a result, the memory size of the Java heap used by the application also increases.

- Increase in the memory size used by a process

  With 64-bit JavaVM, the memory size used by the processes also increases apart from the Java heap memory size. The method of estimating the memory size used by the processes is the same as the method of estimating the core file size. For details on the estimation method, see *3.3.16(1) Setting the maximum size of core files*.

- JNI compatibility

  A native program created as a 32-bit binary and invoked with the JNI cannot be operated on the 64-bit JavaVM. Also, the 32-bit and 64-bit native programs cannot be operated concurrently in one process. Therefore, when using a native program invoked with the JNI, you must re-compile that program for the 64-bit JavaVM.

9.15.3 In Linux

This subsection describes the notes for Linux.

- Font-related messages

  When you start a GUI application on a non-console remote terminal such as the X server, the following messages might be output:

```
Warning: Cannot convert string
"-monotype-arial-regular-r-normal--*-140-*-*-p-*-iso8859-1" to type FontSt
ruct
```

  To avoid this, use the `xfs` command to start the font server, use the `xset` command on the X server, and then specify the font server in the font path. At this time, make sure that the missing font path is specified in the font server configuration file.

9.16 Finalize-retention resolution function

This section describes the finalize-retention resolution function. By using the finalize-retention resolution function, you can resolve the retention of finalization processing and suppress the occurrence of delays in the release of OS resources.

9.16.1 Overview

In an application that defines OS resource release processing in the `finalize()` method, the finalization processing might be retained and delays in the release of OS resources might occur. If you use the finalize-retention resolution function, whether finalization processing is retained in JavaVM is detected and the retained finalization processing is resolved.

The finalize-retention resolution function creates a finalization processing monitoring thread that monitors FinalizerThread when Java starts in order to detect the retention of finalization processing. FinalizerThread is a thread that is always present when Java is executed and processes the `finalize()` methods of objects one by one.

The finalization processing monitoring thread periodically monitors the objects being processed by FinalizerThread. If the following conditions are met, the finalization processing is considered to be retained and a new finalization thread is created.

- The finalization processing of the objects is not progressing in FinalizerThread.
- There is an object in the finalization queue.

Both FinalizerThread and the finalization thread execute the finalization processing to facilitate the resolution of the retained finalization processing. Note that the finalization thread ends when the finalization queue becomes empty.

9.16.2 Output information

The finalize-retention resolution function outputs a message to the standard output at the following timings.

- When the retention of finalization processing is detected and a new finalization thread is created
- When the created finalization thread ends

The following shows the format and an output example of the message that is output at each timing.

- When the retention of finalization processing is detected and a new finalization thread is created

  The following shows the format of the message that is output when retained finalization processing is detected and a new finalization thread is created.

  ```
  # FinalizerWatcherThread: Create: create secondary finalizer thread. [queue length = queue] <date>
  ```

  Description

  queue: Number of elements in the finalization queue

  date: Date and time when the finalization thread was newly created

The following is an output example:

```
# FinalizerWatcherThread: Create: create secondary finalizer thread. [queue
length = 128] <Mon May 26 18:00:36 JST 2008>
```

- When the created finalization thread ends

  The following shows the format of the message that is output when the generated finalization thread ends.

  ```
  # FinalizerWatcherThread: Finish: secondary finalizer thread is finished.
  <date>
  ```

  Description

  date: Date and time when the created finalization thread ended

The following is an output example:

```
# FinalizerWatcherThread: Finish: secondary finalizer thread is finished.  <
Mon May 26 20:12:26 JST 2008>
```

9.16.3 Settings for execution environment

To use the finalize-retention resolution function, you must specify settings for the following:

- J2EE server
- Batch server
- Java application

(1) Specifying settings for the J2EE server

Specify the J2EE server settings in the Easy Setup definition file. Specify the definition of the finalize-retention resolution function in the JavaVM startup parameter (`add.jvm.arg`) in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file. The parameter value to be specified is as follows:

- `-DJP.co.Hitachi.soft.jvm.autofinalizer=true`

  This enables the finalize-retention resolution function. If this function is enabled, a monitoring thread is created when Java starts. The default value is `true`, so specify `false` to disable the function.

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*.

(2) Specifying settings for the batch server

Specify the batch server settings in the Easy Setup definition file. Specify the definition of the finalize-retention resolution function in the JavaVM startup parameter (`add.jvm.arg`) in the `<configuration>` tag of the logical J2EE server (j2ee-server) in the Easy Setup definition file. The parameter value to be specified is as follows:

- `-DJP.co.Hitachi.soft.jvm.autofinalizer=true`

  This enables the finalize-retention resolution function. If this function is enabled, a monitoring thread is created when Java starts. The default value is `true`, so specify `false` to disable the function.

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*.

(3) Specifying settings for the Java application

Specify the Java application settings in the `usrconf.cfg` (option definition file for Java applications). Specify the definition of the finalize-retention resolution function in the `add.jvm.arg` key in `usrconf.cfg`. The parameter value to be specified is as follows:

- `-DJP.co.Hitachi.soft.jvm.autofinalizer=true`

  This enables the finalize-retention resolution function. If this function is enabled, a monitoring thread is created when Java starts. The default value is `true`, so specify `false` to disable the function.

For details on `usrconf.cfg` (option definition file for Java applications), see *12.2.1 usrconf.cfg (Option definition file for Java applications)* in the *uCosminexus Application Server Definition Reference Guide*.

9.16.4 Notes

The following are notes on using the finalize-retention resolution function.

- The value of the JavaVM startup parameter is referenced when Java starts. Therefore, even if you change the value of the JavaVM startup parameter by using the `java.lang.System.setProperty()` method or any other method while JavaVM is running, the finalize-retention resolution function will not be enabled.
- If multiple JavaVM startup parameters are specified, the value of the last specified parameter takes effect.
- The finalize-retention resolution function creates one resident thread to monitor the retention of finalization processing. The function also creates one thread to resolve the retention when it detects the retention of finalization processing. If the retention of finalization processing does not affect the system, you can suppress the creation of a thread to monitor the retention of finalization processing and a thread to resolve the retention by specifying `-DJP.co.Hitachi.soft.jvm.autofinalizer=false` for the `add.jvm.arg` parameter or the `add.jvm.arg` key to disable the finalize-retention resolution function.

9.17 Asynchronous log file output function

This section describes the asynchronous log file output function.

9.17.1 Overview

This function changes the output of log data to files so that it can be done asynchronously. Here, *asynchronous* means that, by preparing a thread dedicated to output processing of log data to files, log data can be output to files without synchronizing the timing of processing with the threads that execute processing other than log output to files. As an example, the following describes the log file output when a GC occurs.

During a GC, program processing stops. In the Hitachi JavaVM log file function, the thread executing the GC also outputs the GC information to the log file, so it is not possible to proceed to the next processing until the processing to output the GC information to the log file is finished.

By enabling this function, the processing to output the GC information to the log file is executed by a dedicated thread. This reduces the processing delays caused by threads that execute other processing waiting for the processing to output the GC information to the log file to finish.

Note that using this function might result in some missing log data. This function temporarily stores the log data in a buffer and then outputs it to a log file. If the amount of log data output is very large, or if the file I/O processing is extremely slow, some log data might not be output. Therefore, we recommend that you use this function if you have tuned the system to achieve 100 msec or less of program processing stop time due to a GC. If you have not performed such tuning, we do not recommend that you use this function.

9.17.2 Target log files

This function targets the following two files:

- Hitachi JavaVM log file
- Event log of the Explicit Memory Management functionality

9.17.3 Error cases

If the speed of writing to the buffer is faster than the speed of I/O processing to the file, the buffer might be rounded and the log might not be output.

Output format when the buffer is rounded and no log is output

```
[ASY]<skip_count> log is skipped.

skip_count: Number of skipped lines when the buffer is rounded and the log c
annot be output
```

If the creation of a dedicated thread for output to the log used by this function fails, the following messages might be output to the standard output and JavaVM does not start.

When the creation of a thread for the Hitachi JavaVM log file fails

```
Error occurred during initialization of VM
Could not create thread for VM: Asynchronous javalog thread creation failed
. thread_count threads exist.

thread_count: Number of threads that exist
```

When the creation of a thread for the Explicit management heap log file fails

```
Error occurred during initialization of VM
Could not create thread for VM: Asynchronous ehjavalog thread creation faile
d. thread_count threads exist.

thread_count: Number of threads that exist
```

9.17.4 Notes

When using the asynchronous log output function, if a value greater than 4096 is specified for the `-XX:HitachiOutOfMemoryStackTraceLineSize` or `-XX:HitachiJavaClassLibTraceLineSize` option, the function operates as if 4096 was specified as the option value. For details on the operation in this case, see *14.2 Details of JavaVM extension options* in the *uCosminexus Application Server Definition Reference Guide*.

9.17.5 Memory requirements

This function creates a new buffer and thread. If the function is enabled, this buffer and thread remain alive until JavaVM ends. Therefore, using this function consumes memory for the buffer and thread. The following shows the memory consumption.

First, the following shows the memory consumption of the buffer.

```
(Size of one log line = 4096 bytes) × (Number of lines stored in the buffer
= 1024 lines) = 4 Mbytes
```

If the Explicit Memory Management functionality is used, the memory consumption is doubled to 8 Mbytes because a buffer for the event log of the Explicit Memory Management functionality is also created.

Next, the memory consumption of the thread is the value specified in the `-Xss` option. Similar to the memory consumption of the buffer, if the Explicit Memory Management functionality is used, the memory consumption doubles because a thread for the event log of the Explicit Memory Management functionality is also created.

Table 9–24: Default values for -Xss for each platform

Platform	Default value for -Xss
Linux(EM64T)	1Mbyte
Windows	1Mbyte
AIX	1Mbyte

9.18 Object-pointer compression function

This section describes the object-pointer compression function.

9.18.1 Overview

The object-pointer compression function improves the memory usage efficiency of the Java application by compressing the size of Java objects created by a Java application. By enabling this function, you can reduce the usage of the Java heap area and explicit heap area# during execution of the Java application running on the Application Server.

You can enable the object-pointer compression function by specifying `-XX:+UseCompressedOops`. For details about the `UseCompressedOops` option, see *14.2 Details of JavaVM extension options* in the *uCosminexus Application Server Definition Reference Guide*.

If you enable this function, the memory usage during execution and the execution performance of JavaVM might change. In addition, the execution performance of JavaVM when this function is enabled varies depending on the difference in the OS environment when JavaVM is executed.

#: The explicit heap area applies only when the Explicit Memory Management functionality is enabled.

9.18.2 Prerequisites

This function is available if the total set size of each heap area (Java heap area and explicit heap area#) is less than 32 GB. If the total set size is 32 GB or more, this function is disabled even if you specify `-XX:+UseCompressedOops`.

```
Value specified for the heap area size, which enables the object-pointer com
pression function

(1) When the Explicit Memory Management functionality is used
  (value-specified-for--Xmx-option) + (value-specified-for--XX:HitachiExplic
itHeapMaxSize-option) < 32 GB

(2) When the Explicit Memory Management functionality is not used
  (value-specified-for--Xmx option) < 32 GB
```

#

The explicit heap area applies only when the Explicit Memory Management functionality is enabled.

9.18.3 Notes

1. If you enable this function, the usage of the Java heap area and explicit heap area when the Java application is executed decreases, and the amount of change depends on the Java application. If you consider the change in the memory usage due to the enabling of this function to be a problem, disable this function.
2. Note that enabling this function might affect execution performance such as throughput when the Java application is executed. This function affects the execution performance in the following two points:

   (a) The size of data (Java object) handled by JavaVM is reduced, which enables more efficient memory access (performance improvement factor).

(b) JavaVM requires calculations to compress the size of Java objects (performance degradation factor).

Due to the two factors (a) and (b), when this function is enabled, the performance when JavaVM is executed changes. Regarding (b), the calculation method also changes depending on the difference in the OS environment when JavaVM is started, which affects the execution performance. (For details, see *(Supplement)* below.) If you consider the change in performance due to the enabling of this function to be a problem, disable this function.

(Supplement) Compression of Java object size

When the object-pointer compression function is enabled, JavaVM compresses and manages the size of Java objects. There are multiple compression methods (hereinafter called *modes*). When JavaVM is started, it selects a mode depending on the execution environment status. From then on, JavaVM applies the mode selected when it was started to compress the size of Java objects.

The following two conditions are used to select a mode:

1. Total of the values specified for the Java heap area size and the explicit heap area size

2. Where the heap areas of 1 are allocated in the virtual address space

Due to these conditions, depending on whether you change the values specified for the heap area sizes when JavaVM is started and depending on the loading status of other processes or libraries running on the same OS, a different mode is applied each time JavaVM is started.

Which mode is selected depends on where the heap areas are actually allocated in the virtual memory space. For this reason, it is difficult to enable this function so that a specific compression mode is always applied. Note that not only does a performance difference occur when this function is switched between enabled and disabled but a performance difference might also occur each time JavaVM is started when this function is enabled.

As a reference, the following table[#1, #2] gives an overview of the selection conditions and compression methods for the modes that can be selected by JavaVM when the object-pointer compression function is enabled.

Table 9–25: List of modes for the object-pointer compression function

	Selection conditions	Overview
Mode 1	When the total size of each heap area (Java heap area and explicit heap area[#3]) is less than 4 GB and both heap areas exist at a lower position than 4 GB in the virtual address space	The simplest calculation method is used to compress the size of Java objects.
Mode 2	When mode 1 cannot be applied and both heap areas exist at a lower position than 32 GB in the virtual address space.	The next simplest calculation method after mode 1 is used to compress the size of Java objects.
Mode 3	When mode 1 and mode 2 could not be applied	The most complicated method of these modes is used to compress the size of Java objects.

#1

The content in the table shows the internal processing of JavaVM as a reference and is subject to change without prior notice due to subsequent JDK updates and modifications.

#2

In AIX, mode 3 is always selected because enough memory to select mode 1 or mode 2 in Table 9-25 cannot be secured due to platform-specific restrictions.

#3

For the explicit heap area, this condition applies only when the Explicit Memory Management functionality is used.

9.19 Incompatibility between Oracle JDK and the JDK provided by the Application Server

This section describes incompatibility between Oracle JDK and the JDK provided by the Application Server.

9.19.1 Memory management method selected by default

In Oracle JDK 9 or later, the memory management method selected by default was changed to the Garbage First Garbage Collector (G1GC). Note, however, that the memory management method provided by the Application Server is the Serial Garbage Collector (SerialGC), which is the same as the previous version, 09-70.

9.19.2 Runtime image

The Application Server does not support creation and reconfiguration of runtime images. In addition, it does not include the `jlink` command for creating runtime images.

9.19.3 Module-related options

The Application Server has module-related options that are not supported or that have restrictions.

(1) Non-supported options

The following options are not supported by the Application Server.

- `--limit-modules`
- `--module` (or `-m`)
- `--patch-module`
- `--upgrade-module-path`

For details about each option, see the following Web page.

`https://docs.oracle.com/en/java/javase/11/tools/java.html`

(2) Options having restrictions

The following options have restrictions. For these options, do not specify modules in the JDK[#] as modules to be updated.

- `--add-exports`
- `--add-opens`
- `--add-reads`

#

Modules in the JDK are all modules described on the following Web page.
`https://docs.oracle.com/javase/jp/11/docs/api/index.html`

For details about each option, see the following Web page.

`https://docs.oracle.com/en/java/javase/11/tools/java.html`

The following shows usage examples that are subject to the restrictions.

(Usage example 1)

`--add-exports=java.base/jdk.internal.jimage=my.module`

Do not specify the `java.base` module as a module to be updated.

(Usage example 2)

`--add-opens=java.base/jdk.internal.jimage=my.module`

Do not specify the `java.base` module as a module to be updated.

(Usage example 3)

`--add-reads=java.logging=my.module`

Do not specify the `java.logging` module as a module to be updated.

10 Migrating from Application Server of Earlier Versions (In the J2EE Server Mode) (INTENTIONALLY DELETED)

(INTENTIONALLY DELETED)

10.1 (INTENTIONALLY DELETED)

(INTENTIONALLY DELETED)

11 Migrating to the Recommended Functionality

This chapter describes the migration from the functionality that was supported until now by Application Servers to the recommended functionality.

11.1 Notes on migration to a database connection using HiRDB Type4 JDBC Driver

To migrate from a database connection using Cosminexus DABroker Library to a database connection using HiRDB Type4 JDBC Driver, see the description on the migration from DABroker for Java in the *HiRDB UAP Development Guide*.

11.2 Migration to a database connection using Oracle JDBC Thin Driver from DABroker Library

There are differences in specifications for the JDBC drivers of DABroker Library and Oracle JDBC Thin Driver. Therefore, when you change the Oracle connection method from DABroker Library to Oracle JDBC Thin Driver, set up the following properties in the `config-property` tag of the Hitachi Connector property file in order to ensure compatibility:

- `appendZero`

  Set the `appendZero` property to `true` in the Hitachi Connector property file.

  For details on the `appendZero` property, see *4.1.10 Properties that you can specify in the <config-property> tag set up for DB Connector* in the *uCosminexus Application Server Application and Resource Definition Reference Guide*.

- `forceFixedString`

  Set the `forceFixedString` property to `true` in the Hitachi Connector property file.

  For details on the `forceFixedString` property, see *4.1.10 Properties that you can specify in the <config-property> tag set up for DB Connector* in the *uCosminexus Application Server Application and Resource Definition Reference Guide*.

12 Migrating from Version 9 to Version 11

This chapter describes the migration from version 9 to version 11.

12.1 Overview

Application Server version 11 and Developer version 11 support new Java EE 7 specifications such as Servlet 3.1 and WebSocket 1.0, which support Hitachi Application Framework and other companies' frameworks that require Java EE 7 specifications.

To support Java EE 7 specifications, some functionality in version 11 was changed, becoming incompatible with version 9 or earlier. This chapter describes the functionality that was changed and the migration method for migrating from version 9 to version 11.

12.2 New functionality of version 11 and changes from version 9

This section describes the new functionality of version 11 and the changes from version 9.

12.2.1 NIO HTTP server functionality

Application Server version 11 has enhanced support for new functionality that mainly handles the asynchronous processing of web applications, such as asynchronous servlets of Servlet 3.0, asynchronous I/O APIs of Servlet 3.1, and WebSocket 1.0. Accordingly, the *Web server integration functionality using the redirector* and the *in-process HTTP server functionality*, which required conventional synchronous processing, were updated, and the *NIO HTTP server functionality*, which is the in-process HTTP server functionality newly implemented by using Java NIO technology, was installed.

The new functionality of Application Server version 11 requires non-blocking I/O of the NIO HTTP server functionality. For this reason, the conventional redirector functionality and in-process HTTP server functionality cannot be used. Requests to web applications on the J2EE server are unified to use HTTP communication by the NIO HTTP server functionality when going through the Web server or when directly accessing the J2EE server.

For this reason, you need to revise the system design supporting the redirector functionality and the in-process HTTP server functionality to a design supporting the NIO HTTP server.

For details, see *12.4 Migration guide for system design*.

Figure 12–1: When migrating the Web server integration configuration that uses the redirector to Application Server version 11

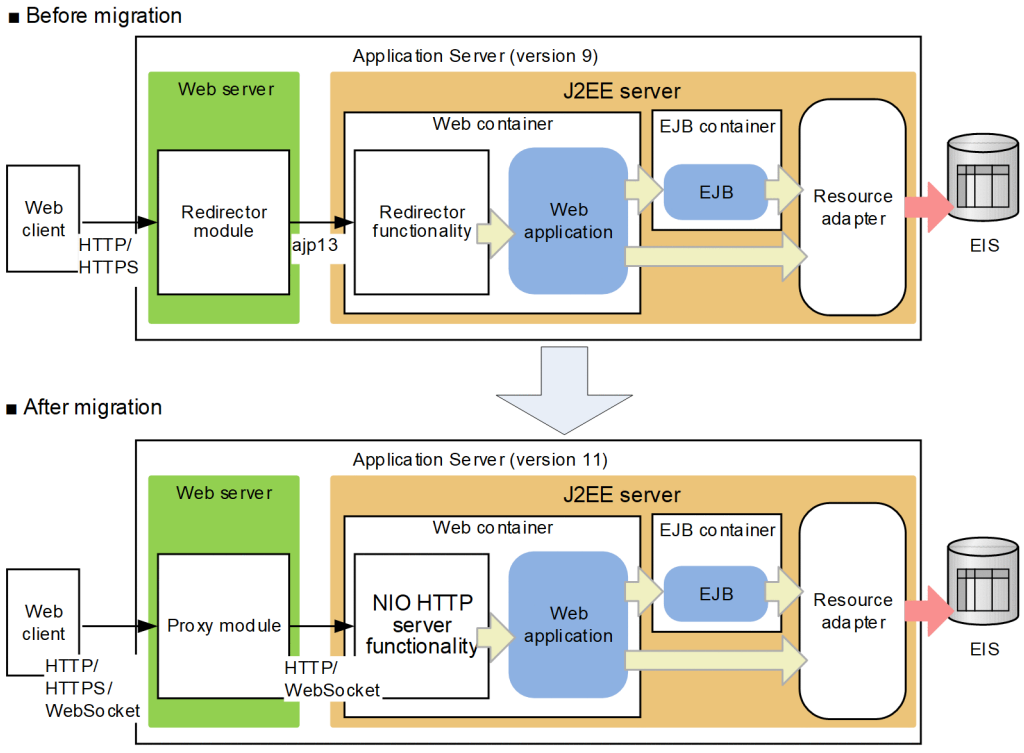
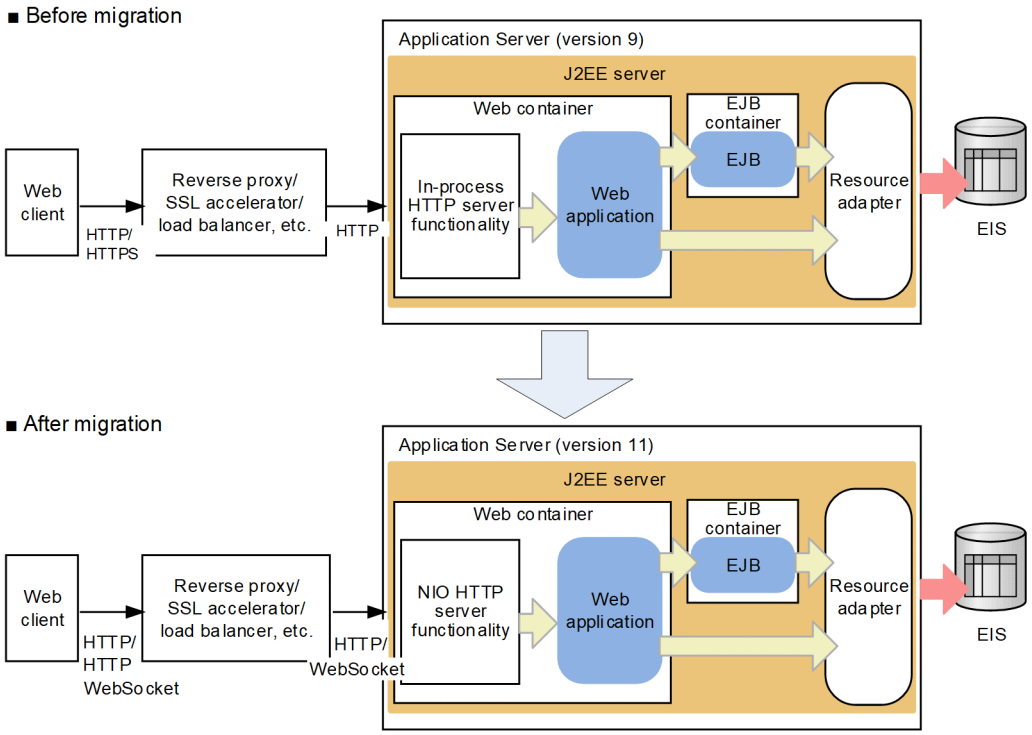


Figure 12–2: When migrating the in-process HTTP server configuration to Application Server version 11



12.2.2 Support for new Java EE 7 specifications

In Application Server version 11, the supported versions and scope of support for the following Java EE components were changed. For some specifications, you need to perform migration work on the application side. If the relevant specifications are used in an application, see the *Reference* column for each specification in the following table.

Table 12–1: Java EE 7 standard specifications supported by Application Server version 11

Specification name	Version supported by Application Server		Reference
	V9	V11	
Java Servlet (Servlet)	3.0	3.1	12.3.2
Contexts and Dependency Injection for Java (CDI)	1.0	1.2	12.3.3
Java API for RESTful Web Services (JAX-RS)	1.1	2.0	12.3.4
Java Persistence API (JPA)	1.0	2.1[#1]	12.3.5
Java Server Faces (JSF)	2.1	2.2	12.3.6
Expression Language (EL)	2.2	3.0	No migration required
Bean Validation (BV)	1.0	1.1	
Common Annotations for the Java Platform (Common Annotations)	1.1	1.2	
Java Transaction API (JTA)	1.1	1.2[#2]	
Batch Applications for the Java Platform (JavaBatch)	--	1.0	

Specification name	Version supported by Application Server		Reference
	V9	V11	
Java API for JSON Processing (JSON-P)	--	1.0	
Java API for WebSocket (WebSocket)	--	1.0	
Concurrency Utilities for Java EE	--	1.0#3	

Note

Specifications that have not changed from Application Server version 9 are not included in this table.

#1

Only the JPA provider is provided and the JPA container function cannot be used. For this reason, you cannot use container-managed persistence units or persistence context.

#2

In addition to the functionality of version 9 and earlier, `@javax.transaction.Transactional` annotation is additionally supported.

#3

The ContextService function cannot be used.

12.2.3 V9 compatibility mode

For systems that prioritize compatibility with version 9 or earlier and do not use new functionality and for uCosminexus Service Platform users, Application Server version 11 provides *V9 compatibility mode*, which restores the operation of Application Server to specifications equivalent to those in version 9

For details about V9 compatibility mode, see the *uCosminexus Application Server Compatibility Guide*.

Note that this chapter assumes that users will not use V9 compatibility mode.

12.2.4 Functions not supported in version 11

Application Server version 11 has functions that do not support migration from version 9. With a few exceptions, these functions can be used in V9 compatibility mode.

(1) Non-supported web container functions

(a) Request distribution by using the round-robin method

The proxy module built into the Web server of Application Server version 11 does not support request distribution (load balancer) by using the round-robin method, which was supported through Web server integration by using the conventional redirector.

To achieve request distribution by using the round-robin method, a load balancer or a Web server that has a load balancer function is separately required.

(b) Request distribution by using the POST data size

The proxy module built into the Web server of Application Server version 11 does not support request distribution by using the POST data size, which was supported through Web server integration by using the conventional redirector.

To achieve request distribution by using the POST data size, a load balancer that has that function or a Web server that has a load balancer function is separately required.

(c) Gateway specification functionality by the J2EE server

The NIO HTTP server of Application Server version 11 does not have the *gateway specification functionality*, which was provided by the conventional redirector and the in-process HTTP server.

If the URL of a gateway such as a load balancer or an SSL accelerator needs to be returned to the client as the URL to be added to the Location header when the URL is transferred, specify settings to return the URL of the appropriate gateway, for example, by rewriting the Location header by using the function of the load balancer or SSL accelerator.

(d) Error page customization function for each web container

The NIO HTTP server of Application Server version 11 does not have the *error page customization function for each web container*, which was provided by the conventional redirector and the in-process HTTP server.

If you want to return a user-specific error page corresponding to the error status code as a response, define it for each application by following the Servlet specifications, or configure Web server integration via the reverse proxy function of the Web server to customize the error page by using the Web server functionality.

(e) Function that limits HTTP-enabled methods

The NIO HTTP server of Application Server version 11 does not have the *function that limits HTTP-enabled methods*, which was provided by the conventional in-process HTTP server.

If you need to limit HTTP-enabled methods, configure Web server integration via the reverse proxy function of the Web server to limit HTTP-enabled methods by using the Web server functionality.

(f) Request distribution functionality that uses the redirector

The NIO HTTP server of Application Server version 11 does not have the *request distribution functionality that uses the redirector*, which was provided by the conventional in-process HTTP server.

If you need to redirect each URL, configure Web server integration via the reverse proxy function of the Web server to redirect URLs by using the Web server functionality.

(2) Non-supported extended functionality

Application Server version 11 does not support the *EADs session failover functionality*. This functionality cannot be used even in V9 compatibility mode.

(3) Non-supported compatibility functionality

The following compatibility functionality, which was retained in Application Server version 9 and earlier for compatibility with a previous version, cannot be used in Application Server version 11. The functionality cannot be used even in V9 compatibility mode.

Table 12–2: Compatibility functionality that is no longer supported in Application Server version 11

Category	Functionality	Alternative functionality
Compatibility functionality	Basic mode	Use the J2EE server.
	Servlet engine mode (Web container server)	Use the J2EE server.
	Simple web server functionality	Use the NIO HTTP server functionality. For details, see the manual *uCosminexus Application Server Web Container Functionality Guide*.
	Subdirectory exclusive mode for EJB client application logs	Use the default subdirectory shared mode.
	Memory session failover functionality	Use the database session failover functionality. For details, see *6. Database session failover functionality* in the *uCosminexus Application Server Expansion Guide*.
	Switching of multiple built execution environments	None
	Test function for J2EE applications	None
	Check of JSP source compliant with JSP 1.1 and JSP 1.2 specifications (cjjsp2java)	None
	V7 compatibility mode for logical server application management	None
Security management functionality	Inheritance of the login status by using the session failover functionality	Use the database session failover functionality. For details, see *6. Database session failover functionality* in the *uCosminexus Application Server Expansion Guide*.
Command	Commands used by the Management Server (commands for compatibility with previous versions) • `cmx_define_application` command • `cmx_deploy_application` command • `cmx_register_application` command • `cmx_start_application` command • `cmx_stop_application` command • `cmx_undefine_application` command • `cmx_undeploy_application` command • `cmx_unregister_application` command • `cmx_define_resource` command • `cmx_deploy_resource` command • `cmx_register_resource` command • `cmx_start_resource` command • `cmx_stop_resource` command • `cmx_undefine_resource` command • `cmx_undeploy_resource` command • `cmx_unregister_resource` command • `cmx_add_serverref` command • `cmx_delete_serverref` command	Use commands of the J2EE server. For details, see *2. Commands Used with a J2EE Server* in the *uCosminexus Application Server Command Reference Guide*.

12.3 Application migration guide

This section describes the procedure for migrating user applications.

12.3.1 Migration to alternative functionality

You might need to migrate the functionality used by Application Server version 9 and Developer version 9 to alternative functionality so that the functionality can operate in Application Server version 11 and Developer version 11. Check the following table, and migrate applications as needed. Functionality not included in the table is compatible with Application Server version 9 and Developer version 9.

Table 12–3: Cases in which application migration is required

Functionality	Case in which migration is required	Reference
Servlet	If the application uses non-supported Servlet 3.0 APIs and the `UnsupportedOperationException` exception is expected to be thrown	12.3.2(1)
	If the application uses the dynamic servlet definition API to invoke the `addMapping` method of `ServletRegistration` to map to the context root (URL pattern is `/`), but the mapping is not expected to be performed correctly	12.3.2(2)
CDI	If the application uses CDI annotations, but CDI is not expected to operate without `beans.xml` being stored in the application archive	12.3.3(1)
	If a library having a Portable Extension implementation class is added to the class path or stored in the application, but the Portable Extension is not expected to operate	12.3.3(2)
JAX-RS	If the application was using the Cosminexus JAX-RS engine (if `cjjaxrs.jar` was added to the class path)	12.3.4(1)
	If the application was using parameters unique to the Cosminexus JAX-RS engine	12.3.4(2)
	If the application was using client APIs for RESTful Web Services unique to the Cosminexus JAX-RS engine	12.3.4(3)
JPA	If the application was using parameters unique to the CJPA provider (User properties that start with `cosminexus.jpa`, or values specified in `<property>` tags in `persistence.xml`)	12.3.5(1)
	If the application was using the persistence units or persistence context for container management. This case applies when one of the following is used: • The `javax.persistence.PersistenceUnit` annotation • The `javax.persistence.PersistenceContext` annotation • The `<persistence-unit-ref>` element of DD • The `<persistence-context-ref>` element of DD	12.3.5(2)
	If the application was using a database other than HiRDB for the persistence destination database	12.3.5(3)
JSF	If the application was using JSF (if `cjsf.jar` was added to the class path)	12.3.6(1)

12.3.2 Changes in servlets

(1) Handling of non-supported Servlet 3.0 APIs

Application Server version 11 and Developer version 11 now support some Servlet 3.0 APIs that were not supported in version 9. For this reason, when the relevant APIs are used, they operate according to the Servlet 3.0 specifications in Application Server version 11 and Developer version 11 even though the `UnsupportedOperationException` exception was thrown in Application Server version 9.

The relevant APIs are listed in the following table. If the application always expects the `UnsupportedOperationException` exception to be thrown from these APIs, you need to modify the application so that the implementation is based on the assumption that no exception will be thrown.

Table 12–4: Servlet 3.0 APIs that are supported in Application Server version 11 and Developer version 11

Package	Class	Method
`javax.servlet`	`AsyncContext`	All methods
	`AsyncListener`	All methods
	`ServletRequest`	`startAsync`
		`isAsyncStarted`
		`isAsyncSupported`
		`getAsyncContext`
		`getDispatcherType`
	`AsyncEvent`	All methods
	`ServletRequestWrapper`	`startAsync`
		`isAsyncStarted`
		`isAsyncSupported`
		`getAsyncContext`
		`getDispatcherType`
	`Registration`	`setAsyncSupported`

In addition, for APIs that remain unsupported in Application Server version 11 and Developer version 11, no exception will be thrown by default in Application Server version 11 and Developer version 11 even though the `UnsupportedOperationException` exception was thrown in Application Server version 9 and Developer version 9. The relevant APIs are listed in the following table.

Table 12–5: Servlet 3.0 APIs that are not supported in Application Server version 11 and Developer version 11

Package	Class	Method	Operation for the value specified for webserver.servlet_api.unsupported.throwUnsupportedOperationException	
			false (default)	true
javax.servlet	ServletContext	getJspConfigDescriptor	Returns null.	Throws the UnsupportedOperationException exception.

If the application always expects the `UnsupportedOperationException` exception to be thrown from these APIs, modify the application so that the implementation is based on the assumption that no exception will be thrown, or add the following definition to the user property file for J2EE servers.

```
webserver.servlet_api.unsupported.throwUnsupportedOperationException=true
```

(2) Processing to dynamically add a servlet that maps to the context root

In Application Server version 9 and Developer version 9, if you specified / to indicate the context root for the argument of the `addMapping` method of `javax.servlet.ServletRegistration`, the mapping could not be overwritten because the default servlet was already mapped.

In Application Server version 11 and Developer version 11, you can overwrite the mapping to the context root only once for the same ServletContext. This enables you to map a user-defined servlet (instead of the default servlet) to the context root.

If the application does not expect Application Server version 11 and Developer version 11 to operate accordingly but expects the `addMapping` method to refuse to overwrite the mapping, you need to modify the application so that the `addMapping` method that performs mapping to the context root is not invoked. If you migrate to Application Server version 11 and Developer version 11 without modifying the application, the mapping to the default servlet might be overwritten with mapping to a user servlet.

12.3.3 Changes in CDI

(1) Changes in the conditions to determine whether CDI is enabled when beans.xml is omitted

In Application Server version 11 and Developer version 11, the handling of the `beans.xml` file and the conditions to determine whether CDI is enabled were changed in association with the support for the CDI 1.2 specifications.

Under the CDI 1.2 specifications, CDI is automatically enabled when the CDI annotation is determined to be used even if the application archive does not include `beans.xml`. For this reason, if an application that could be deployed in Application Server version 9 and Developer version 9 does not expect CDI to be enabled, it might fail to be deployed in Application Server version 11 and Developer version 11.

The CDI 1.2 specifications stipulate to provide an option for restoring the determination conditions to the CDI 1.0 criteria to avoid such a situation. Application Server version 11 and Developer version 11 also provide that option. To restore the conditions to determine whether CDI is enabled to the CDI 1.0 criteria, add the following definition to the user property file for J2EE servers.

```
ejbserver.javaee.cdi.beansXmlRequired=true
```

(2) Changes in the handling of Portable Extension APIs

In Application Server version 9 and Developer version 9, CDI's Portable Extension APIs were not supported, and ignored even if the implementation class of Portable Extension existed in the class path.

Although Application Server version 11 and Developer version 11 continue to not support the use of Portable Extension APIs by user applications, the implementation class of Portable Extension runs because CDI's Portable Extension APIs are used within the product.

For the JAR file that stores the implementation class of Portable Extension (a class that implements the `javax.enterprise.inject.spi.Extension` interface) and its service definition file (`META-INF/services/javax.enterprise.inject.spi.Extension` file), do not add it to the class path of the J2EE server and do not store it in the application. Failure to follow this instruction might cause the implementation class of Portable Extension to run, causing the application to fail to be deployed or changing the behavior of the application.

12.3.4 Changes in JAX-RS

(1) Change in the JAX-RS engine

Application Server version 11 and Developer version 11 newly provide the JAX-RS 2.0 engine, which supports JAX-RS 2.0. This engine is stored in a different JAR file from the conventional JAX-RS engine of Application Server version 9 or earlier and of Developer version 9 or earlier. For this reason, the class path settings required to use the JAX-RS engine are different.

For Application Server version 9 or earlier and Developer version 9 or earlier

```
add.class.path=<cosminexus.home>\jaxrs\lib\cjjaxrs.jar
```

For Application Server version 11 or later and Developer version 11 or later

```
add.class.path=<cosminexus.home>\CC\javaee\1100\lib\jaxrs-impl.jar
add.class.path=<cosminexus.home>\CC\javaee\1100\lib\jaxrs-jackson.jar
```

If the JAX-RS engine (`cjjaxrs.jar`) of Application Server version 9 or earlier and of Developer version 9 or earlier is still set for the class path, the application will not operate normally. Be sure to revise the class path settings.

(2) Abolition of Cosminexus's original parameters for the JAX-RS engine

The JAX-RS 2.0 engine of Application Server version 11 and Developer version 11 does not support Cosminexus's self-defined files and parameters, which were supported by the JAX-RS engine of Application Server version 9 or earlier and of Developer version 9 or earlier. The definition files and parameters described in *13.1 Action definition file* of the *uCosminexus Application Server Web Service Development Guide* cannot be used in Application Server version 11 or Developer version 11.

(3) Abolition of the client APIs for RESTful Web Services

The JAX-RS 2.0 specifications newly standardized the client APIs. As a result, Cosminexus's original client APIs for RESTful Web Services, which were supported by the JAX-RS engine of Application Server version 9 or earlier and of Developer version 9 or earlier, are not supported. The functions described in *25. Support Range of the Client APIs for RESTful Web Services* of the *uCosminexus Application Server Web Service Development Guide* cannot be used in Application Server version 11 or Developer version 11.

For applications that use the client APIs of JAX-RS, replace those APIs with JAX-RS 2.0 standard APIs by following the JAX-RS 2.0 specifications and the JAX-RS 2.0 API specifications.

JAX-RS 2.0 API specifications (provided by Oracle)
`https://docs.oracle.com/javaee/7/api/javax/ws/rs/client/package-summary.html`

12.3.5 Changes in JPA

(1) Abolition of the CJPA provider

The conventional CJPA provider cannot be used because it does not support JPA 2.1 APIs. Instead, Application Server version 11 and Developer version 11 newly provide a JPA provider that supports JPA 2.1. However, Cosminexus's original functions provided by the conventional CJPA provider cannot be used. The following parameters are ignored even if they are specified.

- Properties starting with `cosminexus.jpa` specified in the `<property>` tag in `persistence.xml`
- Keys starting with `cosminexus.jpa` specified in the user property file for J2EE servers
- Parameters starting with `cosminexus.jpa` specified for the logical J2EE server by the Smart Composer functionality

(2) Persistence units and persistence context of container management cannot be used

The JPA 2.1 support scope for Application Server version 11 and Developer version 11 does not support the JPA container functionality. For this reason, applications that assume the use of the following APIs and DD elements to obtain the persistence units and persistence context for container management cannot be migrated.

- The `javax.persistence.PersistenceUnit` annotation
- The `javax.persistence.PersistenceContext` annotation
- The `<persistence-unit-ref>` element of DD
- The `<persistence-context-ref>` element of DD

In addition, the following parameters cannot be specified:

- Keys starting with `ejbserver.jpa` specified in the user property file for J2EE servers
- Parameters starting with `ejbserver.jpa` specified for the logical J2EE server by the Smart Composer functionality

Acquisition of the EntityManagerFactory object can be substituted with the persistence unit for application management. In accordance with the JPA 2.1 specifications, obtain the EntityManagerFactory object by using the following implementation example.

```
EntityManagerFactory factory
= Persistence.createEntityManagerFactory("persistence-unit-name");
EntityManager manager = factory.createEntityManager();
```

(3) Changes in the database supported as the persistence destination

In the JPA 2.1 support scope for Application Server version 11 and Developer version 11, only the following database can be used as the persistence destination:

- HiRDB version 10

If persistence to other databases is required, prepare a JPA provider library# corresponding to JPA 2.1 and the persistence destination database, and add it to the class path instead of the Cosminexus JPA 2.1 provider.

#

EclipseLink, Hibernate JPA, or some other library

12.3.6 Changes in JSF

(1) Change in the JSF library

Application Server version 11 and Developer version 11 now support JSF 2.2. The library for JSF 2.2 is stored in a different JAR file from the conventional library for JSF 2.1 of Application Server version 9 or earlier and of Developer version 9 or earlier, and is added to the class path by default. For this reason, you no longer need to add the following class paths when using JSF.

For Application Server version 9 or earlier and Developer version 9 or earlier

```
add.class.path=<cosminexus.home>\CC\lib\cjsf.jar
add.class.path=<cosminexus.home>\CC\lib\cjstl.jar
```

For Application Server version 11 and Developer version 11

You do not need to add class paths.

If the library for JSF 2.1 (`cjsf.jar`) of Application Server version 9 or earlier and Developer version 9 or earlier is still set for the class path, the application will not operate normally. Be sure to revise the class path settings.

12.4 Migration guide for system design

This section describes the changes in system design such as tuning parameters and the formula for estimating resources.

For details about the unit, default value, and specifiable range for each parameter, see the *uCosminexus Application Server Definition Reference Guide*.

12.4.1 Performance tuning

In Application Server version 11 and Developer version 11, the following viewpoints were changed from the viewpoints of performance tuning of the J2EE application execution platform:

- Optimization of the number of concurrent executions
- Timeout setting

(1) Optimization of the number of concurrent executions

In Application Server version 9 or earlier and Developer version 9 or earlier, a request processing thread of the Web container was always assigned to a connection and the request processing thread was occupied until the response body was sent to the client.

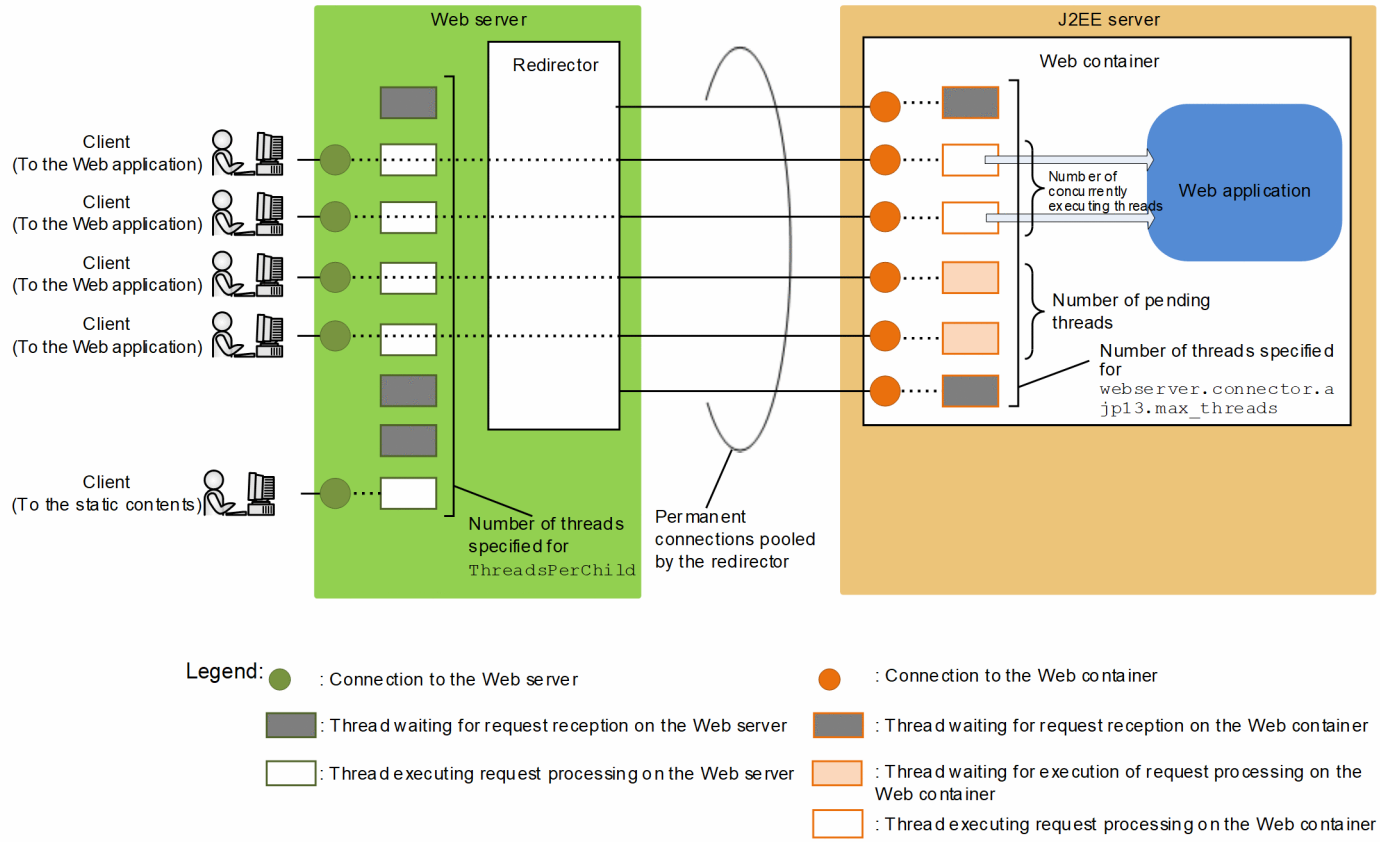
Application Server version 11 or later and Developer version 11 or later support the standard specifications that are premised on non-blocking I/O such as asynchronous servlets of Servlet 3.0, asynchronous I/O APIs of Servlet 3.1, and WebSocket. For this reason, management of the connections that receive requests and the threads that process received requests are separated to enable a connection to use multiple processing threads and enable processing that receives a request and processing that sends a response to be processed by separate threads.

As a result, Application Server version 11 and Developer version 11 have a different approach to control of the number of concurrent executions from that of version 9 or earlier. The following describes the differences in the points to set control of the number of concurrent executions for each Web server configuration.

(a) When migrating from the Web server integration configuration using the redirector

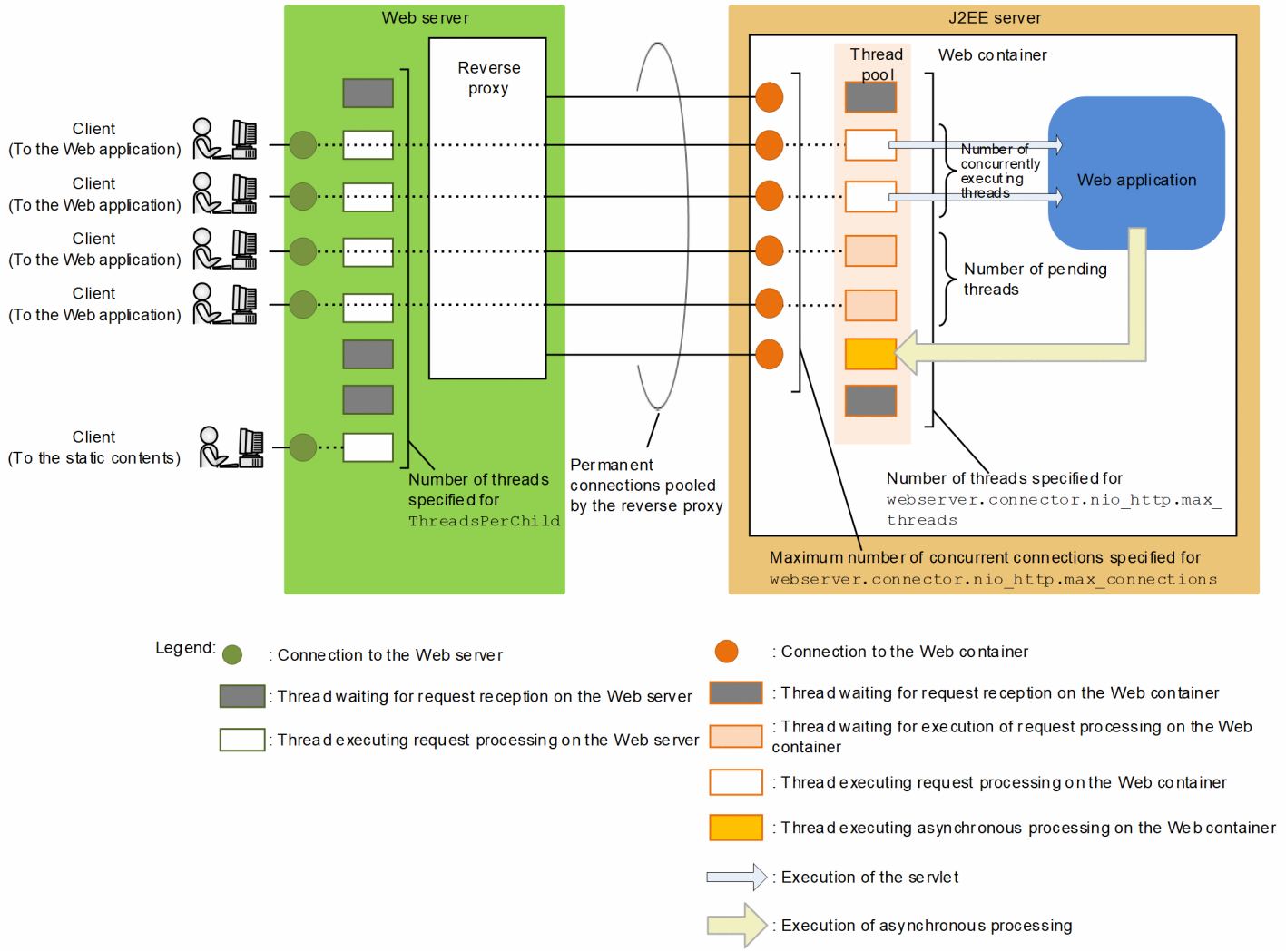
The redirector functionality used in Application Server version 9 or earlier and Developer version 9 or earlier used permanent connections to connect the Web server and the Web container, and occupied a request processing thread for each connection on the Web container. For this reason, the number of threads to be generated is always the same as the maximum number of connections that can be concurrently connected. In addition, by controlling the number of concurrently executing threads to control the threads that could be concurrently executed among the generated threads, you could suppress performance degradation and resource depletion. The following figure shows the mechanism of controlling the number of threads in Application Server version 9 and Developer version 9.

Figure 12–3: Mechanism of controlling the number of threads in Application Server version 9 and Developer Version 9 (for Web server integration)



With the NIO HTTP server functionality of Application Server version 11 and Developer version 11, even if permanent connections are used to connect the Web server and the Web container, the request processing threads on the Web container are not occupied by the connections. Each time the Web container detects the reception of data, it assigns a thread from the thread pool. In addition, by controlling the number of concurrently executing threads to control the threads that can be concurrently executed among the threads assigned to requests for processing a request of the servlet, you can suppress performance degradation and resource depletion in the same way as Application Server version 9 and Developer version 9. The following figure shows the mechanism of controlling the number of threads in Application Server version 11 and Developer version 11.

Figure 12–4: Mechanism of controlling the number of threads in Application Server version 11 and Developer version 11 (for Web server integration)



The NIO HTTP server functionality allows you to define the number of threads to be generated in advance in the thread pool when the J2EE server starts and the maximum number of threads to be dynamically generated. The following table describes the recommended values set for the number of threads when you migrate from Application Server version 9 and Developer version 9 in the Web server integration configuration. Note, however, that this does not apply when you use the WebSocket functionality. For details, see *(c) Notes when increasing the maximum number of concurrent connections to use the WebSocket functionality*.

Table 12–6: Recommended values set for the number of concurrent executions (for migration from a Web server integration configuration)

Parameter of the setting destination NIO HTTP server	Recommended setting value	
	When using only the functionality up to Application Server version 9 and Developer version 9	When using asynchronous processing supported in Application Server version 11 and Developer version 11
`webserver.connector.nio_http.max_threads`	The same value as `webserver.connector.ajp13.max_threads`	Value of `webserver.connector.ajp13.max_threads` + Maximum number of concurrent executions of asynchronous processing[#1] to be concurrently executed in excess of the number of connections

Parameter of the setting destination NIO HTTP server	Recommended setting value	
	When using only the functionality up to Application Server version 9 and Developer version 9	When using asynchronous processing supported in Application Server version 11 and Developer version 11
`webserver.connector.nio_http.min_threads`	The same value as `webserver.connector.ajp13.max_threads`	
`webserver.connector.nio_http.max_connections`	Total number of connections to the J2EE server#2	

#1

Asynchronous processing to be concurrently executed in excess of the number of connections includes the following:

- Invoking the `start` method of `javax.servlet.AsyncContext` of Servlet 3.0
- Invoking the `sendBinary` method, the `sendObject` method, and the `sendText` method of the `javax.websocket.RemoteEndpoint.Async` interface of WebSocket

#2

Total number of connections to the J2EE server is the sum of the number of connections to the NIO HTTP server on the J2EE server. If multiple processes connect to the J2EE server, set the sum of the maximum number of connections from each process.

For details on the maximum number of connections from the reverse proxy of the Cosminexus HTTP Server to the J2EE server, see *4.7.4(5) Points to be noted for performance* in the *uCosminexus Application Server HTTP Server User Guide*.

(b) When migrating from the in-process HTTP server configuration

The in-process HTTP server functionality used in Application Server version 9 or earlier and Developer version 9 or earlier assigned a thread generated in advance in the thread pool to each connection each time a connection request was sent from the client, and occupied that thread until the connection with the client is disconnected. In addition, by controlling the number of concurrently executing threads to control the threads that can be concurrently executed among the generated threads, you could suppress performance degradation and resource depletion.

In the NIO HTTP server functionality of Application Server version 11 and Developer version 11, the request processing thread on the Web container is not occupied by the connection, and the Web container assigns a thread from the thread pool each time it detects the reception of data. In addition, by controlling the number of concurrently executing threads to control the threads that can be concurrently executed among the threads assigned to the requests for processing a request of the servlet, you can suppress performance degradation and resource depletion in the same way as Application Server version 9 and Developer version 9.

The NIO HTTP server functionality allows you to define the number of threads to be generated in advance in the thread pool when the J2EE server starts and the maximum number of threads to be dynamically generated. The following table describes the recommended values set for the number of threads when you migrate from Application Server version 9 and Developer version 9 in the in-process HTTP server configuration. Note, however, that this does not apply when you use the WebSocket functionality. For details, see *(c) Notes when increasing the maximum number of concurrent connections to use the WebSocket functionality*.

Table 12–7: Recommended values set for the number of concurrent executions (for migration from an in-process HTTP server configuration)

Parameter of the setting destination NIO HTTP server	Recommended setting value in Application Server version 11 and Developer version 11	
	When using only the functionality up to Application Server version 9 and Developer version 9	When using asynchronous processing supported in Application Server version 11 and Developer version 11
`webserver.connector.nio_http.max_threads`	The same value as `webserver.connector.inprocess_http.max_connections`	Value of `webserver.connector.inprocess_http.max_connections` + Maximum number of concurrent executions of asynchronous processing# to be concurrently executed in excess of the number of connections
`webserver.connector.nio_http.min_threads`	The smallest value of `webserver.connector.inprocess_http.max_connections`, `webserver.connector.inprocess_http.max_spare_threads`, and `webserver.connector.inprocess_http.init_threads`	
`webserver.connector.nio_http.max_connections`	The same value as `webserver.connector.inprocess_http.max_connections`	

#

Asynchronous processing to be concurrently executed in excess of the number of connections includes the following:

- Invoking the `start` method of `javax.servlet.AsyncContext` of Servlet 3.0
- Invoking the `sendBinary` method, the `sendObject` method, and the `sendText` method of the `javax.websocket.RemoteEndpoint.Async` interface of WebSocket

(c) Notes when increasing the maximum number of concurrent connections to use the WebSocket functionality

For the NIO HTTP server functionality of Application Server version 11 and Developer version 11, the number of threads and the maximum number of concurrent connections do not necessarily have to be the same value. Even if the number of threads is smaller than the number of concurrent connections, no pending processing will occur if the maximum number of concurrently executing threads, which concurrently execute processing to receive data, is within the maximum number of threads. In addition, even if pending processing occurs because no thread can be assigned, the processing is queued in the infinite-length pending queue and the processing to receive data will resume when a thread becomes free.

If you use the WebSocket functionality added in Application Server version 11 and Developer version 11, you might need to increase the maximum number of concurrent connections to maintain connections with many WebSocket clients for a long time. In Application Server version 9 or earlier and Developer version 9 or earlier, for both Cosminexus HTTP Server and the in-process HTTP server, the maximum number of concurrent connections was limited to 1024. However, in Application Server version 11 and Developer version 11, the upper limit was increased in consideration of WebSocket support.

However, if you increase the maximum number of threads for the NIO HTTP server to match the maximum number of concurrent connections, the memory usage also increases in proportion to the number of threads. In addition, the number of threads that can be generated by a process is limited by the OS. If the memory capacity that can be secured for the minimum number of threads or the upper limit of the number of threads of the OS is insufficient, `OutOfMemoryError` might occur during the processing to start the J2EE server, causing the J2EE server to fail to start. If the memory capacity that can be secured for the maximum number of threads or the upper limit of the number of threads of the OS is insufficient, `OutOfMemoryError` might occur during the processing to receive requests, causing the J2EE server process to go down.

If you want to increase the minimum or maximum number of threads for the NIO HTTP server, appropriately re-estimate the memory size (physical memory size, the value specified for `-Xmx` of the J2EE server process) as well.

For details about how to estimate the maximum number of threads, see *5.2.1 Estimating the resources used by J2EE server* in the *uCosminexus Application Server System Design Guide*. For details about memory usage, see *5.3.1 Estimating virtual memory usage of the J2EE server* in the *uCosminexus Application Server System Design Guide*.

(2) Timeout setting

In Application Server version 11 or later and Developer version 11 or later, the locations where a timeout occurs and the parameter for which the timeout value is set are different from those in Application Server version 9 or earlier and Developer version 9 or earlier. The following describes the differences in the timeout setting points for each Web server configuration.

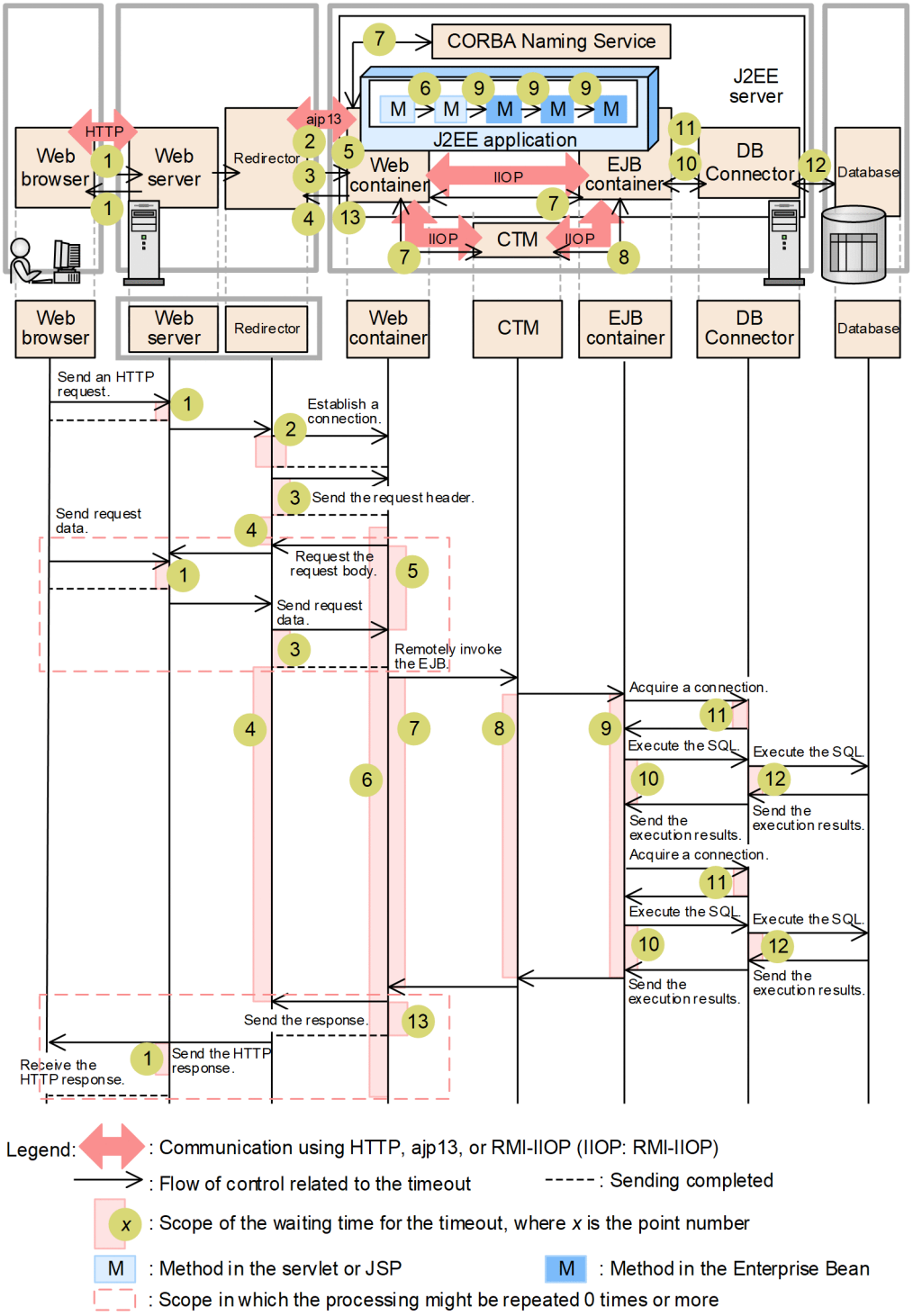
(a) When migrating from the Web server integration configuration using the redirector

In Application Server version 11 and Developer version 11, the redirector module (`mod_jk`) changes to the proxy module (`mod_proxy`) and the request receiver on the Web container changes to the NIO HTTP server.

For this reason, the timeouts set for the redirector and the Web container in Application Server version 9 and Developer version 9 change to the timeouts set for the reverse proxy and the NIO HTTP server in Application Server version 11 and Developer version 11.

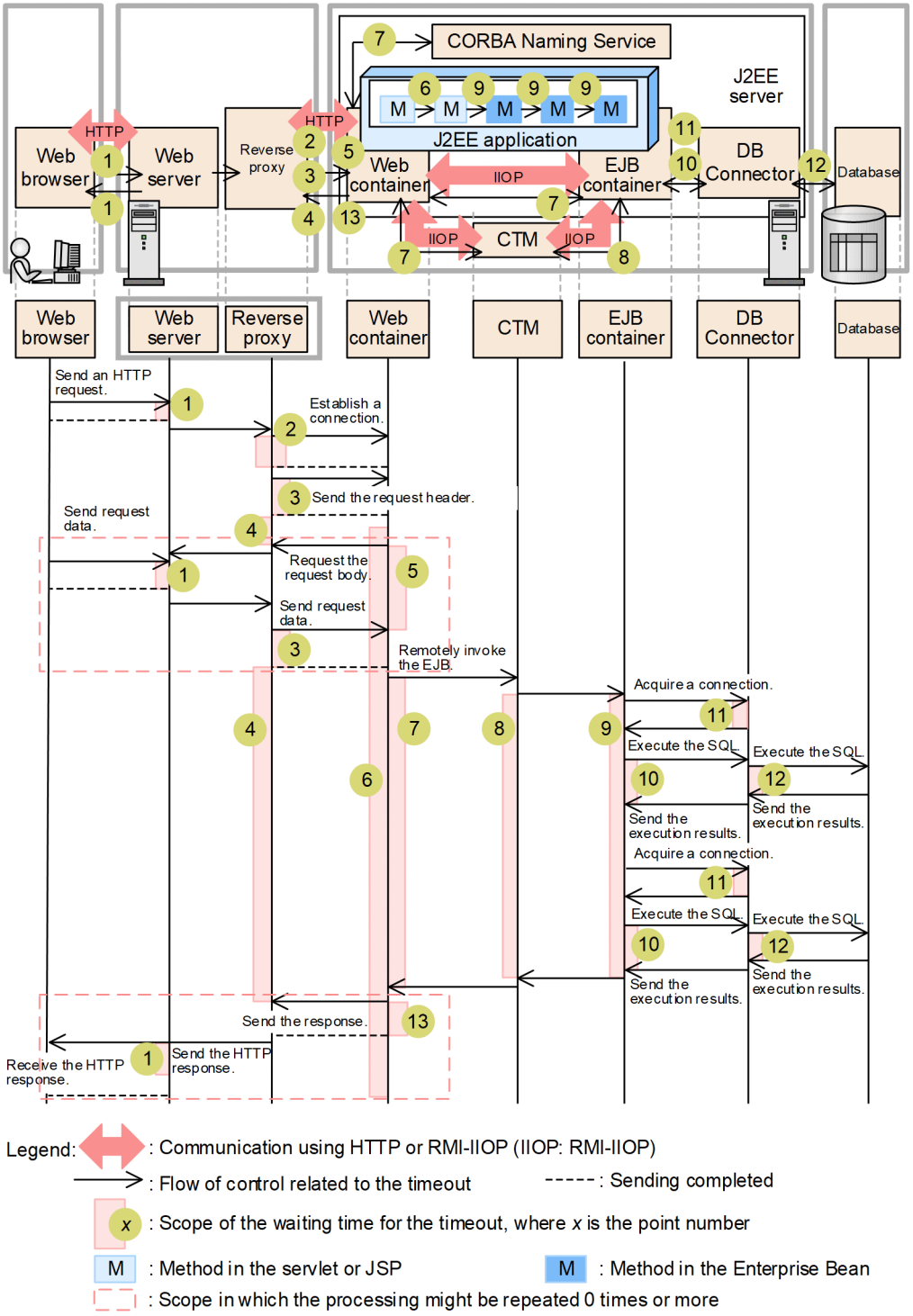
The following figure shows the points for which a timeout can be set in Application Server version 9 and Developer version 9.

Figure 12–5: Points for which a timeout can be set in Application Server version 9 and Developer version 9 (for Web server integration)



The following figure shows the points for which a timeout can be set in Application Server version 11 and Developer version 11.

Figure 12–6: Points for which a timeout can be set in Application Server version 11 and Developer Version 11 (for Web server integration)



The following table describes the differences for each point that needs to be migrated from Application Server version 9 and Developer version 9 to Application Server version 11 and Developer version 11.

Table 12–8: Difference in the timeout set for each point (migration from the Web server integration configuration)

Point	Timeout type	
	Application Server version 9 and Developer version 9	Application Server version 11 and Developer version 11
2	A timeout for establishing a connection to the Web container, which is set on the redirector. Setup target • `mod_jk.conf` (redirector action definition file) • Logical Web server (`web-server`) Location of setup `JkConnectTimeout`	A timeout for establishing a connection to the Web container (NIO HTTP server), which is set on the reverse proxy. Setup target • `httpsd.conf` • Logical Web server (`web-server`) Location of setup The `connectiontimeout` key of `ProxyPass`, the `timeout` key of `ProxyPass`, or `Timeout`
3	A timeout for sending the request header and the request body to the Web container, which is set on the redirector. Setup target • `mod_jk.conf` (redirector action definition file) • Logical Web server (`web-server`) Location of setup `JkSendTimeout`	A timeout for sending the request header and the request body to the Web container (NIO HTTP server), which is set on the reverse proxy. Setup target • `httpsd.conf` • Logical Web server (`web-server`) Location of setup The `timeout` key of `ProxyPass`, or `Timeout`
4	A timeout for receiving data from the Web container, which is set on the redirector. Setup target • `worker.properties` (worker definition file) • Logical Web server (`web-server`) Location of setup `worker.`*worker-name*`.receive_timeout`	A timeout for receiving data from the Web container (NIO HTTP server), which is set on the reverse proxy. Setup target • `httpsd.conf` • Logical Web server (`web-server`) Location of setup The `timeout` key of `ProxyPass`, or `Timeout`
5	A timeout for receiving data from the redirector, which is set on the Web container. Setup target • `usrconf.properties` (user property file for J2EE servers) • Logical J2EE server (`j2ee-server`) Location of setup `webserver.connector.ajp13.receive_timeout`	A timeout for receiving data from the reverse proxy, which is set on the Web container (NIO HTTP server). Setup target • `usrconf.properties` (user property file for J2EE servers) • Logical J2EE server (`j2ee-server`) Location of setup `webserver.connector.nio_http.receive_timeout`
13	A timeout for sending the response to the redirector, which is set on the Web container. Setup target • `usrconf.properties` (user property file for J2EE servers) • Logical J2EE server (`j2ee-server`) Location of setup `webserver.connector.ajp13.send_timeout`	A timeout for sending data to the reverse proxy, which is set on the Web container (NIO HTTP server). Setup target • `usrconf.properties` (user property file for J2EE servers) • Logical J2EE server (`j2ee-server`) Location of setup `webserver.connector.nio_http.send_timeout`

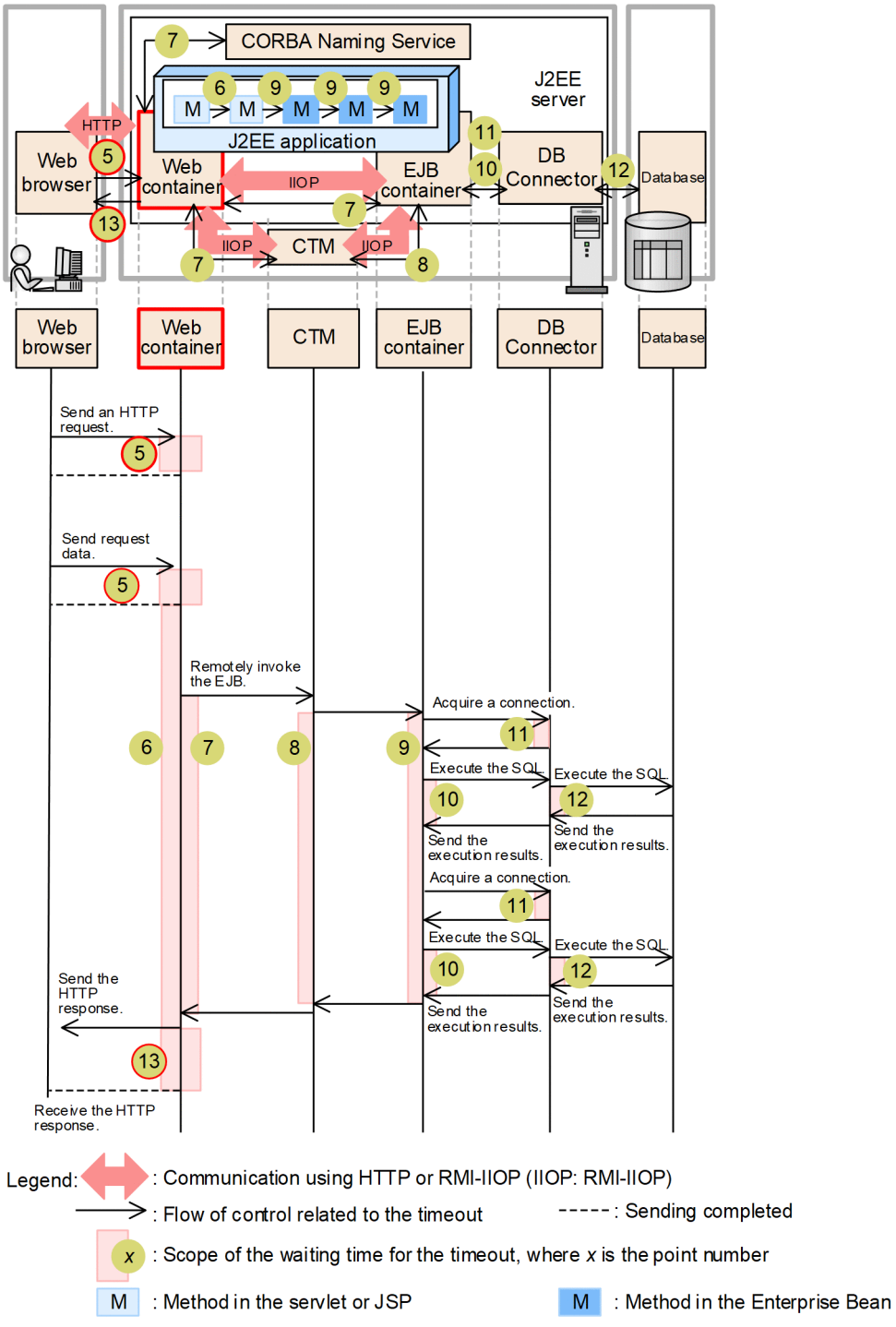
(b) When migrating from the in-process HTTP server configuration

In Application Server version 11 and Developer version 11, the in-process HTTP server is replaced with the NIO HTTP server.

For this reason, the timeouts set for the in-process HTTP server in Application Server version 9 and Developer version 9 change to the timeouts set for the NIO HTTP server in Application Server version 11 and Developer version 11.

The following figure shows the points for which a timeout can be set.

Figure 12–7: Points for which a timeout can be set (for the in-process HTTP server)



The following table describes the differences for each point that needs to be migrated from Application Server version 9 and Developer version 9 to Application Server version 11 and Developer version 11.

Table 12–9: Difference in the timeout set for each point (migration from the in-process HTTP server configuration)

Point	Timeout type	
	Application Server version 9 and Developer version 9	Application Server version 11 and Developer version 11
5	A timeout for receiving a request from the client, which is set on the Web container. Setup target • `usrconf.properties` (user property file for J2EE servers) • Logical J2EE server (`j2ee-server`) Location of setup `webserver.connector.inprocess_http.receive_timeout`	A timeout for receiving data from the client, which is set on the NIO HTTP server. Setup target • `usrconf.properties` (user property file for J2EE servers) • Logical J2EE server (`j2ee-server`) Location of setup `webserver.connector.nio_http.receive_timeout`
13	A timeout for sending the response to the client, which is set on the Web container. Setup target • `usrconf.properties` (user property file for J2EE servers) • Logical J2EE server (`j2ee-server`) Location of setup `webserver.connector.inprocess_http.send_timeout`	A timeout for sending data to the client, which is set on the NIO HTTP server. Setup target • `usrconf.properties` (user property file for J2EE servers) • Logical J2EE server (`j2ee-server`) Location of setup `webserver.connector.nio_http.send_timeout`

12.4.2 Estimating the resources to be used

In Application Server version 11 and Developer version 11, the following estimation formula among the formulas for estimating the resources used by the J2EE application execution platform has changed:

- Formula for estimating the resources used by the J2EE server

(1) Estimating the resources used by the J2EE server

In Application Server version 11 or later and Developer version 11 or later, the in-process HTTP server was replaced with the NIO HTTP server functionality and new functionality that performs asynchronous processing were added. As a result, the number of threads for the J2EE server process and the number of file descriptors have changed.

For details, see *5.2.1 Estimating the resources used by J2EE server* in the *uCosminexus Application Server System Design Guide*.

12.5 Migration guide for system maintenance information

This section describes the changes from Application Server version 9 and Developer version 9 regarding the handling of maintenance information such as logs and trace based performance analysis.

12.5.1 Changes of the output destination log files

In Application Server version 11 and Developer version 11, the output destination log files for some maintenance information were changed or abolished.

Table 12–10: Changes in the output destinations of maintenance information

Output source	Category	Output destination	
		Application Server version 9 and Developer version 9	Application Server version 11 and Developer version 11
J2EE server	Access log of the in-process HTTP server or the NIO HTTP server	*ejb.server.log.directory*\http\cjhttp_access.inprocess_http[*n*].log	*ejb.server.log.directory*\cj_access_niohttp[*n*].log
	WebSocket access log	--	*ejb.server.log.directory*\cj_access_websocket[*n*].log
	Development investigation log	*ejb.server.log.directory*\cjdevelopment[*n*].log	The introduced base log output function outputs the equivalent content to the message log and the exception log.
Redirector	Message log of the redirector	*Application-Server-installation-directory*\CC\web\redirector\logs Alternatively, the value specified for JkLogFileDir	Abolished
	Maintenance trace log of the redirector	*Application-Server-installation-directory*\CC\web\redirector\logs Alternatively, the value specified for JkLogFileDir	Abolished

12.5.2 Changes in the access log

The NIO HTTP server functionality of Application Server version 11 and Developer version 11 has an access log output function equivalent to the access log of the conventional in-process HTTP server. The following table describes the changes in the properties for which access log settings are specified.

Table 12–11: Recommended values for the properties for the access log of the NIO HTTP server

Setting item	Property name in Application Server version 9 and Developer version 9	Property name in Application Server version 11 and Developer version 11	Recommended value in Application Server version 11 and Developer version 11
Maximum file size	`webserver.logger.access_log.inprocess_http.filenum`	`ejbserver.logger.channels.define.NIOHTTPAccessLogFile.filenum`	The same value as that in Application Server version 9 and Developer version 9
Maximum number of files	`webserver.logger.access_log.inprocess_http.filesize`	`ejbserver.logger.channels.define.NIOHTTPAccessLogFile.filesize`	The same value as that in Application Server version 9 and Developer version 9
Format	`webserver.logger.access_log.`*format-name*	`ejbserver.logger.access_log.nio_http.format`	Default value + Format argument specified individually

12.5.3 Changes in the trace collection points of the trace based performance analysis

In Application Server version 11 and Developer version 11, some collection points of the trace based performance analysis were changed or abolished. If the trace based performance analysis functionality was used, replace the event IDs according to the following table.

Table 12–12: Changes in the trace collection points of the trace based performance analysis

Output source	Trace collection point	Event ID		Level
		Application Server version 9 and Developer version 9	Application Server version 11 and Developer version 11	
Web container (Web server integration)	Immediately after the acquisition of a request or the completion of the request header analysis	0x8200	0x8236	A
	Immediately after the completion of request processing	0x8300	0x8336	A
Web container (In-process HTTP server)	When a request is acquired or immediately after the completion of request header analysis	0x8211	0x8236[#1]	A/B
	Immediately after the completion of request processing	0x8311	0x8336[#1]	A/B
	Immediately before the reading of data from the Web client starts	0x8212	0x8237	B
	Immediately after the completion of the reading of data from the Web client	0x8312	0x8337	B
	Immediately before the writing of data to the Web client starts	0x8213	0x8238	B

Output source	Trace collection point	Event ID		Level
		Application Server version 9 and Developer version 9	Application Server version 11 and Developer version 11	
	Immediately after the completion of the writing of data to the Web client	0x8313	0x8338	B
Redirector	All collection points	0x8000 to 0x8104	Abolished[#2]	A/B

#1

In Application Server version 9 and Developer version 9, for level B, more detailed information than level A was output. However, in Application Server version 11 and Developer version 11, the same content is output for level A and level B.

#2

The root application information is output to the access log and request log of HTTP Server. Use those logs for comparison as a substitute.

12.5.4 Changes in messages

In Application Server version 11 and Developer version 11, the message IDs and their content output to the message log were changed or abolished. The following table lists the message IDs of Application Server version 11 and Developer version 11 that are output under the equivalent conditions. If you monitored the relevant messages, replace the message IDs according to the following table.

Table 12–13: Changes in the message IDs

Output source	Message in Application Server version 9 and Developer version 9		Replacement ID in Application Server version 11 and Developer version 11	Log level
	ID	Content		
Web container	KDJE39236-I	Web server integration will start.	None	Error
	KDJE39237-I	The in-process HTTP server will start. The Web server integration functionality will be disabled.	KDJE39562-I	Error
	KDJE39238-W	The host name specified in the property `webserver.connector.inprocess_http.permitted.hosts` could not be resolved.	KDJE39563-W	Error
	KDJE39239-W	Access to the in-process HTTP server from a forbidden host was denied.	KDJE39564-W	Error
	KDJE39240-W	The host (`{0}`) specified in the property `webserver.connector.inprocess_http.bind_host` could not be resolved.	KDJE39565-W	Error
	KDJE39241-E	The in-process HTTP server could not be started on the specified port number (`{0}`).	KDJE39566-E	Error
	KDJE39242-W	The host (`{0}`) specified in the property `webserver.connector.inprocess_http.bind_host` could not be resolved.	KDJE39567-W	Error

Output source	Message in Application Server version 9 and Developer version 9		Replacement ID in Application Server version 11 and Developer version 11	Log level
	ID	Content		
	KDJE39243-E	The HTTP method of the sent request is not allowed.	None	Error
	KDJE39244-E	The request line of the sent request has exceeded the maximum size.	None	Error
	KDJE39245-E	The request header of the sent request has exceeded the maximum size.	KDJE39568-E	Error
	KDJE39246-E	The request body of the sent request has exceeded the maximum size.	KDJE39569-E	Error
	KDJE39247-E	The number of HTTP headers included in the sent request has exceeded the maximum number.	KDJE39570-E	Error
	KDJE39256-W	The file sent as the response body of the redirect functionality could not be read.	None	Error
	KDJE39257-E	An error occurred while the file sent as the response body of the redirect functionality was being read.	None	Error
	KDJE39258-W	An error occurred while the response body of the redirect functionality was being sent.	None	Information
	KDJE39259-W	The file sent as the response body of the error contents customization functionality was not sent to the client because it could not be read.	None	Error
	KDJE39260-E	An error occurred while the file sent as the response body of the error contents customization functionality was being read.	None	Error
	KDJE39261-W	An error occurred while the response body of the error contents customization functionality was being sent.	None	Information
	KDJE39262-E	The request processing will be denied because the number of connections with the Web client has exceeded the maximum value.	None	Error
	KDJE39263-W	The number of request processing threads is insufficient.	None	Error
	KDJE39265-E	A timeout occurred while the request header from the client was being read.	KDJE39576-E	Warning
	KDJE39267-W	The accessed request was denied. A request beginning with `/ejb/` or `/web/` cannot be used on the in-process HTTP server.	KDJE39571-W	Error
	KDJE39268-E	The file sent as the response body of the redirect functionality could not be read.	None	Error
	KDJE39269-E	The in-process HTTP server denied the accessed request because the body information of the HTTP request was too large.	None	Error
	KDJE39270-W	The error page could not be customized because the response was already committed.	None	Error
	KDJE39274-E	A failure occurred while a connection with the client was being established.	KDJE39572-E	Error
	KDJE39276-W	An attempt to establish a connection with the client succeeded although a failure occurred while a connection with the client was being established.	KDJE39573-W	Error
	KDJE39277-E	The in-process HTTP server denied the accessed request because an invalid request header was received.	KDJE39574-E	Error
	KDJE39514-W	Some request processing threads have not finished.	KDJE39575-W	Error

Output source	Message in Application Server version 9 and Developer version 9		Replacement ID in Application Server version 11 and Developer version 11	Log level
	ID	Content		
	KDJE39561-W	The function that narrows the search range of `ServletContainerInitializer` is enabled.	None	Error
Redirector	KDJE41000 to KDJE41041	All messages	None#	--

#

Use the error log of HTTP Server as a substitute.

12.6 Parameter replacement reference

This section describes how to calculate the values set for the parameters added in Application Server version 11 and Developer version 11 by using the values set in Application Server version 9 and Developer version 9.

12.6.1 User property definitions for J2EE servers

The following table describes the recommended property values that are specified in the user property file for J2EE servers, among the parameters added in Application Server version 11 and Developer version 11. For details about the properties, see the *uCosminexus Application Server Definition Reference Guide*.

Table 12–14: Migration destination properties of the user properties for J2EE servers

Property name in Application Server version 11 and Developer version 11	Recommended value in Application Server version 11 and Developer version 11	
	For Web server integration	For the in-process HTTP server
webserver.connector.nio_http.max_connections	See 12.4.1(1)(a).	See 12.4.1(1)(b).
webserver.connector.nio_http.max_threads		
webserver.connector.nio_http.min_threads		
webserver.connector.nio_http.receive_timeout	See 12.4.1(2)(a).	See 12.4.1(2)(b).
webserver.connector.nio_http.send_timeout		
webserver.connector.nio_http.backlog	The same value as webserver.connector.ajp13.backlog	The same value as webserver.connector.inprocess_http.backlog
webserver.connector.nio_http.bind_host	If the host name of the transfer destination URL of the reverse proxy is localhost, the recommended value is localhost. For other cases, the recommended value is an IP address or host name that can communicate with the Web server.	The same value as webserver.connector.inprocess_http.bind_host
webserver.connector.nio_http.hostname_lookups	false (default value)	The same value as webserver.connector.inprocess_http.hostname_lookups
webserver.connector.nio_http.limit.max_headers	100 (default value)	The same value as webserver.connector.inprocess_http.limit.max_headers
webserver.connector.nio_http.limit.max_request_body	-1 (default value)	The same value as webserver.connector.inprocess_http.limit.max_request_body
webserver.connector.nio_http.limit.max_request_header	16384 (default value)	The same value as webserver.connector.inprocess_http.limit.max_request_header

Property name in Application Server version 11 and Developer version 11	Recommended value in Application Server version 11 and Developer version 11	
	For Web server integration	For the in-process HTTP server
webserver.connector.nio_http.keep_alive.max_requests	0 (default value)	The same value as webserver.connector.inprocess_http.persistent_connection.max_requests
webserver.connector.nio_http.keep_alive.timeout	0 (default value)	The same value as webserver.connector.inprocess_http.persistent_connection.timeout
webserver.connector.nio_http.port	Any port number that can communicate with the Web server	The same value as webserver.connector.inprocess_http.port
webserver.connector.nio_http.response.header.server	CosminexusComponentContainer (default value)	The same value as webserver.connector.inprocess_http.response.header.server
webserver.connector.nio_http.idle_thread_timeout	60 (default value)	60 (default value)
webserver.connector.nio_http.permitted.hosts	When the host name of the transfer destination URL of the reverse proxy is localhost localhost For other cases The IP address or host name of the Web server	The same value as webserver.connector.inprocess_http.permitted.hosts
webserver.connector.nio_http.max_servlet_execute_threads	The same value as webserver.connector.ajp13.max_threads	The same value as webserver.connector.inprocess_http.max_execute_threads
ejbserver.logger.channels.define.NIOHTTPAccessLogFile.filenum	16 (default value)	The same value as webserver.logger.access_log.inprocess_http.filenum
ejbserver.logger.channels.define.NIOHTTPAccessLogFile.filesize	4194304 (default value)	The same value as webserver.logger.access_log.inprocess_http.filesize
ejbserver.logger.access_log.nio_http.format	%h %{X-Forwarded-For}i %l %u %d %rootap "%r" %s %b %D %S (default value)	When the value of webserver.logger.access_log.inprocess_http.usage_format is common or combined Default value For other cases In addition to the default value, the format argument individually specified for webserver.logger.access_log.*format-name*. *format-name* is the character string specified for webserver.logger.access_log.inprocess_http.usage_format.
webserver.context.stop_asyncwait_timeout	30 (default value)	

Property name in Application Server version 11 and Developer version 11	Recommended value in Application Server version 11 and Developer version 11	
	For Web server integration	For the in-process HTTP server
webserver.servlet_api.unsupported.throwUnsupportedOperationException	See 12.3.2(1).	

12.6.2 Definitions of the redirector

If the Web server integration configuration using the redirector was used in Application Server version 9 and Developer version 9, the Web server integration functionality using the redirector is replaced with the reverse proxy functionality of HTTP Server. The following table describes the replacements from the parameters specified for the conventional redirector to the directives of the reverse proxy functionality.

Table 12–15: Migration destinations of the definition files of the redirector

Specification destination in Application Server version 9 and Developer version 9		Corresponding specification destination in Application Server version 11 and Developer version 11
Specification destination file	Parameter name	
mod_jk.conf (redirector action definition file)	JkConnectTimeout	See 12.4.1(2)(a).
	JkSendTimeout	
	JkPrfId	HWSPrfId directive
	JkRequestRetryCount	None (Retries are never performed.)
	Other	None
worker.properties (worker definition file)	worker.*worker-name*.host	Host name of the URL specified for the second argument of the ProxyPass directive
	worker.*worker-name*.port	Port number of the URL specified for the second argument of the ProxyPass directive
	worker.*worker-name*.receive_timeout	See 12.4.1(2)(a).
	Other	None

12.7 Abolished parameter reference

This section describes the parameters abolished in Application Server version 11 and Developer version 11.

12.7.1 Files used by the J2EE server

(1) usrconf.properties (user property file for J2EE servers)

The following table lists the keys that cannot be specified in Application Server version 11 and Developer version 11.

Table 12–16: Abolished parameters in the user property file for J2EE servers

Abolished key name of Application Server version 9 and Developer version 9	Remarks
Keys starting with `webserver.connector.inprocess_http`	The in-process HTTP server functionality cannot be used.
Keys starting with `webserver.logger.access_log`	
`webserver.logger.communication_trace.inprocess_http.filenum`	
`webserver.logger.thread_trace.inprocess_http.filenum`	
`webserver.container.thread_control.enabled`	The specified value is ignored and the value is always assumed to be true.
Keys starting with `webserver.connector.ajp12`	The Web server integration functionality using the redirector cannot be used.
Keys starting with `webserver.connector.ajp13`	
`ejbserver.server.eheap.ajp13.enabled`	
Keys starting with `webserver.eadssfo.`	The EADs session failover functionality cannot be used.
`webserver.jsp.el2_2.enabled`	The specified value is ignored and the value is always assumed to be EL 3.0.
Keys starting with `cosminexus.jpa.`	Parameters unique to the CJPA provider cannot be used.
Keys starting with `ejbserver.jpa`	The JPA container functionality cannot be used.
`ejbserver.server.j2ee.feature`	Operation modes compatible with old versions such as 1.3basic cannot be specified.

12.7.2 Files used by Web server integration

The following files cannot be used:

- Redirector action definition file for HTTP Server (`mod_jk.conf`)
- Worker definition file (`workers.properties`)
- Redirector action definition file for Microsoft IIS (`isapi_redirect.conf`)
- Mapping definition file for Microsoft IIS (`usrworkermap.properties`)

12.7.3 Files used by JPA

(1) persistence.xml

The following properties cannot be specified as properties specified for the `<property>` tag in `persistence.xml`.

- Properties starting with `cosminexus.jpa`

12.7.4 Parameters specified for the Smart Composer functionality

(1) Parameters specified for the logical J2EE server

The parameters described in *Table 12-16 Abolished parameters in the user property file for J2EE servers* cannot be specified as parameters for the logical J2EE server.

(2) Parameters specified for the logical Web server

The following parameters cannot be specified as parameters for the logical Web server:

- Parameters that set up the redirector action definition for HTTP Server
- Parameters that set up the worker definition

Appendixes

A. List of Snapshot Logs to Be Collected

This section describes the execution environment directories of the configuration software that are to be collected when you want to collect snapshot logs at the same time. Note that if you expand the collected ZIP file, the directories are extracted in the same directory configuration as that before collection.

When you collect all snapshot logs at the same time, the log files and definition files of the configuration software and libraries of a system built with the application server are collected. You can change the directory paths to be collected by editing the following files. For details about the settings for collecting the snapshot log, see the subsection *3.3.3 Settings for collecting snapshot logs (Systems for executing J2EE applications)* or the subsection *3.3.4 Settings for collecting snapshot log (Systems for executing batch applications)*.

The following table describes the correspondence with the data required for troubleshooting and the snapshot log data type.

Table A–1: Correspondence with the data necessary for troubleshooting and snapshot log data type

Data necessary for troubleshooting		Data type of snapshot log
Application Server log	Message log	Message log
	User log	Other logs
	Exception log	Other logs
	Maintenance log	Maintenance log
EJB client application system log	Message log	Message log
	User log	Other logs
	Exception log	Other logs
	Maintenance log	Maintenance log
Trace based performance analysis		Performance, error analysis trace
Thread dumps of JavaVM		Dump
GC logs of JavaVM		Other logs
Memory dump		Dump
JavaVM log file		Other logs
Error report files		Dump
OS state or log		Other logs
Statistical information of OS		Other logs
Definition information		Definition information
Operation directory		User data
Resource setting		User data
Web server log		Message log
		Other logs
		Access log
		Internal interface
JavaVM stack trace		Other logs

Data necessary for troubleshooting	Data type of snapshot log
Event log of the Explicit Memory Management functionality	Other logs

Important note

The snapshot log is collected for directories created by default during installation of the configuration software in the default state. If the output destination and working directory of the log are changed, customize the collection destination.

A.1 Overview of the list of the snapshot log to be collected

This appendix describes the storage location of the snapshot log data and the collection method, directories, and file name described in the list.

(1) Data storage location

The snapshot log is collected in three files, namely `snapshotlog.conf`, `snapshotlog.2.conf`, and `snapshotlog.param.conf`.

- `snapshotlog.conf`

  This is a file that describes the logs to be collected as primary delivery data. The storage location of this file is as follows:

  - In Windows

    Cosminexus-installation-directory`\manager\config\snapshotlog.conf`
  - In UNIX

    `/opt/Cosminexus/manager/config/snapshotlog.conf`
- `snapshotlog.2.conf`

  This is a file that describes the logs to be collected as secondary delivery data. The storage location of this file is as follows:

  - In Windows

    Cosminexus-installation-directory`\manager\config\snapshotlog.2.conf`
  - In UNIX

    `/opt/Cosminexus/manager/config/snapshotlog.2.conf`
- `snapshotlog.param.conf`

  This file describes the logs to be collected for the definition sending data. The storage location of the file is as follows:

  - In Windows

    Application-Server-installation-directory`\manager\config\snapshotlog.param.conf`
  - In UNIX

    `/opt/Cosminexus/manager/config/snapshotlog.param.conf`

(2) Method of collecting snapshot log

The following types are included in the methods of collecting the snapshot log. The following table describes the method of collecting the snapshot log.

Table A–2: Methods of collecting snapshot log

No.	How to collect	Category
1	Collecting the log from the Snapshot Log window of the **Start/ stop logical server** anchor of the management portal	Collection method A
2	Executing management command (`mngsvrutil collect snapshot` command)	Collection method A
3	Auto collecting after detecting the error of the logical server	Collection method A
4	Execute `snapshotlog` command using the standard `snapshotlog.conf`, `snapshotlog.2.conf`, or `snapshotlog.param.conf`	Collection method B

(3) Availability of snapshot log collection and changes in settings related to collection

In the tables from Table A-5 to Table A-45, availability of collection and availability of changes in settings for the `mngsvrutil` command or `snapshotlog` command are categorized with labels. The following table describes each label.

Table A–3: Definition of label used in Table A-5 to Table A-45

Label used in table	Default collection	Changes in log output destination and storage destination of definition file	Collection when log output destination and storage destination of definition file is changed
A	Collected.	Possible.	Possible.
Y	Collected.	Not possible.	Not possible.
C	Collected.	Possible.	Possible but with some restrictions[#1].
D	Not collected.	Not possible.	Not possible[#2].

#1

You must edit the definition file (`snapshotlog.conf`, `snapshotlog.2.conf`, or `snapshotlog.param.conf`) for collecting the snapshot log and set the file to the directory targeted for collection.

#2

You cannot change the log output destination or storage destination of a definition file. You must edit the definition file (`snapshotlog.conf`, `snapshotlog.2.conf`, or `snapshotlog.param.conf`) for collecting the snapshot log and set the file to the directory targeted for collection.

Also, the *Data* column from Table A-5 to Table A-45 indicates the data collected when the `mngsvrutil` command or `snapshotlog` command is executed.

If the data is collected using another method, or if `mngsvrutil collect snapshot 2` is executed, all the data is collected regardless of the contents of the *Data* column in the table.

The notations and symbols in the Data column are as follows:

- Primary: Primarily sent data
  This data is collected when the `snapshotlog` command specifying `snapshotconf.conf`, or `mngsvrutil collect snapshot 1` is executed.
- Secondary: Secondarily sent data
  This data is collected when the `snapshotlog` command specifying `snapshotconf.2.conf`, or `mngsvrutil collect snapshot 2` is executed.

- Definition: Definition sending data
  This data is collected when the `snapshotlog` command is executed specifying `snapshotlog.param.conf`.
- Symbols in the table
  Y: Collected
  --: Not collected

(4) Rules for the coding to be collected

The common method to code the directories and files to be collected is as follows:

- The asterisk (`*`) in the path of the directory or file to be collected indicates any character string of 0 or more characters.
- `\*\*\*` in Windows and `/*/*/*` in UNIX indicates the hierarchy in which the files are collected. For example, in the case of `log\*\*` in Windows or `log/*/*` in UNIX, the files that are two levels below the log directory are to be collected.

The following table describes the significance of the files and directories to be collected.

Table A–4: Significance of files and directories to be collected

Directory or file name	Contents	Default value	Changed Value
Installation-directory-of-Cosminexus	Installation directory path name of the Application Server	• In Windows Decided when installing the Application Server. • In UNIX `/opt/Cosminexus`	`${cosminexus.home}`
CTM-identifier	`-CTMID` option name of `ctmstart` command	--	.+
CTM-spool-directory-(ctmspool)	Directory path name specified for environment variable `CTMSPOOL`	*Installation-directory-of-Cosminexus*`/CTM/spool`	`&{ctmspool}`
DABroker-operation-directory-(dabroker)	Directory path name set after DABroker installation by `dabsetup` command	`/opt/DABroker`	`<DABroker_Oparation>`
EJB-client-definition-file-storage-directory	Directory path name specified in environment variable `CJCLUSRCONFDIR` or directory path name, which executed the `cjclstartap` command	--	`<ejb.client.definition.file.dir>`
EJB-client-log-subdirectory-(ejb.client.ejb.log)	Sub directory name specified in `ejb.client.ejb.log` key of `usrconf.cfg` for Java application	`system`	.+
EJB-client-log-subdirectory1-(ejbserver.client.ejb.log)	Directory path name specified in `ejbserver.client.ejb.log` key of `usrconf.properties` for Java application	--	.+
EJB-client-log-subdirectory2-(ejb.client.log.appid)	Subdirectory name specified in `ejb.client.log.appid`	`ejbcl`	.+

Directory or file name	Contents	Default value	Changed Value
	key of `usrconf.cfg` for Java application		
EJB-client-log-subdirectory2-(ejbserver.client.log.appid)	Directory path name specified in `ejbserver.client.log.appid` key of `usrconf.properties` for Java application	--	`.+`
EJB-client-log-output-directory-(ejb.client.log.directory)	Directory path name specified in `ejb.client.log.directory` key of `usrconf.cfg` for Java application	--	`<ejb.client.log.directory>`
EJB-client-log-output-directory-(ejbserver.client.log.directory)	Directory path name specified in `ejbserver.client.log.directory` key of `usrconf.properties` for Java application	*Installation-directory-of-Cosminexus*`/CC/client/logs`	`<ejbserver.client.log.directory>`
HCSC-server-property-(methodtrace-filepath)	Directory path specified in the `methodtrace-filepath` property with the HCSC server definition command `cscsvconfig`	*Installation-directory-of-Cosminexus*`/CC/server/public/ejb/`*server-name*`/logs/csc/maintenance`	`<methodtrace-filepath>`
HCSC-server-property-(requesttrace-filepath)	Directory path specified in the `requesttrace-filepath` property with the HCSC server definition command `cscsvconfig`	*Installation-directory-of-Cosminexus*`/CC/server/public/ejb/`*server-name*`/logs/csc/maintenance`	`<requesttrace-filepath>`
HCSC-repository-root-(cscmng.repository.root)	Repository root specified in the `cscmng.repository.root` key of HCSC-Manager definition `cscmng.properties`	*Installation-directory-of-Cosminexus*`/CSC/repository`	`<cscmng.repository.root>`
HCSC-log-output-directory-(cscmng.log.dir)	Log directory path specified in the `cscmng.log.dir` key of HCSC-Manager definition `cscmng.properties`	*Installation-directory-of-Cosminexus*`/CSC/log/manager`	`<cscmng.log.dir>`
HWS-access-log-directory-(HttpsdCustomLogFileDir)	Directory path name of the file name specified in `CustomLog` directory of `httpsd.conf`	*Installation-directory-of-HWS*`/servers/HWS_`*server-name*`/logs`	`&{hws.logfile.dir}`
Installation-directory-of-HWS	Cosminexus HTTP Server installation directory path	In Windows *Installation-directory-of-Cosminexus*`/httpsd` In UNIX `opt/hitachi/httpsd`	`${hws.home}`
HWS-error-log-directory-(HttpsdErrorLogFileDir)	Directory path name of the file name specified in `ErrorLogr` directory of `httpsd.conf`	*Installation-directory-of-HWS*`/servers/HWS_`*server-name*`/logs`	`&{hws.logfile.dir}`
HWS-cache-server-run-directory-(SSLCacheServerRunDir)	Directory path name specified in `SSLCacheServerRunDir` directory of `httpsd.conf`	*HWS-server-root-directory-(ServerRoot)*	`<SSLCacheServerRunDir>`

Directory or file name	Contents	Default value	Changed Value
HWS-core-dump-output-directory-(CoreDumpDirectory)	Directory path name specified in `CoreDumpDirectory` directory of `httpsd.conf`	*Installation-directory-of-HWS*`/servers/HWS_`*server-name*	`&{core.dump.directory}`
HWS-server-root-directory-(ServerRoot)	Directory path name specified in `ServerRoot` directory of `httpsd.conf`	*Installation-directory-of-HWS*	`<ServerRoot>`
HWS-trace-directory-(HttpsdTraceLogFileDir)	Directory path name of the file name specified in `HWSTraceLogFile` directory of `httpsd.conf`	*Installation-directory-of-HWS*`/servers/HWS_`*server-name*`/logs`	`&{hws.logfile.dir}`
HWS-processID-file-(PidFile)	File name specified in `PidFile` directory of `httpsd.conf`	*Installation-directory-of-HWS*`/servers/HWS_`*server-name*`/logs/httpd.pid`	`<PidFile>`
Redirector-log-output-directory-for-HWS-(JkLogFileDir)	Directory path specified in the `JkLogFileDir` key of `mod_jk.conf`	*Installation-directory-of-Cosminexus*`/CC/web/redirector/servers/`*server-name*`/logs`	`&{jklogfiledir}`
Redirector-trace-log-output-directory-for-HWS-(JkTraceLogFileDir)	Directory path specified in the `JkTraceLogFileDir` key of `mod_jk.conf`	*Installation-directory-of-Cosminexus*`/CC/web/redirector/servers/`*server-name*`/logs`	`&{jktracelogfiledir}`
HWS-request-log-directory-(HttpsdRequestLogFileDir)	Directory path name of the file name specified in `HWSRequestLog` directory of `httpsd.conf`	*Installation-directory-of-HWS*`/servers/HWS_`*server-name*`/logs`	`&{hws.logfile.dir}`
IIS-access-log-directory-(IIS_log_dir)	Access log output directory path name of Microsoft IIS	*System-root-directory-(systemroot)*`/system32/LogFiles/W3SVC1`	`<IIS_log_dir>`
Redirector-trace-log-output-directory-for-IIS-(trace_log_file_dir)	Directory path name specified in `trace_log_file_dir` key of `isapi_redirect.conf`	*Installation-directory-of-Cosminexus*`/CC/web/redirector/servers/`*server-name*`/logs`	`&{jktracelogfiledir}`
Redirector-log-output-directory-for-IIS-(log_file_dir)	Directory path name specified in `log_file_dir` key of `isapi_redirect.conf`	*Installation-directory-of-Cosminexus*`/CC/web/redirector/servers/`*server-name*`/logs`	`&{jklogfiledir}`
J2EE-server-working-directory-(ejb.public.directory)	Directory path name specified in the `ejb.public.directory` key of `usrconf.cfg` for a J2EE server	*Installation-directory-of-Cosminexus*`/CC/server/public`	`&{ejb.public.directory}`
J2EE-server-log-output-directory-(ejb.server.log.directory)	Directory path specified in the `ejb.server.log.directory` key of `usrconf.cfg` for a J2EE server	*Installation-directory-of-Cosminexus*`/CC/server/public/ejb/`*server-name*`/logs`	`&{ejb.server.log.directory}`
JAXR-trace-file-name-(com.cosminexus.xml.registry.trace.file_path)	File path name specified in `com.cosminexus.xml.registry.trace.file_path` key of system property	`${cosminexus.home}/c4web/logs/JAXRAPITrace`	`<com.cosminexus.xml.registry.trace.file_path>`

Directory or file name	Contents	Default value	Changed Value
Temporary-directory-for-JSP-(webserver.work.directory)	Directory path specified in the `webserver.work.directory` key of `usrconf.properties` of a J2EE server	*Installation-directory-of-Cosminexus*`/CC/server/repository/`*server-name*`/web`	`&{webserver.work.directory}`
Message-ID-list-file-for-Management-event-publication-(manager.mevent.message_id.list)	File path name specified in `manager.mevent.message_id.list` key of `mevent.`*server-name*`.properties`	--	`<manager.mevent.message_id.list>`
Manager-log-output-directory-(com.cosminexus.manager.log.dir)	Directory path specified in the `com.cosminexus.manager.log.dir` key of `manager.cfg`	*Installation-directory-of-Cosminexus*`/manager/log`	`${com.cosminexus.manager.log.dir}`
PRF-identifier	`-PRFID` option name of `cprfstart` command	--	`.+`
PRF-spool-directory-(prfspool)	Directory path name specified for environment variable `PRFSPOOL`	*Installation-directory-of-Cosminexus*`/PRF/spool`	`&{prfspool}`
RM-installation-directory	Directory path name specified for environment variable `HRMDIR`	*Installation-directory-of-Cosminexus*`/RM`	`<HRMDIR>`
TP1-onnector-log-output-directory-(jp.co.hitachi_system.tp1connector.logdestination)	Directory path name specified in `jp.co.hitachi_system.tp1connector.logdestination` key of system property	*user-home-directory-(user.home)*	`<jp.co.hitachi_system.tp1connector.logdestination>`
UNIX-syslog-(syslog)	`syslog` file path name in UNIX	• In AIX `/var/adm/ras/syslog*` • In Linux `/var/log/messages*`	`syslog`
VBROKER_ADM-directory-(vbroker_adm)	Directory path specified in the environment variable `VBROKER_ADM`	*Installation-directory-of-Cosminexus*`/TPB/adm`	`<VBROKER_ADM>`
Web-container-server-log-output-directory-(web.server.log.directory)	Directory path specified in the `web.server.log.directory` key of `usrconf.cfg` for a J2EE server	*Installation-directory-of-Cosminexus*`/CC/web/containers/`*server-name*`/logs`	`<web.server.log.directory>`
Windows-event-log-(EventLog)	File where event log of Windows is collected using `wmic` command	--	`<EventLog>`
Windows-Crash-dump-output-directory-(CrashDumpDir)	Windows crash dump output destination directory path name	*Users-Application-Data-directory-(LOCALAPPDATA)*`/CrashDumps`	`<CrashDumpDir>`
XML-Security-Core-trace-output-directory-(com.cosminexus.xml.security.logging.trace_dir)	Directory path specified in the `com.cosminexus.xml.security.logging.trace_dir` key of `usrconf.cfg` for a J2EE server	*Installation-directory-of-Cosminexus*`/CC/server/public/ejb/`*server-name*	`<com.cosminexus.xml.security.logging.trace_dir>`

Directory or file name	Contents	Default value	Changed Value
In-process-HTTP-server-access-log-file-(webserver.logger.access_log.inprocess_http.filename)	Directory path specified in the `webserver.logger.access_log.inprocess_http.filename` key of `usrconf.properties` for a J2EE server	*Installation-directory-of-Cosminexus*`/CC/server/public/ejb/`*server-name*`/logs/http/cjhttp_access.inprocess_http`	`&{webserver.logger.access_log.inprocess_http.filename}.`
statistics-file-output-destination-directory-(ejbserver.management.stats_file.dir)	Directory path specified in the `ejbserver.management.stats_file.dir` key of `usrconf.properties` for a J2EE server	*Installation-directory-of-Cosminexus*`/CC/server/public/ejb/`*server-name*`/stats`	`&{ejbserver.management.stats_file.dir}`
Audit-log-output-directory-(auditlog.directory)	Directory path name specified in `auditlog.directory` key of audit log definition file	*Installation-directory-of-Cosminexus*`/auditlog`	`<auditlog.directory>`
Audit-log-message-output-directory-(auditlog.raslog.message.directory)	Directory path name specified in `auditlog.raslog.message.directory` key of the Audit log definition file	*Installation-directory-of-Cosminexus*`/auditlog`	`<auditlog.raslog.message.directory>`
Audit-log-exception-output-director-(auditlog.raslog.exception.directory)	Directory path name specified in `auditlog.raslog.exception.directory` key of Audit log definition file	*Installation-directory-of-Cosminexus*`/auditlog`	`<auditlog.raslog.exception.directory>`
Context-root	Root name of the J2EE web application executed on the server	--	`.+`
Server-management-command-log-output-directory-(admin_ejb.server.log.directory)	Directory path name specified in the `-Dejbserver.log.directory` option of `USRCONF_JVM_ARGS` key of the option definition file for server management commands	*Installation-directory-of-Cosminexus*`/CC/admin/logs`	`<admin_ejb.server.log.directory>`
Server-name	Logical server name or real server name set by Management Server	--	`.+`
System-drive-(SystemDrive)	Directory path name specified for environment variable `SystemDrive`	--	`%{SystemDrive}`
System-root-directory-(systemroot)	Directory path name specified for environment variable `SystemRoot`	--	`%{SystemRoot}`
Status-file-directory-(ejbserver.distributedtx.ots.status.directory1)	Directory path specified in the `ejbserver.distributedtx.ots.status.directory1` key of `usrconf.properties` of a J2EE server	*Installation-directory-of-Cosminexus*`/CC/server/public/ejb/`*server-name*`/otsstatus`	`&{ejbserver.distributedtx.ots.status.directory1}`
Trace-file-name-of-integrated-user-management-(com.cosminexus.admin.auth.trace.prefix)	File path name specified in `com.cosminexus.admin.auth.trace.prefix` option of `ua.conf`	--	`<com.cosminexus.admin.auth.trace.prefix>`

Directory or file name	Contents	Default value	Changed Value
Batch-application-definition-file-storage-directory	Directory path name specified in environment variable CJBATCHUSRCONFDIR or directory path name, which executed the cjexecjob, cjkilljob, or cjlistjob commands	--	<batch.application.definition.file.dir>
Batch-application-log-output-directory-(batch.log.directory)	Directory path name specified in batch.log.directory of usrconf.cfg for batch applications	*Installation-directory-of-Cosminexus*/CC/batch/logs	<batch.log.directory>
User-Application-Data-directory-(LOCALAPPDATA)	Directory path name specified for environment variable LOCALAPPDATA	--	%{LOCALAPPDATA}
User-home-directory-(user.home)	Home directory path name of the executing user • In collection method A: Home directory of operation management agent start user • In collection method B: Home directory of the user executing snapshotlog command	--	${user.home}and <user.home>
Spare-status-file-directory-(ejbserver.distributedtx.ots.status.directory2)	Directory path specified in the ejbserver.distributedtx.ots.status.directory2 key of usrconf.properties of a J2EE server	--	&{ejbserver.distributedtx.ots.status.directory2}
Virtual-server-manager-log-directory-of-08-50-mode-(vmx.log.dir)	Directory path name specified in the vmx.log.dir key of vmx.properties	*Installation-directory-of-Cosminexus*/manager/vmx/log	<vmx.log.dir>
Processing-data-storage-directory-of-virtual-server-manager-of-08-50-mode-(vmx.spool.dir)	Directory path name specified in the vmx.spool.dir key of vmx.properties	*Installation-directory-of-Cosminexus*/manager/vmx/spool	<vmx.spool.dir>
Processing-data-storage-directory-of--virtual-server-manager-(vmi.spool.dir)	Directory path name specified in the vmi.spool.dir key of vmx.properties	*Installation-directory-of-Cosminexus*/manager/vmi/spool	<vmi.spool.dir>
server-communication-agent-log-output-directory-(sinaviagent.log.dir)	Directory path name specified in the sinaviagent.log.dir key of sinaviagent.cfg	*Installation-directory-of-Cosminexus*/sinagent/log	<sinaviagent.log.dir>
server-communication-agent-processing-data-storage-directory-(sinaviagent.spool.dir)	Directory path name specified in the sinaviagent.spool.dir key of sinaviagent.cfg	*Installation-directory-of-Cosminexus*/sinagent/spool	<sinaviagent.spool.dir>
FTP-adapter-installation-directory-(ftpadp.home)	Installation directory path name of FTP Adapter	--	<ftpadp.home>
FTP-adapter-command-message-log-output-destination-directory-	Directory path name of operation command message log output destination of FTP Adapter	*FTP-adapter-installation-directory-(ftpadp.home)*/log	<ftpadp.command.messagelog.filepath>

Directory or file name	Contents	Default value	Changed Value
(ftpadp.command.messagelog.filepath)			
HCSC-mail-adapter-operation-command-message-log-output-destination-directory-(mailadp.command.messagelog.filepath)	Directory path name of operation command execution log of mail adapter incorporated in Cosminexus Service Coordinator	*Installation-directory-of-Cosminexus*/CSC/log/adapter/command	<mailadp.command.messagelog.filepath>
HCSC-mail-adapter-maintenance-log-output-destination-directory-(mailadp.methodtrace.filepath)	Directory path name specified in the mailadp.methodtrace.filepath property in the execution environment property file of mail adapter	*Installation-directory-of-Cosminexus*/CC/server/public/ejb/*server-name*/logs/CSCADP/MAILADP/maintenance/	<mailadp.methodtrace.filepath>
CTM-regulator-setup-file-(ctm.RegOption)	Path of the CTM regulator configuration file	--	&{ctm.RegOption}
OTM-gateway-setup-file-(ctm.TSCGwOption)	Path of the OTM gateway configuration file	--	&{ctm.TSCGwOption}
Directory-to-output-temporary-PRF-trace-file-(adminagent.prftrace_dir)	Path of the directory to output the temporary PRF trace file	*Application-Server-installation-directory*/manager/tmp	${adminagent.prftrace_dir}
Name-of-Explicit-Memory-Management-functionality-exclusion-setup-file-(jvm.exmemexcludeclass.File)	File path specified in the JavaVM extension option -XX:ExplicitMemoryExcludeClassListFile	*Application-Server-installation-directory*/jdk/usrconf/exmemexcludeclass.cfg	<jvm.exmemexcludeclass.File>
Name-of-Explicit-Memory-Management-functionality-non-exclusion-setup-file-(jvm.exmemnotexcludeclass.File)	File path specified in the JavaVM extension option -XX:ExplicitMemoryNotExcludeClassListFile	*Application-Server-installation-directory*/jdk/usrconf/exmemnotexcludeclass.cfg	<jvm.exmemnotexcludeclass.File>
HWS-WebSocket-log-directory-(HttpsdWebSocketLogFileDir)	Directory path of the file name specified for the HWSWebSocketLog directive in httpsd.conf	*HWS-installation-directory*/servers/HWS_*server-name*/logs	&{hws.logfile.dir}

Legend:

--: No default value

A.2 Cosminexus Component Container

The following table describes the logs to be collected in relation to a Cosminexus Component Container.

Table A–5: Collection method related to Cosminexus Component Container (In Windows)

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
J2EE server	Message log	Operation log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjmessage*.log	Y	--	--	A	C
		Log operation log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjlogger.log	Y	--	--	A	C
		Resource adapter operation log deployed and used as J2EE resource adapter	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/connectors/*.log	Y	--	--	A	C
		Resource adapter operation log used by including in J2EE application (normal mode)	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/connectors/*J2EE-application-name*/*.log	Y	--	--	A	C
		Resource adapter operation log used by including in the J2EE application (test mode)	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/connectors/TEST#*J2EE-application-name*/*.log	Y	--	--	A	C
		Web servlet log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/web_servlet*.log	Y	--	--	A	C
	Other logs	User output log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/user_out*.log	Y	--	--	A	C
		User error log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/user_err*.log	Y	--	--	A	C
		JavaVM maintenance information and GC log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/javalog*.log	Y	--	--	A	C
		JavaVM Explicit Memory Management functionality event log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/ehjavalog*.log	Y	--	--	A	C
		Check development log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjdevelopment*.log	Y	--	--	A	C
		Exception information when an error occurs	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjexception*.log	Y	--	--	A	C

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		JavaVM event log	*J2EE-server-working-directory-(ejb.public.directory)*/ejb/*server-name*/hs_err*	Y	--	--	A	A
			J2EE-server-working-directory-(ejb.public.directory)/ejb/*server-name*/replay_pid*.log	Y	--	--	A	A
		User log of J2EE applications	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/user/*	Y	--	--	A	C
	Maintenance log	Maintenance information	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CC/maintenance/cjmaintenance*.log	Y	--	--	A	C
		Console message	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CC/maintenance/cjconsole*.log	Y	--	--	A	C
		EJB container maintenance information	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CC/maintenance/cjejbcontainer*.log	Y	--	--	A	C
		Web container maintenance information	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CC/maintenance/cjwebcontainer*.log	Y	--	--	A	C
		Start process standard output information	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CC/maintenance/cjstdout*.log	Y	--	--	A	C
		Start process standard error information	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CC/maintenance/cjstderr*.log	Y	--	--	A	C
		Termination process information	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CC/maintenance/cj_shutdown*.log	Y	--	--	A	C
	Dump	Thread dump	*J2EE-server-working-directory-(ejb.public.directory)*/ejb/*server-name*/javacore*	Y	--	--	A	A
	Statistical information	Operation information file	*statistics-file-output-destination-directory-(ejbserver.management.stats_file.dir)*/*	--	Y	--	A	C
		Memory monitoring log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/watch/cjmemorywatch*.log	Y	--	--	A	C
		Thread monitoring log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/watch/cjthreadwatch*.log	Y	--	--	A	C

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		Thread dump monitoring log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/watch/cjthreaddumpwatch*.log	Y	--	--	A	C
		HTTP request pending queue monitoring log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/watch/cjrequestqueuewatch*.log	Y	--	--	A	C
		HTTP session count monitoring log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/watch/cjhttpsessionwatch*.log	Y	--	--	A	C
		Connection pool monitoring log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/watch/cjconnectionpoolwatch*.log	Y	--	--	A	C
	Others	Log showing J2EE server start, stop or abnormal termination	*Windows-event-log-(EventLog)*	Y	--	--	D	D
	Definition information	Option definition file for J2EE servers	*Installation-directory-of-Cosminexus*/CC/server/usrconf/ejb/*server-name*/usrconf.cfg	Y	--	Y	Y	Y
		User property file for the J2EE server	*Installation-directory-of-Cosminexus*/CC/server/usrconf/ejb/*server-name*/usrconf.properties	Y	--	Y	Y	Y
		Security policy file for J2EE server	*Installation-directory-of-Cosminexus*/CC/server/usrconf/ejb/*server-name*/server.policy	Y	--	Y	Y	Y
		Backup of various definition files created with MNG	*Installation-directory-of-Cosminexus*/CC/server/usrconf/ejb/*server-name*/*	--	Y	--	Y	Y
		Protected area list file	*Installation-directory-of-Cosminexus*/CC/server/usrconf/criticalList.cfg	Y	--	Y	Y	Y
		Resource setting information	*J2EE-server-work-directory-(ejb.public.directory)*/ejb/*server-name*/import/**	--	Y	--	D	D
			J2EE-server-work-directory-(ejb.public.directory)/ejb/*server-name*/rars/**	--	Y	--	D	D
		Maintenance information	*Installation-directory-of-Cosminexus*/CC/server/version/**	Y	--	Y	Y	Y
	User data	Contents of EJB server working directory	*J2EE-server-working-directory-(ejb.public.directory)*/ejb/*server-name*/**	--	Y	--	D	D

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		Contents of Web container working directory	*J2EE-server-working-directory-(ejb.public.directory)*/web/*server-name*/**	--	Y	--	D	D
		Contents of temporary directory for JSP	*Temporary-directory-for-JSP-(webserver.work.directory)*/**	--	Y	--	D	D
Server management commands	Message log	Operation log	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*/cjmessage*.log	Y	--	--	C	C
		Log operation log	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*/cjlogger.log	Y	--	--	C	C
		Operation log in compatible mode	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*/*message*.log	Y	--	--	C	C
	Other logs	Exception information when an error occurs	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*/cjexception*.log	Y	--	--	C	C
		Exception information when an error occurs in compatible mode	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*/*exception*.log	Y	--	--	C	C
	Maintenance log	Maintenance information	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*/CC/maintenance/cjmaintenance*.log	Y	--	--	C	C
		Console message	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*/CC/maintenance/cjconsole*.log	Y	--	--	C	C
		Server management command maintenance information	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*/CC/maintenance/cjserveradmin*.log	Y	--	--	C	C
		Other logs	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*/*/*	Y	--	--	C	C
			server-management-command-log-output-directory-(admin_ejb.server.log.directory)/*/*/*	Y	--	--	C	C
			server-management-command-log-output-directory-(admin_ejb.server.log.directory)/*/*/*/*	--	Y	--	C	C

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
	Definition information	Definition file for server management command	*Installation-directory-of-Cosminexus*/CC/admin/usrconf/*	Y	--	Y	Y	Y
Batch application	Message log	Operation log of cjexecjob, cjkilljob, and cjlistjob command	*batch-application-log-output-directory-(batch.log.directory)*/cjmessage*.log	Y	--	--	C	C
	Maintenance log	Other logs	*batch-application-log-output-directory-(batch.log.directory)*/*/*	Y	--	--	C	C
	Definition information	Option definition file for batch application	*Definition-file-storage-directory-of-batch-application*/usrconf.cfg	Y	--	--	D	D
		User property file for batch application	*Definition-file-storage-directory-of-batch-application*/usrconf.properties	Y	--	--	D	D
Resource adapter version upgrade command (cjrarupdate)	Message log	Operation log	*Installation-directory-of-Cosminexus*/CC/logs/cjrarupdatemessage*.log	Y	--	--	Y	Y
	Other logs	Exception information when an error occurs	*Installation-directory-of-Cosminexus*/CC/logs/cjrarupdateexception*.log	Y	--	--	Y	Y
	Maintenance log	Maintenance information	*Installation-directory-of-Cosminexus*/CC/logs/cjrarupdatemaintenance*.log	Y	--	--	Y	Y
In-process HTTP server	Process log	Process result of in-process HTTP server	*In-process-HTTP-server-access-log-file-(webserver.logger.access_log.inprocess_http.filename)**.log	Y	--	--	A	C
	Module trace	Thread trace information	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/http/maintenance/thr/cjhttp_thr.*.inprocess_http.mm	Y	--	--	A	C
	Communication trace	Communication trace information	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/http/maintenance/comm/cjhttp_comm.*.inprocess_http.mm	Y	--	--	A	C
NIO HTTP server	Access log	Processing results of the NIO HTTP server (HTTP)	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cj_access_niohttp*.log	Y	--	--	A	C
		Processing results of the NIO HTTP server (WebSocket)	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cj_access_websocket*.log	Y	--	--	A	C

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
Migration command (cjenvupdate)	Message log	Operation log of the cjenvupdate command	*Installation-directory-of-Cosminexus*/CC/logs/cjenvupdatemessage*.log	Y	--	--	Y	Y
	Other logs	Exception information of the cjenvupdate command	*Installation-directory-of-Cosminexus*/CC/logs/cjenvupdateexception*.log	Y	--	--	Y	Y
	Maintenance log	Maintenance information of the cjenvupdate command	*Installation-directory-of-Cosminexus*/CC/logs/cjenvupdatemaintenance*.log	Y	--	--	Y	Y
In-process transaction service	Others	In-process transaction service status file	*Status-file-directory-(ejbserver.distributedtx.ots.status.directory1)*/*	--	Y	--	A	C
			Status-file-directory-(ejbserver.distributedtx.ots.status.directory1)/*/*	--	Y	--	A	C
		In-process transaction service spare status file	*Spare-status-file-directory-(ejbserver.distributedtx.ots.status.directory2)*/*	--	Y	--	A	D
			Spare-status-file-directory-(ejbserver.distributedtx.ots.status.directory2)/*/*	--	Y	--	A	D
EJB client	Message log	Operation log	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*/*subdirectory-1-(ejbserver.client.ejb.log)*/*subdirectory-2-(ejbserver.client.log.appid)*/cjclmessage*.log	Y	--	--	C	C
			EJB-client-log-output-directory-(ejb.client.log.directory)/*EJB-client-log-subdirectory-1-(ejb.client.ejb.log)*/*EJB-client-log-subdirectory-2-(ejb.client.log.appid)*/cjclmessage*.log	Y	--	--	D	D
		cjclstartap command operation log	*EJB-client-log-output-directory-(ejb.client.log.directory)*/cjclstartap*.log	Y	--	--	D	D
		cjcldellog command operation log	*Installation-directory-of-Cosminexus*/CC/client/logs/cjcldellog.log	Y	--	--	Y	Y
	Other logs	User output log	*EJB-client-log-output-directory-(ejb.client.log.directory)*/*EJB-client-log-subdirectory-(ejb.client.ejb.log)*/*EJB-client-log-subdirectory2-(ejb.client.log.appid)*/user_out*.log	Y	--	--	D	D

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		User error log	*EJB-client-log-output-directory-(ejb.client.log.directory)*/*EJB-client-log-subdirectory-(ejb.client.ejb.log)*/*EJB-client-log-subdirectory2-(ejb.client.log.appid)*/`user_err*.log`	Y	--	--	D	D
		JavaVM maintenance information, GC log	*EJB-client-log-output-directory-(ejb.client.log.directory)*/*EJB-client-log-subdirectory-(ejb.client.ejb.log)*/*EJB-client-log-subdirectory2-(ejb.client.log.appid)*/`javalog*.log`	Y	--	--	D	D
		Exception information when an error occurs	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*/*subdirectory-1-(ejbserver.client.ejb.log)*/*subdirectory-2-(ejbserver.client.log.appid)*/`cjclexception*.log`	Y	--	--	C	C
			EJB-client-log-output-directory-(ejb.client.log.directory)/*EJB-client-log-subdirectory-1-(ejb.client.ejb.log)*/*EJB-client-log-subdirectory-2-(ejb.client.log.appid)*/`maintenance/cjclmaintenance*.log`	Y	--	--	D	D
		EJB client application user log	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*/`user/*`	Y	--	--	C	C
			EJB-client-log-output-directory-(ejb.client.log.directory)/`user/*`	Y	--	--	D	D
	Maintenance log	Maintenance information	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*/*subdirectory-1-(ejbserver.client.ejb.log)*/*subdirectory-2-(ejbserver.client.log.appid)*/`cjclmaintenance*.log`	Y	--	--	C	C
			EJB-client-log-output-directory-(ejb.client.log.directory)/*EJB-client-log-subdirectory-1-(ejb.client.ejb.log)*/*EJB-client-log-subdirectory-2-(ejb.client.log.appid)*/`maintenance/cjclmaintenance*.log`	Y	--	--	D	D
		EJB container maintenance information	*EJB-client-log-output-directory-(ejb.client.log.directory)*/*EJB-client-log-subdirectory-(ejb.client.ejb.log)*/*EJB-client-log-subdirectory2-(ejb.client.log.appid)*/`maintenance/cjejbcontainer*.log`	Y	--	--	D	D
		Start process standard output information	*EJB-client-log-output-directory-(ejb.client.log.directory)*/*EJB-client-log-subdirectory-(ejb.client.ejb.log)*/*EJB-client-log-subdirectory2-(ejb.client.log.appid)*/`maintenance/cjstdout*.log`	Y	--	--	D	D

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		Start process standard error information	*EJB-client-log-output-directory-(ejb.client.log.directory)*/*EJB-client-log-subdirectory-(ejb.client.ejb.log)*/*EJB-client-log-subdirectory2-(ejb.client.log.appid)*/maintenance/cjstderr*.log	Y	--	--	D	D
		Log operation information	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*/cjlogger.log	Y	--	--	C	C
			EJB-client-log-output-directory-(ejb.client.log.directory)/cjlogger.log	Y	--	--	D	D
	Definition information	Option definition file for EJB client	*EJB-client-definition-file-storage-directory*/usrconf.cfg	Y	--	--	D	D
		User property file for EJB client	*EJB-client-definition-file-storage-directory*/usrconf.properties	Y	--	--	D	D
Redirector (Web server)	Message log	Redirector message log for HWS	*Redirector-log-output-directory-for-HWS-(JkLogFileDir)*/hws_redirect*.log	Y	--	--	A	C
		Redirector message log for previous version compatible HWS	*Redirector-log-output-directory-for-HWS-(JkLogFileDir)*/hws_redirect*.log	Y	--	--	A	C
		Redirector message log for previous version compatible IIS	*Redirector-log-output-directory-for-IIS-(log_file_dir)*/isapi_redirect*.log	Y	--	--	C	C
	Maintenance log	Trace log for maintenance of redirector for HWS	*Redirector-trace-log-output-directory-for-HWS-(JkTraceLogFileDir)*/hws_rd_trace*.log	Y	--	--	A	C
		Trace log for maintenance of redirector for previous version compatible HWS	*Redirector-trace-log-output-directory-for-HWS-(JkTraceLogFileDir)*/hws_rd_trace*.log	Y	--	--	A	C
		Trace log for maintenance of redirector for previous version compatible IIS	*Redirector-trace-log-output-directory-for-IIS-(trace_log_file_dir)*/iis_rd_trace*.log	Y	--	--	C	C
	Definition information	Redirector action definition file for HWS	*Installation-directory-of-Cosminexus*/CC/web/redirector/servers/*server-name*/mod_jk.conf	Y	--	Y	Y	Y
		Worker definition file	*Installation-directory-of-Cosminexus*/CC/web/	Y	--	Y	Y	Y

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
			redirector/servers/*server-name*/workers.properties					
		Redirector action file for previous version compatible HWS	*Installation-directory-of-Cosminexus*/CC/web/redirector/mod_jk.conf	Y	--	Y	Y	Y
		Redirector action definition file for previous version compatible IIS	*Application-Server-installation-directory*/CC/web/redirector/isapi_redirect.conf	Y	--	Y	Y	Y
		Worker definition file for previous version compatibility	*Installation-directory-of-Cosminexus*/CC/web/redirector/workers.properties	Y	--	Y	Y	Y
		Mapping definition file for previous version compatibility	*Application-Server-installation-directory*/CC/web/redirector/uriworkermap.properties	Y	--	Y	Y	Y
TP1/Message Queue - Access	Maintenance log	Method trace, message log related to API and Cosminexus interface	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/connectors/*.log	Y	--	--	A	C
	API Trace	API Trace file	*J2EE-server-working-directory-(ejb.public.directory)*/ejb/*server-name*/mqc.api*	Y	--	--	A	A
TP1 Connector	Message log	TP1 Connector operation log deployed and used as J2EE resource adapter	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/connectors/*.log	Y	--	--	A	C
		TP1 Connector operation log used by including in J2EE application (Normal mode)	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/connectors/*J2EE-application-name*/*.log	Y	--	--	A	C
		TP1 Connector operation log used in Non-managed environment	*TP1Connector-log-output-directory-(jp.co.hitachi_system.tp1connector.logdestination)*/tp1connector*.log	Y	--	--	C	C
TP1/Client/J	Other logs	Debug trace information	*user-home-directory-(user.home)*/TP1clientJ/dcClt*.dmp	--	Y	--	C	C
Manager	Message log	Administration agent log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/adminagent*.log	Y	--	--	A	Y

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		Administration agent start, stop command log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/adminagentctl.exe.*.log	Y	--	--	A	Y
		Integrated message log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/message/mngmessage*.log	Y	--	--	A	Y
		Administration agent service log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/adminagentsv.exe.*.log	Y	--	--	A	Y
		Management agent log and trace	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/mngagent*.*.log	Y	--	--	A	Y
		Management Server service log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/mngsvr.exe.*.log	Y	--	--	A	Y
		Management Server service start, stop command log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/mngsvrctl.exe.*.log	Y	--	--	A	Y
		Management Server log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/mngsvr*.log	Y	--	--	A	Y
	Other logs	Standard error output of Administration agent	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/adminagent.err.*.log	Y	--	--	A	Y
		Standard output of Administration agent	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/adminagent.out.*.log	Y	--	--	A	Y
		Standard command line error output of Administration agent	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/adminagent.err	Y	--	--	A	Y
		Console log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/processConsole*.log	Y	--	--	A	Y
		Administration agent service standard output	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/adminagentsv.exe.out	Y	--	--	A	Y
		Administration agent service standard error output	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/adminagentsv.exe.err	Y	--	--	A	Y

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		Management Server service standard error output	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/mngsvr.exe.err	Y	--	--	A	Y
		Management Server service standard output	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/mngsvr.exe.out	Y	--	--	A	Y
		Configuration file for automatic allocation used with the Explicit Memory Management functionality	*Installation-directory-of-Cosminexus*/CC/server/usrconf/ejb/*server-name*/auto_explicit_memory.cfg	Y	--	Y	C	C
		User-extended trace based performance analysis configuration file used for user-extended trace based performance analysis	*Installation-directory-of-Cosminexus*/CC/server/usrconf/ejb/*server-name*/userprf.cfg	Y	--	Y	C	C
		Definition file for the setup commands of Setup Wizard	*Installation-directory-of-Cosminexus*/manager/setup/config/*	Y	--	Y	Y	Y
		Setup command log for Setup Wizard	*Installation-directory-of-Cosminexus*/manager/setup/log/*.log	Y	--	--	C	C
	Maintenance log	Command maintenance log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/maintenance/mngcmd*.log	--	Y	--	Y	Y
		Administration agent maintenance log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/maintenance/adminagent*.log	Y	--	--	Y	Y
		Maintenance log in RMI process executed by Administration Agent	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/maintenance/mngrmi*.log	--	Y	--	Y	Y
		Management Server maintenance log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/maintenance/mngsvr*.log	Y	--	--	Y	Y
	Definition information	Administration agent property file	*Installation-directory-of-Cosminexus*/manager/config/adminagent.properties	Y	--	Y	Y	Y

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		Administration Agent option definition file	*Installation-directory-of-Cosminexus*/manager/config/adminagentuser.cfg	Y	--	Y	Y	Y
		Administration Agent setup file	*Installation-directory-of-Cosminexus*/manager/config/adminagent.xml	Y	--	Y	Y	Y
		Management agent property file	*Installation-directory-of-Cosminexus*/manager/config/mngagent.*server-name*.properties	Y	--	Y	Y	Y
		Management Server environment setup file	*Installation-directory-of-Cosminexus*/manager/config/mserver.properties	Y	--	Y	Y	Y
		Option definition file for Management Server	*Installation-directory-of-Cosminexus*/manager/config/mserver.cfg	Y	--	Y	Y	Y
		Environment variable setup file for Management Server	*Installation-directory-of-Cosminexus*/manager/config/mserverenv.cfg	Y	--	Y	Y	Y
		Manager setup file	*Installation-directory-of-Cosminexus*/manager/config/manager.cfg	Y	--	Y	Y	Y
		Property file for Management action execution	*Installation-directory-of-Cosminexus*/manager/config/maction.properties	Y	--	Y	Y	Y
		Property file for Management event issue	*Installation-directory-of-Cosminexus*/manager/config/mevent.*server-name*.properties	--	Y	Y	Y	Y
		Message ID list file for Management event issue	*Message-ID-list-file-for-Management-event-issue-(manager.mevent.message_id.list)*	--	Y	Y	D	D
		Client-side definition file of the mngsvrutil command	*User-home-directory-(user.home)*/.mngsvrutilrc	--	Y	Y	C	C
		Server-side definition file of the mngsvrutil command	*Installation-directory-of-Cosminexus*/manager/config/mngsvrutil.properties	Y	--	Y	Y	Y
		Client-side definition file of mngsvrutil command	*Installation-directory-of-Cosminexus*/manager/config/mngsvrutilcl.properties	Y	--	Y	Y	Y

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		Monitor start command setup file for JP1/IM linkage	*user-home-directory-(user.home)*/.mngsvrmonitorrc	--	Y	Y	C	C
		System log message mapping file for JP1/IM linkage	*Installation-directory-of-Cosminexus*/manager/config/mserver.jp1event.system.mapping.properties	Y	--	Y	Y	Y
			Installation-directory-of-Cosminexus/manager/config/manager.jp1event.system.mapping.properties	Y	--	Y	Y	Y
			Installation-directory-of-Cosminexus/manager/config/manager.*server-name*.jp1event.system.mapping.properties	Y	--	Y	Y	Y
	Performance, error analysis trace	PRF Trace	*Directory-to-output-temporary-PRF-trace-file-(adminagent.prftrace_dir)*/*.zip	--	Y	--	A	D
	Internal interface trace	Integrated trace log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/trace/mngtrace*.log	Y	--	--	Y	Y
	Others	OS state information etc	*Installation-directory-of-Cosminexus*/manager/tmp/*	Y	--	--	Y	Y
Smart Composer	Definition information	Server settings property file	*Installation-directory-of-Cosminexus*/manager/config/cmxserver.properties	Y	--	Y	Y	Y
		Client settings property file	*User-home-directory-(user.home)*/.cmxrc	--	Y	Y	C	C
		Client common settings property file	*Installation-directory-of-Cosminexus*/manager/config/cmxclient.properties	Y	--	Y	Y	Y
		Load balancer definition property file	*Installation-directory-of-Cosminexus*/manager/config/lb.properties	Y	--	Y	Y	Y
Integrated user management	Definition information	JAAS configuration file	*Installation-directory-of-Cosminexus*/manager/config/jaas.conf	Y	--	Y	Y	Y
		Configuration file for integrated user management	*Installation-directory-of-Cosminexus*/manager/config/ua.conf	Y	--	Y	Y	Y
	API Trace	Trace of integrated user management	*Integrated-user-management-trace-file-name-(com.cosminexus.admin.auth.trace.prefix)*.*.log	--	Y	--	D	D

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
Virtual server manager in the 08-50 mode	Definition information	Virtual server manager property file	*Installation-directory-of-Cosminexus*/manager/vmx/config/vmx.properties	Y	--	Y	D	Y
		Client common settings property file of virtual server manager	*Installation-directory-of-Cosminexus*/manager/vmx/config/vmxclient.properties	Y	--	Y	D	Y
		Option definition file for VMware vCenter Server connection processing	*Installation-directory-of-Cosminexus*/manager/vmx/config/vmx_avcs_usrconf.cfg	Y	--	Y	D	Y
		Property file for VMware vCenter Server connection processing	*Installation-directory-of-Cosminexus*/manager/vmx/config/vmx_avcs_usrconf.properties	Y	--	Y	D	Y
		Common definition file for VMware vCenter Server connection processing	*Installation-directory-of-Cosminexus*/manager/vmx/config/vmx_avcs_cjwconf.properties	Y	--	Y	D	Y
	Other log	Log of the cjclstartap command used with the virtual server manager	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/vmx_avcs_logs/**	--	Y	--	D	Y
		Log information collected from the virtual server	*Virtual-server-manager-log-directory-in-08-50-mode-(vmx.log.dir)*/mngunit/*/*.zip	--	Y	--	D	C
Virtual server manager	Definition information	Virtual server manager property file	*Installation-directory-of-Cosminexus*/manager/vmi/config/vmi.properties	Y	--	Y	D	Y
		Client settings property file of virtual server manager	*User-home-directory-(user.home)*/.vmirc	--	Y	Y	C	C
		Client common settings property file of virtual server manager	*Installation-directory-of-Cosminexus*/manager/vmi/config/vmiclient.properties	Y	--	Y	D	Y
		Load balancer connection settings property file	*Installation-directory-of-Cosminexus*/manager/vmi/config/lb/*.properties	Y	--	Y	D	Y

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
Server Communication Agent	Message log	Server Communication Agent log	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*/sinaviagent*.log	Y	--	--	D	Y
		Service log of Server Communication Agent	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*/sinaviagentsv*.log	Y	--	--	D	C
		snactl command log	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*/snactl*.log	Y	--	--	D	C
	Other logs	Error information during Server Communication Agent startup	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*/sinaviagent.err	Y	--	--	D	C
		Standard output of Server Communication Agent	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*/sinaviagent.out	Y	--	--	D	C
		Console output information of command processes started by Server Communication Agent	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*/processConsole*.log	Y	--	--	D	C
	Maintenance log	Maintenance log of Server Communication Agent	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*/maintenance/sinaviagent*.log	--	Y	--	D	C
		Maintenance log of Server Communication Agent service	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*/maintenance/sinaviagentsv*.log	--	Y	--	D	C
		Maintenance log of snactl command	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*/maintenance/snactl*.log	--	Y	--	D	C
		Javalog of Server Communication Agent	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*/maintenance/sinaviagent.javalog*.log	--	Y	--	D	C
	Definition information	Option definition file for Server Communication Agent	*Installation-directory-of-Cosminexus*/sinagent/config/sinaviagent.cfg	Y	--	Y	D	Y
		Server Communication Agent property file	*Installation-directory-of-Cosminexus*/sinagent/config/sinaviagent.properties	Y	--	Y	D	Y

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
Cosminexus Web Services - Base	Message log	Message log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjmessage*.log	Y	--	--	A	C
			Web-container-server-log-output-directory-(web.server.log.directory)/cjweb_message*.log	Y	--	--	C	C
			EJB-client-log-output-directory-(ejbserver.client.log.directory)/*subdirectory-1-(ejbserver.client.ejb.log)*/*subdirectory-2-(ejbserver.client.log.appid)*/cjclmessage*.log	Y	--	--	C	C
		Server trace, client trace, default trace, application log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/WS/*.log	Y	--	--	A	C
			Web-container-server-log-output-directory-(web.server.log.directory)/WS/*.log	Y	--	--	C	Y
		Client trace, default trace, provided command trace, application log	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*/*subdirectory-1-(ejbserver.client.ejb.log)*/*subdirectory-2-(ejbserver.client.log.appid)*/WS/*.log	Y	--	--	C	C
		JAXR Trace	*JAXR-trace-file-name-(com.cosminexus.xml.registry.trace.file_path)**.log	Y	--	--	C	C
	Definition information	Common definition file	*Installation-directory-of-Cosminexus*/c4web/conf/c4webcom.cfg	Y	--	Y	Y	Y
		Server definition file	*Installation-directory-of-Cosminexus*/c4web/conf/c4websv.cfg	Y	--	Y	Y	Y
		Service deploy definition	*Installation-directory-of-Cosminexus*/CC/server/public/web/*server-name*/*context-root*/WEB-INF/server-config.xml	Y	--	Y	Y	Y
			Installation-directory-of-Cosminexus/CC/web/containers/*server-name*/webapps/*context-root*/WEB-INF/server-config.xml	Y	--	Y	Y	Y
		Client definition file	*Installation-directory-of-Cosminexus*/CC/server/public/web/*server-name*/*context-root*/WEB-INF/classes/c4webcl.properties	Y	--	Y	Y	Y
			Installation-directory-of-Cosminexus/CC/web/containers/*server-name*/webapps/*context-root*/WEB-INF/classes/c4webcl.properties	Y	--	Y	Y	Y

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
Cosminexus JAX-WS	Message log	`cjwsimport` command operation log	*Cosminexus-installation-directory*`/jaxws/logs/cjwsimport*.log`	Y	--	--	Y	Y
		`apt` command operation log	*Cosminexus-installation-directory*`/jaxws/logs/cjwapt*.log`	Y	--	--	Y	Y
		Service and client operation log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/CJW/cjwmessage*.log`	Y	--	--	A	C
		Client operation log	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*`/`*subdirectory-1-(ejbserver.client.ejb.log)*`/`*subdirectory-2-(ejbserver.client.log.appid)*`/CJW/cjwmessage*.log`	Y	--	--	C	C
		Service and client operation log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/CJR/cjrmessage*.log`	Y	--	--	A	C
	Other logs	Service and client communication log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/CJW/cjwtransport*.log`	Y	--	--	A	C
		Client communication log	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*`/`*subdirectory-1-(ejbserver.client.ejb.log)*`/`*subdirectory-2-(ejbserver.client.log.appid)*`/CJW/cjwtransport*.log`	Y	--	--	C	C
		Service and client communication log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/CJR/cjrtransport*.log`	Y	--	--	A	C
	Exception log	`cjwsimport` command exception log	*Cosminexus-installation-directory*`/jaxws/logs/cjwsimportex*.log`	Y	--	--	Y	Y
		`apt` command exception log	*Cosminexus-installation-directory*`/jaxws/logs/cjwaptex*.log`	Y	--	--	Y	Y
		Service and client exception log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/CJW/cjwexception*.log`	Y	--	--	A	C
		Client exception log	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*`/`*subdirectory-1-(ejbserver.client.ejb.log)*`/`*subdirectory-2-(ejbserver.client.log.appid)*`/CJW/cjwexception*.log`	Y	--	--	C	C
		Service and client exception log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/CJR/cjrexception*.log`	Y	--	--	A	C

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
	Definition information	Action definition file	*Installation-directory-of-Cosminexus*/jaxws/conf/*.properties	Y	--	Y	Y	Y
		Process wise action definition file	*Process-wise-action-definition-file-(com.cosminexus.jaxws.confpath)*	Y	--	Y	D	D
		Action definition file	*Installation-directory-of-Cosminexus*/jaxrs/conf/*.properties	Y	--	Y	C	C
		Process-wise action definition file	*Process-wise-action-definition-file-(com.cosminexus.jaxrs.confpath)*	Y	--	Y	D	D
	Definition file	web.xml	*J2EE-server-working-directory-(ejb.public.directory)*/web/*server-name*/*/WEB-INF/web.xml	--	Y	Y	A	A
		cosminexus-jaxws.xml	*J2EE-server-working-directory-(ejb.public.directory)*/web/*server-name*/*/WEB-INF/cosminexus-jaxws.xml	--	Y	Y	A	A
		Handler chain setup file	*J2EE-server-working-directory-(ejb.public.directory)*/web/*server-name*/**	--	Y	Y	D	D
		WSDL	*J2EE-server-working-directory-(ejb.public.directory)*/web/*server-name*/*/WEB-INF/wsdl/*	--	Y	Y	A	A
		Catalog file	*J2EE-server-working-directory-(ejb.public.directory)*/web/*server-name*/*/WEB-INF/classes/META-INF/jax-ws-catalog.xml	--	Y	Y	A	A
			J2EE-server-working-directory-(ejb.public.directory)/ejb/*server-name*/apps/*/*/*/jax-ws-catalog.xml	--	Y	Y	A	A
Cosminexus JPA provider	Message log	Cosminexus JPA provider operation log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjpa/cjpaoperation*.log	Y	--	--	A	C
		PRF daemon and PRF command log	*PRF-spool-directory-(prfspool)*/log/PRF-identifier/ctmlog*	Y	--	--	A	C
		Resource adapter operation log deployed and used as J2EE resource adapter	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/connectors/*.log	Y	--	--	A	C
		Message log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjmessage*.log	Y	--	--	A	C

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
	Other logs	Exception information when an error occurs	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjexception*.log	Y	--	--	A	C
	Definition information	Option definition file for J2EE server	*Installation-directory-of-Cosminexus*/CC/server/usrconf/ejb/*server-name*/*.cfg	Y	--	Y	Y	Y
		User property file for the J2EE server	*Installation-directory-of-Cosminexus*/CC/server/usrconf/ejb/*server-name*/*.properties	Y	--	Y	Y	Y
		Security policy file for J2EE server	*Installation-directory-of-Cosminexus*/CC/server/usrconf/ejb/*server-name*/*.policy	Y	--	Y	Y	Y
	User data	Contents of EJB server working directory	*J2EE-server-working-directory-(ejb.public.directory)*/ejb/*server-name*/**	--	Y	--	D	D
		Contents of Web container working directory	*J2EE-server-working-directory-(ejb.public.directory)*/web/*server-name*/**	--	Y	--	D	D
Cosminexus JMS provider	Message log	CJMSP Broker log	*Installation-directory-of-Cosminexus*/CC/cjmsp/var/instances/*instance-name*/log/*.log	Y	--	--	C	C
		Management command log	*Installation-directory-of-Cosminexus*/CC/cjmsp/var/admin/log/*.log	Y	--	--	C	C
		CJMSP resource adapter log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjms/*resource-adapter-name*/*.log	Y	--	--	A	C
	Definition information	CJMSP Broker Common Property File	*Installation-directory-of-Cosminexus*/CC/cjmsp/lib/props/broker/commonconfig.properties	Y	--	Y	Y	Y
		CJMSP Broker Property File	*Installation-directory-of-Cosminexus*/CC/cjmsp/var/instances/*instance-name*/props/config.properties	Y	--	Y	C	C
		Management command definition file	*Installation-directory-of-Cosminexus*/CC/cjmsp/var/admin/config/admin.properties	Y	--	Y	C	C

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For collection method
  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

Table A–6: Collection method related to Cosminexus Component Container (In UNIX)

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
J2EE server	Message log	Operation log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjmessage*.log	Y	--	--	A	C
		Log operation log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjlogger.log	Y	--	--	A	C
		Resource adapter operation log deployed and used as J2EE resource adapter.	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/connectors/*.log	Y	--	--	A	C
		Resource adapter operation log used by including in J2EE application (normal mode)	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/connectors/*J2EE-application-name*/*.log	Y	--	--	A	C
		Resource adapter operation log used by including in J2EE application (test mode)	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/connectors/TEST#*J2EE-application-name*/*.log	Y	--	--	A	C
		Web servlet log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/web_servlet*.log	Y	--	--	A	C
	Other logs	User output log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/user_out*.log	Y	--	--	A	C
		User error log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/user_err*.log	Y	--	--	A	C
		JavaVM maintenance information and GC log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/javalog*.log	Y	--	--	A	C

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		JavaVM Explicit Memory Management functionality event log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/ehjavalog*.log	Y	--	--	A	C
		Development check log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjdevelopment*.log	Y	--	--	A	C
		Exception information when an error occurs	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjexception*.log	Y	--	--	A	C
		JavaVM event log	*J2EE-server-working-directory-(ejb.public.directory)*/ejb/*server-name*/hs_err*	Y	--	--	A	A
			J2EE-server-working-directory-(ejb.public.directory)/ejb/*server-name*/replay_pid*.log	Y	--	--	A	A
		User log of J2EE applications	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/user/*	Y	--	--	A	C
	Maintenance log	Maintenance information	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CC/maintenance/cjmaintenance*.log	Y	--	--	A	C
		Console message	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CC/maintenance/cjconsole*.log	Y	--	--	A	C
		EJB container maintenance information	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CC/maintenance/cjejbcontainer*.log	Y	--	--	A	C
		Web container maintenance information	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CC/maintenance/cjwebcontainer*.log	Y	--	--	A	C
		Start process standard output information	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CC/maintenance/cjstdout*.log	Y	--	--	A	C

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		Start process standard error information	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CC/maintenance/cjstderr*.log	Y	--	--	A	C
		Termination process information	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CC/maintenance/cj_shutdown*.log	Y	--	--	A	C
	Dump	Thread dump	*J2EE-server-working-directory-(ejb.public.directory)*/ejb/*server-name*/javacore*	Y	--	--	A	A
		core dump	*J2EE-server-working-directory-(ejb.public.directory)*/ejb/*server-name*/core*	--	Y	--	A	A
	Statistical information	Operation information file	*statistics-file-output-destination-directory-(ejbserver.management.stats_file.dir)*/*	--	Y	--	A	C
		Memory monitoring log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/watch/cjmemorywatch*.log	Y	--	--	A	C
		File descriptor monitoring log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/watch/cjfiledescriptorwatch*.log	Y	--	--	A	C
		Thread monitoring log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/watch/cjthreadwatch*.log	Y	--	--	A	C
		Thread dump monitoring log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/watch/cjthreaddumpwatch*.log	Y	--	--	A	C
		HTTP request pending queue monitoring log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/watch/cjrequestqueuewatch*.log	Y	--	--	A	C
		HTTP session count monitoring log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/watch/	Y	--	--	A	C

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
			cjhttpsessionwatch*.log					
		Connection pool monitoring log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/watch/cjconnectionpoolwatch*.log	Y	--	--	A	C
	Definition information	Option definition file for J2EE server	*Installation-directory-of-Cosminexus*/CC/server/usrconf/ejb/*server-name*/usrconf.cfg	Y	--	Y	Y	Y
		User property file for the J2EE server	*Installation-directory-of-Cosminexus*/CC/server/usrconf/ejb/*server-name*/usrconf.properties	Y	--	Y	Y	Y
		Security policy file for J2EE server	*Installation-directory-of-Cosminexus*/CC/server/usrconf/ejb/*server-name*/server.policy	Y	--	Y	Y	Y
		Backup of various definition files created in MNG	*Installation-directory-of-Cosminexus*/CC/server/usrconf/ejb/*server-name*/*	--	Y	--	Y	Y
		Protected area list file	*Installation-directory-of-Cosminexus*/CC/server/usrconf/criticalList.cfg	Y	--	Y	Y	Y
		Resource setup information	*J2EE-server-work-directory-(ejb.public.directory)*/ejb/*server-name*/import/**	--	Y	--	D	D
			J2EE-server-work-directory-(ejb.public.directory)/ejb/*server-name*/rars/**	--	Y	--	D	D
		Maintenance information	*Installation-directory-of-Cosminexus*/CC/server/version/**	Y	--	Y	Y	Y
	Others	Log showing J2EE server start, stop or abnormal termination	*UNIX-syslog-(syslog)*	Y	--	--	C	C
	User data	Contents of EJB server working directory	*J2EE-server-working-directory-(ejb.public.directory)*/ejb/*server-name*/**	--	Y	--	D	D
		Contents of Web container working directory	*J2EE-server-working-directory-(ejb.public.directory)*/web/*server-name*/**	--	Y	--	D	D

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		Contents of temporary directory for JSP	*Temporary-directory-for-JSP-(webserver.work.directory)*`/**`	--	Y	--	D	D
Server management commands	Message log	Operation log	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*`/cjmessage*.log`	Y	--	--	C	C
		Log operation log	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*`/cjlogger.log`	Y	--	--	C	C
		Operation log in compatible mode	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*`/*message*.log`	Y	--	--	C	C
	Other logs	Exception information when an error occurs	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*`/cjexception*.log`	Y	--	--	C	C
		Exception information when an error occurs in compatible mode	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*`/*exception*.log`	Y	--	--	C	C
	Maintenance log	Maintenance information	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*`/CC/maintenance/cjmaintenance*.log`	Y	--	--	C	C
		Console message	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*`/CC/maintenance/cjconsole*.log`	Y	--	--	C	C
		Server management command maintenance information	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*`/CC/maintenance/cjserveradmin*.log`	Y	--	--	C	C
		Other logs	*server-management-command-log-output-directory-(admin_ejb.server.log.directory)*`/*/*`	Y	--	--	C	C
			server-management-command-log-output-directory-(admin_ejb.server.log.directory)`/*/*/*`	Y	--	--	C	C
			server-management-command-log-output-directory-	--	Y	--	C	C

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
			(admin_ejb.server.log.directory)/*/*/*/*					
	Definition information	Definition file for server management command	*Installation-directory-of-Cosminexus*/CC/admin/usrconf/*	Y	--	--	Y	Y
Batch application	Message log	Operation log of cjexecjob, cjkilljob, and cjlistjob command	*batch-application-log-output-directory-(batch.log.directory)*/cjmessage*.log	Y	--	--	C	C
	Maintenance log	Other logs	*batch-application-log-output-directory-(batch.log.directory)*/*/*	Y	--	--	C	C
	Definition information	Option definition file for batch application	*Definition-file-storage-directory-of-batch-application*/usrconf.cfg	Y	--	--	D	D
		User property file for batch application	*Definition-file-storage-directory-of-batch-application*/usrconf.properties	Y	--	--	D	D
Resource adapter version upgrade command (cjrarupdate)	Message log	Operation log	*Installation-directory-of-Cosminexus*/CC/logs/cjrarupdatemessage*.log	Y	--	--	Y	Y
	Other logs	Exception information when an error occurs	*Installation-directory-of-Cosminexus*/CC/logs/cjrarupdateexception*.log	Y	--	--	Y	Y
	Maintenance log	Maintenance information	*Installation-directory-of-Cosminexus*/CC/logs/cjrarupdatemaintenance*.log	Y	--	--	Y	Y
In-process HTTP server	Access log	Process result of in-process HTTP server	*In-process-HTTP-server-access-log-file-(webserver.logger.access_log.inprocess_http.filename)*.log	Y	--	--	A	C
	Module trace	Thread trace information	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/http/maintenance/thr/cjhttp_thr.*.inprocess_http.mm	Y	--	--	A	C
	Communication trace	Communication trace information	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/http/maintenance/comm/cjhttp_comm.*.inprocess_http.mm	Y	--	--	A	C

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
NIO HTTP server	Access log	Processing results of the NIO HTTP server (HTTP)	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cj_access_niohttp*.log	Y	--	--	A	C
		Processing results of the NIO HTTP server (WebSocket)	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cj_access_websocket*.log	Y	--	--	A	C
Migration command (cjenvupdate)	Message log	Operation log of the cjenvupdate command	*Installation-directory-of-Cosminexus*/CC/logs/cjenvupdatemessage*.log	Y	--	--	Y	Y
	Other logs	Exception information of the cjenvupdate command	*Installation-directory-of-Cosminexus*/CC/logs/cjenvupdateexception*.log	Y	--	--	Y	Y
	Maintenance log	Maintenance information of the cjenvupdate command	*Installation-directory-of-Cosminexus*/CC/logs/cjenvupdatemaintenance*.log	Y	--	--	Y	Y
Command for changing CC Administrator	Message log	Operation log of the cjenvsetup command	*Application-Server-installation-directory*/CC/logs/cjenvsetupmessage*.log	Y	--	--	Y	Y
	Other	File configuration information before executing the cjenvsetup command	*Application-Server-installation-directory*/CC/logs/before_cjenvsetup_files*.txt	Y	--	--	Y	Y
		File configuration information after executing the cjenvsetup command	*Application-Server-installation-directory*/CC/logs/after_cjenvsetup_files*.txt	Y	--	--	Y	Y
In-process transaction service	Others	In-process transaction service status file	*Status-file-directory-(ejbserver.distributedtx.ots.status.directory1)*/*	--	Y	--	A	C
			Status-file-directory-(ejbserver.distributedtx.ots.status.directory1)/*/*	--	Y	--	A	C
		In-process transaction service spare status file	*Spare-status-file-directory-(ejbserver.distributedtx.ots.status.directory2)*/*	--	Y	--	A	D

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
			Spare-status-file-directory-(ejbserver.distributedtx.ots.status.directory2)`/*/*`	--	Y	--	A	D
EJB client	Message log	Operation log	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*`/`*subdirectory-1-(ejbserver.client.ejb.log)*`/`*subdirectory-2-(ejbserver.client.log.appid)*`/cjclmessage*.log`	Y	--	--	C	C
			EJB-client-log-output-directory-(ejb.client.log.directory)`/`*EJB-client-log-subdirectory-1-(ejb.client.ejb.log)*`/`*EJB-client-log-subdirectory-2-(ejb.client.log.appid)*`/cjclmessage*.log`	Y	--	--	D	D
		`cjclstartap` command operation log	*EJB-client-log-output-directory-(ejb.client.log.directory)*`/cjclstartap*.log`	Y	--	--	D	D
		`cjcldellog` command operation log	*Installation-directory-of-Cosminexus*`/CC/client/logs/cjcldellog.log`	Y	--	--	Y	Y
	Other logs	User output log	*EJB-client-log-output-directory-(ejb.client.log.directory)*`/`*EJB-client-log-subdirectory-(ejb.client.ejb.log)*`/`*EJB-client-log-subdirectory2-(ejb.client.log.appid)*`/user_out*.log`	Y	--	--	D	D
		User error log	*EJB-client-log-output-directory-(ejb.client.log.directory)*`/`*EJB-client-log-subdirectory-(ejb.client.ejb.log)*`/`*EJB-client-log-subdirectory2-(ejb.client.log.appid)*`/user_err*.log`	Y	--	--	D	D
		JavaVM maintenance information, GC log	*EJB-client-log-output-directory-(ejb.client.log.directory)*`/`*EJB-client-log-subdirectory-(ejb.client.ejb.log)*`/`*EJB-client-log-subdirectory2-(ejb.client.log.appid)*`/javalog*.log`	Y	--	--	D	D
		Exception information when an error occurs	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*`/`*subdirectory-1-(ejbserver.client.ejb.log)*`/`*subdirectory-2-(ejbserver.client.log.appid)*`/cjclexception*.log`	Y	--	--	C	C

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
			EJB-client-log-output-directory-(ejb.client.log.directory)`/`*EJB-client-log-subdirectory-1-(ejb.client.ejb.log)*`/`*EJB-client-log-subdirectory-2-(ejb.client.log.appid)*`/maintenance/cjclmaintenance*.log`	Y	--	--	D	D
		EJB client application user log	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*`/user/*`	Y	--	--	C	C
			EJB-client-log-output-directory-(ejb.client.log.directory)`/user/*`	Y	--	--	D	D
	Maintenance log	Maintenance information	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*`/`*subdirectory-1-(ejbserver.client.ejb.log)*`/`*subdirectory-2-(ejbserver.client.log.appid)*`/cjclmaintenance*.log`	Y	--	--	C	C
			EJB-client-log-output-directory-(ejb.client.log.directory)`/`*EJB-client-log-subdirectory-1-(ejb.client.ejb.log)*`/`*EJB-client-log-subdirectory-2-(ejb.client.log.appid)*`/maintenance/cjclmaintenance*.log`	Y	--	--	D	D
		EJB container maintenance information	*EJB-client-log-output-directory-(ejb.client.log.directory)*`/`*EJB-client-log-subdirectory-(ejb.client.ejb.log)*`/`*EJB-client-log-subdirectory2-(ejb.client.log.appid)*`/maintenance/cjejbcontainer*.log`	Y	--	--	D	D
		Start process standard output information	*EJB-client-log-output-directory-(ejb.client.log.directory)*`/`*EJB-client-log-subdirectory-(ejb.client.ejb.log)*`/`*EJB-client-log-subdirectory2-(ejb.client.log.appid)*`/maintenance/cjstdout*.log`	Y	--	--	D	D
		Start process standard error information	*EJB-client-log-output-directory-(ejb.client.log.directory)*`/`*EJB-client-log-subdirectory-(ejb.client.ejb.log)*`/`*EJB-client-log-subdirectory2-(ejb.client.log.appid)*`/maintenance/cjstderr*.log`	Y	--	--	D	D

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		Log operation information	*EJB-client-log-output-directory-(ejb.client.log.directory)*/cjlogger.log	Y	--	--	D	D
	Definition information	Option definition file for EJB client	*EJB-client-definition-file-storage-directory*/usrconf.cfg	Y	--	--	D	D
		User property file for EJB client	*EJB-client-definition-file-storage-directory*/usrconf.properties	Y	--	--	D	D
Redirector (Web server)	Message log	Redirector message log for HWS	*Redirector-log-output-directory-for-HWS-(JkLogFileDir)*/hws_redirect*.log	Y	--	--	A	C
		Redirector message log for previous version compatible HWS	*Redirector-log-output-directory-for-HWS-(JkLogFileDir)*/hws_redirect*.log	Y	--	--	A	C
	Maintenance log	Trace log for maintenance of redirector for HWS	*Redirector-trace-log-output-directory-for-HWS-(JkTraceLogFileDir)*/hws_rd_trace*.log	Y	--	--	A	C
		Trace log for maintenance of redirector for previous version compatible HWS	*Redirector-trace-log-output-directory-for-HWS-(JkTraceLogFileDir)*/hws_rd_trace*.log	Y	--	--	A	C
	Definition information	Redirector action definition file for HWS	*Installation-directory-of-Cosminexus*/CC/web/redirector/servers/*server-name*/mod_jk.conf	Y	--	Y	Y	Y
		Worker definition file	*Installation-directory-of-Cosminexus*/CC/web/redirector/servers/*server-name*/workers.properties	Y	--	Y	Y	Y
		Redirector action definition file for previous version compatible HWS	*Installation-directory-of-Cosminexus*/CC/web/redirector/mod_jk.conf	Y	--	--	Y	Y
		Worker definition file for previous version compatibility	*Installation-directory-of-Cosminexus*/CC/web/redirector/workers.properties	Y	--	--	Y	Y
TP1/Message Queue - Access	Maintenance log	Method trace, message log related to API	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/connectors/*.log	Y	--	--	A	C

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		and Cosminexus interface						
	API Trace	API Trace file	*J2EE-server-working-directory-(ejb.public.directory)*`/ejb/`*server-name*`/mqc.api*`	Y	--	--	A	A
uCosminexus TP1 Connector	Message log	TP1 Connector operation log deployed and used as J2EE resource adapter	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/connectors/*.log`	Y	--	--	A	C
		TP1 Connector operation log used by including in J2EE application (Normal mode)	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/connectors/`*J2EE-application-name*`/*.log`	Y	--	--	A	C
		TP1 Connector operation log used in Non-managed environment	*TP1-Connectorlog-output-directory-(jp.co.hitachi_system.tp1connector.logdestination)*`/tp1connector*.log`	Y	--	--	C	C
TP1/Client/J	Other logs	Debug trace information	*user-home-directory-(user.home)*`/TP1clientJ/dcClt*.dmp`	--	Y	--	C	C
Cosminexus Manager	Message log	Integrated message log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*`/message/mngmessage*.log`	Y	--	--	A	Y
		Administration agent log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*`/adminagent*.log`	Y	--	--	A	Y
		Administration agent start, stop command log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*`/adminagentctl.*.log`	Y	--	--	A	Y
		Management agent log and trace	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*`/mngagent*.*.log`	Y	--	--	A	Y
		Management Server startup command log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*`/mngsvrctlstart.*.log`	Y	--	--	A	Y
		Management Server stop command log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*`/mngsvrctlstop.*.log`	Y	--	--	A	Y
		Management Server setup command log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*`/mngsvrctlsetup.*.log`	Y	--	--	A	Y

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		Management Server log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/mngsvr*.log	Y	--	--	A	Y
	Other logs	Standard error output of Administration agent	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/adminagent.err.*.log	Y	--	--	A	Y
		Standard output of Administration agent	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/adminagent.out.*.log	Y	--	--	A	Y
		Standard command line error output of Administration agent	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/adminagent.err	Y	--	--	A	Y
		Console log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/processConsole*.log	Y	--	--	A	Y
		Configuration file for automatic allocation used with the Explicit Memory Management functionality	*Installation-directory-of-Cosminexus*/CC/server/usrconf/ejb/*server-name*/auto_explicit_memory.cfg	Y	--	Y	C	C
		User-extended trace based performance analysis configuration file used for user-extended trace based performance analysis	*Installation-directory-of-Cosminexus*/CC/server/usrconf/ejb/*server-name*/userprf.cfg	Y	--	Y	C	C
		Definition file for the setup commands of Setup Wizard	*Installation-directory-of-Cosminexus*/manager/setup/config/*	Y	--	Y	Y	Y
		Setup command log for Setup Wizard	*Installation-directory-of-Cosminexus*/manager/setup/log/*.log	Y	--	--	C	C
	Maintenance log	Command maintenance log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*/maintenance/mngcmd*.log	--	Y	--	Y	Y
		Maintenance log in RMI process executed by	*Manager-log-output-directory-(com.cosminexus.manager.log.dir*	--	Y	--	Y	Y

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		Administration Agent	*)*`/maintenance/mngrmi*.log`					
		Administration agent maintenance log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*`/maintenance/adminagent*.log`	Y	--	--	Y	Y
		`mngenvsetup` command execution log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*`/maintenance/mngenvsetup*.log`	--	Y	--	Y	Y
		Management Server maintenance log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*`/maintenance/mngsvr*.log`	Y	--	--	Y	Y
	Definition information	Administration agent property file	*Installation-directory-of-Cosminexus*`/manager/config/adminagent.properties`	Y	--	Y	Y	Y
		Settings file for Administration agent automatic start	*Installation-directory-of-Cosminexus*`/manager/config/AdminAgentrc`	Y	--	Y	Y	Y
		Option definition file for Administration agent	*Installation-directory-of-Cosminexus*`/manager/config/adminagentuser.cfg`	Y	--	Y	Y	Y
		Administration agent setup file	*Installation-directory-of-Cosminexus*`/manager/config/adminagent.xml`	Y	--	Y	Y	Y
		Management agent property file	*Installation-directory-of-Cosminexus*`/manager/config/mngagent.`*server-name*`.properties`	Y	--	Y	Y	Y
		Management Server environment setup file	*Installation-directory-of-Cosminexus*`/manager/config/mserver.properties`	Y	--	Y	Y	Y
		Option definition file for Management Server	*Installation-directory-of-Cosminexus*`/manager/config/mserver.cfg`	Y	--	Y	Y	Y
		Environment variable definition file for Management Server	*Installation-directory-of-Cosminexus*`/manager/config/mserverenv.cfg`	Y	--	Y	Y	Y
		Manager setup file	*Installation-directory-of-Cosminexus*`/manager/config/manager.cfg`	Y	--	Y	Y	Y

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		Property file for Management action execution	*Installation-directory-of-Cosminexus*`/manager/config/maction.properties`	Y	--	Y	Y	Y
		Property file for Management event issue	*Installation-directory-of-Cosminexus*`/manager/config/mevent.`*server-name*`.properties`	--	Y	Y	Y	Y
		Message ID list file for Management event issue	*Message-ID-list-file-for-Management-event-issue-(manager.mevent.message_id.list)*	--	Y	Y	D	D
		Client-side definition file of the `mngsvrutil` command	*user-home-directory-(user.home)*`/.mngsvrutilrc`	--	Y	Y	C	C
		Server-side definition file of the `mngsvrutil` command	*Installation-directory-of-Cosminexus*`/manager/config/mngsvrutil.properties`	Y	--	Y	Y	Y
		Client-side definition file of `mngsvrutil` command	*Installation-directory-of-Cosminexus*`/manager/config/mngsvrutilcl.properties`	Y	--	Y	Y	Y
		System log message mapping file for JP1/IM linkage	*Installation-directory-of-Cosminexus*`/manager/config/mserver.jp1event.system.mapping.properties`	Y	--	Y	Y	Y
			Installation-directory-of-Cosminexus`/manager/config/manager.jp1event.system.mapping.properties`	Y	--	Y	Y	Y
			Installation-directory-of-Cosminexus`/manager/config/manager.`*server-name*`.jp1event.system.mapping.properties`	Y	--	Y	Y	Y
	Performance, error analysis trace	PRF trace	*Directory-to-output-temporary-PRF-trace-file-(adminagent.prftrace_dir)*`/*.zip`	--	Y	--	A	D
	Internal interface trace	Integrated trace log	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*`/trace/mngtrace*.log`	Y	--	--	Y	Y

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
	Others	OS state information	*Installation-directory-of-Cosminexus*`/manager/tmp/*`	Y	--	--	Y	Y
Smart Composer	Definition information	Server settings property file	*Installation-directory-of-Cosminexus*`/manager/config/cmxserver.properties`	Y	--	Y	Y	Y
		Client settings property file	*User-home-directory-(user.home)*`/.cmxrc`	--	Y	Y	C	C
		Client common settings property file	*Installation-directory-of-Cosminexus*`/manager/config/cmxclient.properties`	Y	--	Y	Y	Y
		Load balancer definition property file	*Installation-directory-of-Cosminexus*`/manager/config/lb.properties`	Y	--	Y	Y	Y
Integrated user management	Definition information	JAAS configuration file	*Installation-directory-of-Cosminexus*`/manager/config/jaas.conf`	Y	--	Y	Y	Y
		Configuration file for integrated user management	*Installation-directory-of-Cosminexus*`/manager/config/ua.conf`	Y	--	Y	Y	Y
	API Trace	Trace of integrated user management	*Integrated-user-management-trace-file-name-(com.cosminexus.admin.auth.trace.prefix)*`.*.log`	--	Y	--	D	D
Virtual server manager in the 08-50 mode	Definition information	Virtual server manager property file	*Installation-directory-of-Cosminexus*`/manager/vmx/config/vmx.properties`	Y	--	Y	D	Y
		Client common settings property file of virtual server manager	*Installation-directory-of-Cosminexus*`/manager/vmx/config/vmxclient.properties`	Y	--	Y	D	Y
		Option definition file for VMware vCenter Server connection processing	*Installation-directory-of-Cosminexus*`/manager/vmx/config/vmx_avcs_usrconf.cfg`	Y	--	Y	D	Y
		Property file for VMware vCenter Server connection processing	*Installation-directory-of-Cosminexus*`/manager/vmx/config/vmx_avcs_usrconf.properties`	Y	--	Y	D	Y
		Common definition file for VMware vCenter Server connection processing	*Installation-directory-of-Cosminexus*`/manager/vmx/config/vmx_avcs_cjwconf.properties`	Y	--	Y	D	Y

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
	Other log	Log of the `cjclstartap` command used with the virtual server manager	*Manager-log-output-directory-(com.cosminexus.manager.log.dir)*`/vmx_avcs_logs/**`	--	Y	--	D	Y
		Log information collected from the virtual server	*Virtual-server-manager-log-directory-in-08-50-mode-(vmx.log.dir)*`/mngunit/*/*.zip`	--	Y	--	D	C
Virtual server manager	Definition information	Virtual server manager property file	*Installation-directory-of-Cosminexus*`/manager/vmi/config/vmi.properties`	Y	--	Y	D	Y
		Client settings property file of virtual server manager	*User-home-directory-(user.home)*`/.vmirc`	--	Y	Y	C	C
		Client common settings property file of virtual server manager	*Installation-directory-of-Cosminexus*`/manager/vmi/config/vmiclient.properties`	Y	--	Y	D	Y
		Load balancer connection settings property file	*Installation-directory-of-Cosminexus*`/manager/vmi/config/lb/*.properties`	Y	--	Y	D	Y
Server Communication Agent	Message log	Server Communication Agent log	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*`/sinaviagent*.log`	Y	--	--	D	Y
		Service log of Server Communication Agent	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*`/sinaviagentsv*.log`	Y	--	--	D	C
		`snactl` command log	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*`/snactl*.log`	Y	--	--	D	C
	Other logs	Error information during Server Communication Agent startup	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*`/sinaviagent.err`	Y	--	--	D	C
		Standard output of Server Communication Agent	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*`/sinaviagent.out`	Y	--	--	D	C
		Console output information of command processes started by Server	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*`/processConsole*.log`	Y	--	--	D	C

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		Communication Agent						
	Maintenance log	Maintenance log of Server Communication Agent	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*/maintenance/sinaviagent*.log	--	Y	--	D	C
		Maintenance log of snactl command	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*/maintenance/snactl*.log	--	Y	--	D	C
		Javalog of Server Communication Agent	*server-communication-agent-log-output-directory-(sinaviagent.log.dir)*/maintenance/sinaviagent.javalog*.log	--	Y	--	D	C
	Definition information	Option definition file for Server Communication Agent	*Installation-directory-of-Cosminexus*/sinagent/config/sinaviagent.cfg	Y	--	Y	D	Y
		Server Communication Agent property file	*Installation-directory-of-Cosminexus*/sinagent/config/sinaviagent.properties	Y	--	Y	D	Y
Cosminexus Web Services - Base	Message log	Message log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjmessage*.log	Y	--	--	A	C
			Web-container-server-log-output-directory-(web.server.log.directory)/cjweb_message*.log	Y	--	--	C	C
			EJB-client-log-output-directory-(ejbserver.client.log.directory)/*subdirectory-1-(ejbserver.client.ejb.log)*/*subdirectory-2-(ejbserver.client.log.appid)*/cjclmessage*.log	Y	--	--	C	C
		Server trace, client trace, default trace, application log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/WS/*.log	Y	--	--	A	C
			Web-container-server-log-output-directory-(web.server.log.directory)/WS/*.log	Y	--	--	C	Y

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		Client trace, default trace, provided command trace, application log	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*/*subdirectory-1-(ejbserver.client.ejb.log)*/*subdirectory-2-(ejbserver.client.log.appid)*/WS/*.log	Y	--	--	C	C
		JAXR Trace	JAXR-Trace-file-name-*(com.cosminexus.xml.registry.trace.file_path)**.log	Y	--	--	C	C
	Definition information	Common definition file	*Installation-directory-of-Cosminexus*/c4web/conf/c4webcom.cfg	Y	--	Y	Y	Y
		Server definition file	*Installation-directory-of-Cosminexus*/c4web/conf/c4websv.cfg	Y	--	Y	Y	Y
		Service deploy definition	*Installation-directory-of-Cosminexus*/CC/server/public/web/*server-name*/*context-root*/WEB-INF/server-config.xml	Y	--	Y	Y	Y
			Installation-directory-of-Cosminexus/CC/web/containers/*server-name*/webapps/*context-root*/WEB-INF/server-config.xml	Y	--	Y	Y	Y
		Client definition file	*Installation-directory-of-Cosminexus*/CC/server/public/web/*server-name*/*context-root*/WEB-INF/classes/c4webcl.properties	Y	--	Y	Y	Y
			Installation-directory-of-Cosminexus/CC/web/containers/*server-name*/webapps/*context-root*/WEB-INF/classes/c4webcl.properties	Y	--	Y	Y	Y
Cosminexus JAX-WS	Message log	Service and client operation log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CJW/cjwmessage*.log	Y	--	--	A	C
		Client operation log	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*/*subdirectory-1-(ejbserver.client.ejb.log)*/*subdirectory-2-(ejbserver.client.log.appid)*/CJW/cjwmessage*.log	Y	--	--	C	C

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		Service and client operation log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CJR/cjrmessage*.log	Y	--	--	A	C
	Other logs	Service and client communication log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CJW/cjwtransport*.log	Y	--	--	A	C
		Client communication log	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*/*subdirectory-1-(ejbserver.client.ejb.log)*/*subdirectory-2-(ejbserver.client.log.appid)*/CJW/cjwtransport*.log	Y	--	--	C	C
		Service and client communication log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CJR/cjrtransport*.log	Y	--	--	A	C
	Exception log	Service and client exception log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CJW/cjwexception*.log	Y	--	--	A	C
		Client exception log	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*/*subdirectory-1-(ejbserver.client.ejb.log)*/*subdirectory-2-(ejbserver.client.log.appid)*/CJW/cjwexception*.log	Y	--	--	C	C
		Service and client exception log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CJR/cjrexception*.log	Y	--	--	A	C
	Definition information	Action definition file	*Installation-directory-of-Cosminexus*/jaxws/conf/*.properties	Y	--	Y	Y	Y
		Process wise action definition file	*Process-wise-action-definition-file-(com.cosminexus.jaxws.confpath)*	Y	--	Y	D	D
		Action definition file	*Installation-directory-of-Cosminexus*/jaxrs/conf/*.properties	Y	--	Y	C	C
		Process-wise action definition file	*Process-wise-action-definition-file-(com.cosminexus.jaxrs.confpath)*	Y	--	Y	D	D
	Definition file	web.xml	*J2EE-server-working-directory-(ejb.public.directory)*/web/*server-name*/*/WEB-INF/web.xml	--	Y	Y	A	A

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
		cosminexus-jaxws.xml	*J2EE-server-working-directory-(ejb.public.directory)*/web/*server-name*/*/WEB-INF/cosminexus-jaxws.xml	--	Y	Y	A	A
		Handler chain setup file	*J2EE-server-working-directory-(ejb.public.directory)*/web/*server-name*/**	--	Y	Y	D	D
		WSDL	*J2EE-server-working-directory-(ejb.public.directory)*/web/*server-name*/*/WEB-INF/wsdl/*	--	Y	Y	A	A
		Catalog file	*J2EE-server-working-directory-(ejb.public.directory)*/web/*server-name*/*/WEB-INF/classes/META-INF/jax-ws-catalog.xml	--	Y	Y	A	A
			J2EE-server-working-directory-(ejb.public.directory)/ejb/*server-name*/apps/*/*/*/jax-ws-catalog.xml	--	Y	Y	A	A
Cosminexus JPA provider	Message log	Cosminexus JPA provider operation log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjpa/cjpaoperation*.log	Y	--	--	A	C
		PRF daemon and PRF command log	*PRF-spool-directory-(prfspool)*/log/PRF-identifier/ctmlog*	Y	--	--	A	C
		Resource adapter operation log deployed and used as J2EE resource adapter	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/connectors/*.log	Y	--	--	A	C
		Message log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjmessage*.log	Y	--	--	A	C
	Other logs	Exception information when an error occurs	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjexception*.log	Y	--	--	A	C
	Definition information	Option definition file for J2EE server	*Installation-directory-of-Cosminexus*/CC/server/usrconf/ejb/*server-name*/*.cfg	Y	--	Y	Y	Y
		User property file for the J2EE server	*Installation-directory-of-Cosminexus*/CC/server/	Y	--	Y	Y	Y

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
			usrconf/ejb/*server-name*/*.properties					
		Security policy file for J2EE server	*Cosminexus-installation-directory*/CC/server/usrconf/ejb/*server-name*/*.policy	Y	--	Y	Y	Y
	User data	EJB server working directory contents	*J2EE-server-working-directory-(ejb.public.directory)*/ejb/*server-name*/**	--	Y	--	D	D
		Web container working directory contents	*J2EE-server-working-directory-(ejb.public.directory)*/web/*server-name*/**	--	Y	--	D	D
Cosminexus JMS provider	Message log	CJMSP Broker log	*Installation-directory-of-Cosminexus*/CC/cjmsp/var/instances/*instance-name*/log/*.log	Y	--	--	C	C
		Management command log	*Installation-directory-of-Cosminexus*/CC/cjmsp/var/admin/log/*.log	Y	--	--	C	C
		CJMSP resource adapter log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/cjms/*resource-adapter-name*/*.log	Y	--	--	A	C
	Definition information	CJMSP Broker Common Property File	*Installation-directory-of-Cosminexus*/CC/cjmsp/lib/props/broker/commonconfig.properties	Y	--	Y	Y	Y
		CJMSP Broker Property File	*Installation-directory-of-Cosminexus*/CC/cjmsp/var/instances/*instance-name*/props/config.properties	Y	--	Y	C	C
		Management command definition file	*Installation-directory-of-Cosminexus*/CC/cjmsp/var/admin/config/admin.properties	Y	--	Y	C	C

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

A.3 Cosminexus Component Transaction Monitor

The following table describes directory paths to be collected in relation to a Cosminexus Component Container Monitor.

Table A–7: Collection target related to Cosminexus Component Transaction Monitor (In Windows)

Type of data	File to be collected	Default collection destination	Data			How to collect	
			Primary	Secondary	Definition	A	B
Message log	CTM daemon and CTM command log	*CTM-spool-directory-(ctmspool)*`\log\`*CTM-identifier*`\ctmlog*`	Y	--	--	A	C
	CTM domain manager log	*CTM-spool-directory-(ctmspool)*`\log\ctmdmlog*`	Y	--	--	A	C
Dump	Thread dump	*Installation-directory-of-Cosminexus*`\TPB\logj\javacore*`	Y	--	--	B	B
Definition information	CTM regulator configuration file	*CTM-regulator-configuration-file-(ctm.RegOption)*	Y	--	--	A	D
	OTM gateway configuration file	*OTM-gateway-configuration-file-(ctm.TSCGwOption)*	Y	--	--	A	D

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

Table A–8: Collection target related to Cosminexus Component Transaction Monitor (In UNIX)

Type of data	File to be collected	Collection target	Data			How to collect	
			Primary	Secondary	Definition	A	B
Message log	CTM daemon and CTM command log	*CTM-spool-directory-(ctmspool)*`/log/CTM-identifier/ctmlog*`	Y	--	--	A	C
	CTM domain manager log	*CTM-spool-directory-(ctmspool)*`/log/ctmdmlog*`	Y	--	--	A	C

Type of data	File to be collected	Collection target	Data			How to collect	
			Primary	Secondary	Definition	A	B
Dump	Thread dump	*Installation-directory-of-Cosminexus*`/TPB/logj/javacore*`	Y	--	--	B	B
	core dump	*Installation-directory-of-Cosminexus*`/TPB/logj/javacore*`	--	Y	--	B	B
Definition information	CTM regulator configuration file	*CTM-regulator-configuration-file-(ctm.RegOption)*	Y	--	--	A	D
	OTM gateway configuration file	*OTM-gateway-configuration-file-(ctm.TSCGwOption)*	Y	--	--	A	D

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

A.4 Cosminexus DABroker Library

The following table describes the collection targets related to Cosminexus DABroker Library.

Table A–9: Collection target related to Cosminexus DABroker Library (In Windows)

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
Definition information	Operation environment definition file	*Installation-directory-of-Cosminexus*`/DAB/conf/dasysconf`	Y	--	Y	B	B
	Connection destination database definition file	*Installation-directory-of-Cosminexus*`/DAB/conf/dadbenv`	Y	--	Y	B	B
Others	Spool information	*Installation-directory-of-Cosminexus*`/DAB/spool/*`	Y	--	--	B	B

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
		Installation-directory-of-Cosminexus/DAB/spool/*/*	--	Y	--	B	B

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the meaning of the directories and files to be collected that are described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

Table A–10: Collection target related to Cosminexus DABroker Library (In UNIX)

Type of data	File to be collected	Collection target	Data			Collection method	
			Primary	Secondary	Definition	A	B
Definition information	Operation environment definition file	*DABroker-operation-directory-(dabroker)*/conf/dasysconf	Y	--	Y	C	C
	Connection destination database definition file	*DABroker-operation-directory-(dabroker)*/conf/dadbenv	Y	--	Y	C	C
Others	Spool information	*DABroker-operation-directory-(dabroker)*/spool/*	Y	--	--	C	C
		DABroker-operation-directory-(dabroker)/spool/*/*	--	Y	--	C	C

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the meaning of the directories and files to be collected that are described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

A.5 Cosminexus Developer's Kit for Java

The following table describes the logs to be collected for Cosminexus Developer's Kit for Java.

Table A–11: Logs to be collected related to Cosminexus Developer's Kit for Java (In Windows)

Type of data	File to be collected	Default collection destination	Data			How to collect	
			Primary	Secondary	Definition	A	B
Dump	Crash dump	*Windows-Crash-dump-file-(CrashDumpFile)*	--	Y	--	C	C
		Windows-Crash-Dump-output-directory-(CrashDumpDir) / `*.dmp`	--	Y	--	C	C
Definition information	User-extended trace based performance analysis configuration file	*Configuration-file-for-user-extended-trace-based-performance-analysis-(jvm.userprf.File)*	Y	--	Y	C	C
	Explicit Memory Management functionality exclusion configuration file	*Name-of-Explicit-Memory-Management-functionality-exclusion-configuration-file-(jvm.exmemexcludeclass.File)*	Y	--	Y	C	C
	Explicit Memory Management functionality non-exclusion configuration file	*Name-of-Explicit-Memory-Management-functionality-non-exclusion-configuration-file-(jvm.exmemnotexcludeclass.File)*	Y	--	Y	C	C

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

Table A–12: Logs to be collected for Developer's Kit for Java (in UNIX)

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
Definition information	User-extended trace based performance analysis configuration file	*User-extended-trace-based-performance-analysis-configuration-file-(jvm.userprf.File)*	Y	--	Y	C	C
	Explicit Memory Management functionality exclusion configuration file	*Name-of-Explicit-Memory-Management-functionality-exclusion-configuration-file-(jvm.exmemexcludeclass.File)*	Y	--	Y	C	C
	Explicit Memory Management functionality non-exclusion configuration file	*Name-of-Explicit-Memory-Management-functionality-non-exclusion-configuration-file-(jvm.exmemnotexcludeclass.File)*	Y	--	Y	C	C

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data
  Y: Collected
  --: Not collected
- For the collection methods
  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

A.6 Cosminexus Performance Tracer

The following table describes the logs to be collected in relation to Cosminexus Performance Tracer.

Table A–13: Collection target related to Cosminexus Performance Tracer (In Windows)

Type of data	File to be collected	Default collection destination	Data			How to collect	
			Primary	Secondary	Definition	A	B
Message log	PRF daemon and PRF command log	*PRF-spool-directory-(prfspool)*`\log\`*PRF-identifier*`\ctmlog*`	Y	--	--	A	C

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

Table A–14: Collection target related to Cosminexus Performance Tracer (In UNIX)

Type of data	File to be collected	Collection target	Data			How to collect	
			Primary	Secondary	Definition	A	B
Message log	PRF daemon and PRF command log	*PRF-spool-directory-(prfspool)*`/log/`*PRF-identifier*`/ctmlog*`	Y	--	--	A	C

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

A.7 Cosminexus Web Services - Security

The following table describes the logs to be collected in relation to a Cosminexus Web Services - Security.

Table A–15: Collection target related to Cosminexus Web Services - Security (In Windows)

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
Cosminexus Web Services - Security	Message log	Command trace, client trace	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*`/`*subdirectory-1-(ejbserver.client.ejb.log)*`/`*subdirectory-2-(ejbserver.client.log.appid)*`/WS/*.log`	Y	--	--	C	C
		Server trace, client trace	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/WS/*.log`	Y	--	--	A	C

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
			Web-container-server-log-output-directory-(web.server.log.directory)/WS/*.log	Y	--	--	C	C
	Definition information	Environment setup file	*Installation-directory-of-Cosminexus*/wss/conf/cwsscfg.properties	Y	--	Y	Y	Y
		Client deploy definition	*Installation-directory-of-Cosminexus*/CC/server/public/web/*server-name*/*context-root*/WEB-INF/classes/client-config.xml	Y	--	Y	Y	Y
			Installation-directory-of-Cosminexus/CC/web/containers/*server-name*/webapps/*context-root*/WEB-INF/classes/client-config.xml	Y	--	Y	Y	Y
		Policy definition file	*Cosminexus-installation-directory*/CC/server/public/web/*server-name*/*context-root*/WEB-INF/classes/policy-config.xml	Y	--	Y	Y	Y
			Installation-directory-of-Cosminexus/CC/web/containers/*server-name*/webapps/*context-root*/WEB-INF/classes/policy-config.xml	Y	--	Y	Y	Y
		Function definition file	*Installation-directory-of-Cosminexus*/CC/server/public/web/*server-name*/*context-root*/WEB-INF/classes/security-config.xml	Y	--	Y	Y	Y
			Installation-directory-of-Cosminexus/CC/web/containers/*server-name*/webapps/*context-root*/WEB-INF/classes/security-config.xml	Y	--	Y	Y	Y
Cosminexus XML Security - Core	Maintenance log	Trace maintenance information such as method invocation	*XML-Security-Core-Trace-output-directory-(com.cosminexus.xml.security.logging.trace_dir)*/*.log	Y	--	--	C	C

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

Table A–16: Collection target related to Cosminexus Web Services - Security (In UNIX)

Category	Type of data	File to be collected	Collection Target	Data			How to collect	
				Primary	Secondary	Definition	A	B
Cosminexus Web Services - Security	Message log	Command trace, client trace	*EJB-client-log-output-directory-(ejbserver.client.log.directory)*`/`*subdirectory-1-(ejbserver.client.ejb.log)*`/`*subdirectory-2-(ejbserver.client.log.appid)*`/WS/*.log`	Y	--	--	C	C
		Server trace, client trace	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/WS/*.log`	Y	--	--	A	C
			Web-container-server-log-output-directory-(web.server.log.directory)`/WS/*.log`	Y	--	--	C	C
	Definition information	Environment setup file	*Installation-directory-of-Cosminexus*`/wss/conf/cwsscfg.properties`	Y	--	Y	Y	Y
		Client deploy definition	*Installation-directory-of-Cosminexus*`/CC/server/public/web/`*server-name*`/`*context-root*`/WEB-INF/classes/client-config.xml`	Y	--	Y	Y	Y
			Installation-directory-of-Cosminexus`/CC/web/containers/`*server-name*`/webapps/`*context-root*`/WEB-INF/classes/client-config.xml`	Y	--	Y	Y	Y
		Policy definition file	*Installation-directory-of-Cosminexus*`/CC/server/public/web/`*server-name*`/`*context-root*`/WEB-INF/classes/policy-config.xml`	Y	--	Y	Y	Y
			Installation-directory-of-Cosminexus`/CC/web/containers/`*server-name*`/`	Y	--	Y	Y	Y

Category	Type of data	File to be collected	Collection Target	Data			How to collect	
				Primary	Secondary	Definition	A	B
			webapps/*context-root*/WEB-INF/classes/policy-config.xml					
		Function definition file	*Installation-directory*/CC/server/public/web/*server-name*/*context-root*/WEB-INF/classes/security-config.xml	Y	--	Y	Y	Y
			Installation-directory-of-Cosminexus/CC/web/containers/*server-name*/webapps/*context-root*/WEB-INF/classes/security-config.xml	Y	--	Y	Y	Y
Cosminexus XML Security - Core	Maintenance log	Trace maintenance information such as method invocation	*XML-Security-Core-Trace-output-directory-(com.cosminexus.xml.security.logging.trace_dir)*/*.log	Y	--	--	C	C

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

A.8 Cosminexus HTTP Server

The following table describes the logs to be collected in relation to Cosminexus HTTP Server.

Table A–17: Collection target related to Cosminexus HTTP Server (In Windows)

Type of data	File to be collected	Default collection destination	Data			How to collect	
			Primary	Secondary	Definition	A	B
Message log	Error log	*HWS-error-log-directory-(HttpsdErrorLogFileDir)*/error*	Y	--	--	A	C
	Log set (for previous version compatibility)	*Installation-directory-of-HWS*/logs/*	Y	--	--	Y	Y

Type of data	File to be collected	Default collection destination	Data			How to collect	
			Primary	Secondary	Definition	A	B
Other logs	Request log	*HWS-request-log-directory-(HttpsdRequestLogFileDir)*/hwsrequest*	--	Y	--	A	C
	Process ID file	*HWS-process-ID-file-(PidFile)*	Y	--	--	C	C
Process log	Access log	*HWS-access-log-directory-(HttpsdCustomLogFileDir)*/access*	--	Y	--	A	C
Definition information	Definition file	*Installation-directory-of-HWS*/servers/HWS_*server-name*/conf/*.conf	Y	--	Y	Y	Y
	Definition file set (for previous compatibility)	*Installation-directory-of-HWS*/conf/*.conf	Y	--	Y	Y	Y
Interface trace	Internal trace	*HWS-internal-trace-directory-(HttpsdTraceLogFileDir)*/hws.trc*	--	Y	--	A	C
WebSocket log	WebSocket log	*HWSWebSocket-log-directory-(HttpsdWebSocketLogFileDir)*/hws_websocket_log*	--	Y	--	A	C

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

Table A–18: Collection target related to Cosminexus HTTP Server (In UNIX)

Type of data	File to be collected	Collection target	Data			How to collect	
			Primary	Secondary	Definition	A	B
Message log	Error log	*HWS-error-log-directory-(HttpsdErrorLogFileDir)*/error*	Y	--	--	A	C
	Log set (for previous version compatibility)	*Installation-directory-of-HWS*/logs/*	Y	--	--	Y	Y
Other logs	Request log	*HWS-request-log-directory-(HttpsdRequestLogFileDir)*/hwsrequest*	--	Y	--	A	C
	Process ID file	*HWS-process-ID-file-(PidFile)*	Y	--	--	C	C

Type of data	File to be collected	Collection target	Data			How to collect	
			Primary	Secondary	Definition	A	B
Access log	Access log	*HWS-access-log-directory-(HttpsdCustomLogFileDir)*`/access*`	--	Y	--	A	C
Dump	core dump	*HWS-core-dump-output-directory-(CoreDumpDirectory)*`/core*`	--	Y	--	A	C
	gcache server core dump	*Work-directory-of-HWS-cache-server-(SSLCacheServerRunDir)*`/core*`	--	Y	--	C	C
Definition information	Definition file	*Installation-directory-of-HWS*`/servers/HWS_`*server-name*`/conf/*.conf`	Y	--	Y	Y	Y
	Definition file set (for previous version compatibility)	*Installation-directory-of-HWS*`/conf/*.conf`	Y	--	Y	Y	Y
Interface trace	Internal trace	*HWS-internal-trace-directory-(HttpsdTraceLogFileDir)*`/hws.trc*`	--	Y	--	A	C
WebSocket log	WebSocket log	*HWSWebSocket-log-directory-(HttpsdWebSocketLogFileDir)*`/hws_websocket_log*`	--	Y	--	A	C

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

A.9 Microsoft Internet Information Service

The following table describes the logs to be collected related to the information of Microsoft Internet Information Service.

Table A–19: Collection target related to information of Microsoft Internet Information Service (In Windows)

Type of data	File to be collected	Default collection destination	Data			How to collect	
			Primary	Secondary	Definition	A	B
Message log	Access log	*IIS-access-log-directory-(IIS_log_dir)*`/*`	--	Y	--	C	C

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

A.10 HCSC server

The following table describes the logs to be collected for the HCSC server information.

Table A–20: Logs to be collected for the HCSC server information (in Windows)

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
Message log	HCSC-Manager log	*HCSC-log-output-directory-(cscmng.log.dir)*`/*`	Y	--	--	C	C
	Log of user-authentication information management commands	*output-destination-directory-of-message-log-files-of-user-authentication-information-management-commands-(authinfo.command.messagelog.filepath)*`/*`	Y	--	--	C	C
Access log	Request trace	*HCSC-server-property-(requesttrace-filepath)*`/*`	Y	--	--	C	C
Maintenance log	Method trace	*HCSC-server-property-(methodtrace-filepath)*`/*`	Y	--	--	C	C
Definition information	HCSC server definition information	*Application-Server-installation-directory*`/CSC/system/msg/*`	Y	--	Y	Y	Y
	HCSC-Manager definition file	*Application-Server-installation-directory*`/CSC/config/manager/*`	Y	--	Y	Y	Y
	HCSC-Messaging definition file	*Application-Server-installation-directory*`/CSC/config/msg/*`	Y	--	Y	Y	Y
	Repository	*HCSC-repository-root-(cscmng.repository.root)*`/**`	Y	--	--	Y	Y
Other logs	Business process activity trace	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/csc/*`	--	Y	--	A	C

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For the collection methods

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

Table A–21: Logs to be collected for the HCSC server information (in UNIX)

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
Message log	HCSC-Manager log	*HCSC-log-output-directory-(cscmng.log.dir)*`/*`	Y	--	--	C	C
Access log	Request trace	*HCSC-server-property-(requesttrace-filepath)*`/*`	Y	--	--	C	C
Maintenance log	Method trace	*HCSC-server-property-(methodtrace-filepath)*`/*`	Y	--	--	C	C
Definition information	HCSC server definition information	*Application-Server-installation-directory*`/CSC/system/msg/*`	Y	--	Y	Y	Y
	HCSC-Manager definition file	*Application-Server-installation-directory*`/CSC/config/manager/*`	Y	--	Y	Y	Y
	HCSC-Messaging definition file	*Application-Server-installation-directory*`/CSC/config/msg/*`	Y	--	Y	Y	Y
	Repository	*HCSC-repository-root-(cscmng.repository.root)*`/**`	Y	--	--	Y	Y
Other logs	Business process activity trace	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/csc/*`	--	Y	--	A	C

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For the collection methods

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

A.11 HCSC server (FTP receipt)

The following table describes the logs to be collected for the HCSC server (FTP receipt) information.

Table A–22: Logs to be collected for the HCSC server (FTP receipt) information (in Windows)

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
Definition information	FTP receipt definition information	*Application-Server-installation-directory*`/CSC/config/ftprecp/*`	Y	--	Y	Y	Y
	FTP receipt common definition information	*Application-Server-installation-directory*`/CSC/config/ftprecp/common/*`	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For the collection methods

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

Table A–23: Logs to be collected for the HCSC server (FTP receipt) information (in UNIX)

Type of data	File to be collected	Collection target	Data			Collection method	
			Primary	Secondary	Definition	A	B
Definition information	FTP receipt definition information	*Application-Server-installation-directory*`/CSC/config/ftprecp/*`	Y	--	Y	Y	Y
	FTP receipt common definition information	*Application-Server-installation-directory*`/CSC/config/ftprecp/common/*`	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data
A: `mngsvrutil collect snapshot` command
B: `snapshotlog` command

- For data
  Y: Collected
  --: Not collected
- For the collection methods
  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

A.12 HCSC server (TP1 adapter)

The following table describes the logs to be collected related to HCSC server (TP1 adapter) information.

Table A–24: Collection target related to the information of HCSC server (TP1 adapter) (In Window)

Type of data	File to be collected	Default collection destination	Data			How to collect	
			Primary	Secondary	Definition	A	B
Message log	Trace set	*J2EE-server-working-directory-(ejb.public.directory)*`/ejb/`*server-name*`/logs/CSCADP/TP1ADP/maintenance/*/*`	--	Y	--	A	A
Definition information	Definition information for the TP1 adapter	*Installation-directory-of-Cosminexus*`/CSC/custom-adapter/TP1/config/*`	Y	--	Y	Y	Y

Legend:
Primary: Primary delivery data
Secondary: Secondary delivery data
Definition: Definition sending data
A: `mngsvrutil collect snapshot` command
B: `snapshotlog` command

- For data
  Y: Collected
  --: Not collected
- For methods of collection
  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

Table A–25: Collection target related to the information of HCSC server (TP1 adapter) (In UNIX)

Type of data	File to be collected	Collection target	Data			How to collect	
			Primary	Secondary	Definition	A	B
Message log	Trace set	*J2EE-server-working-directory-(ejb.public.directory)*`/ejb/`*server-*	--	Y	--	A	A

Type of data	File to be collected	Collection target	Data			How to collect	
			Primary	Secondary	Definition	A	B
		name/logs/CSCADP/TP1ADP/maintenance/*/*					
Definition information	Definition information for the TP1 adapter	*Installation-directory-of-Cosminexus*/CSC/custom-adapter/TP1/config/*	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

A.13 HCSC server (File adapter)

The following table describes the logs to be collected related to the information of HCSC server (file adapter).

Table A–26: Collection target related to the information of HCSC server (file adapter) (In Windows)

Type of data	File to be collected	Default Collection destination	Data			How to collect	
			Primary	Secondary	Definition	A	B
Message log	Trace set	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CSCADP/FFADP/maintenance/*/*	--	Y	--	A	A
Definition information	Definition information for the file adapter	*Installation-directory-of-Cosminexus*/CSC/custom-adapter/File/config/*	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

Table A–27: Collection target related to the information of HCSC server (file adapter) (In UNIX)

Type of data	File to be collected	Collection target	Data			How to collect	
			Primary	Secondary	Definition	A	B
Message log	Trace set	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CSCADP/FFADP/maintenance/*/*	--	Y	--	A	A
Definition information	Definition information for the file adapter	*Installation-directory-of-Cosminexus*/CSC/custom-adapter/File/config/*	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

A.14 HCSC server (Object Access adapter)

The following table describes the logs to be collected related to the information of HCSC server (Object Access adapter).

Table A–28: Collection target related to the information of HCSC server (Object Access adapter) (In Windows)

Type of data	File to be collected	Default collection destination	Data			How to collect	
			Primary	Secondary	Definition	A	B
Message log	Trace Set	*J2EE-server-working-directory-(ejb.public.directory)*/ejb/*server-name*/logs/CSCADP/OAADP/maintenance/*/*	--	Y	--	A	A
Definition information	Definition information for the OA adapter	*Installation-directory-of-Cosminexus*/CSC/custom-adapter/OA/config/*	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

Table A–29: Collection target related to the information of HCSC server (Object Access adapter) (In UNIX)

Type of data	File to be collected	Collection target	Data			How to collect	
			Primary	Secondary	Definition	A	B
Message log	Trace Set	*J2EE-server-working-directory-(ejb.public.directory)*`/ejb/`*server-name*`/logs/CSCADP/OAADP/maintenance/*/*`	--	Y	--	A	A
Definition information	Definition information for the OA adapter	*Installation-directory-of-Cosminexus*`/CSC/custom-adapter/OA/config/*`	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

A.15 HCSC server (Message Queue adapter)

The following table describes the logs to be collected related to the functions of HCSC server (Message Queue adapter).

Table A–30: Collection target related to the information of HCSC server (Message Queue adapter) (In Windows)

Type of data	File to be collected	Default collection destination	Data			How to collect	
			Primary	Secondary	Definition	A	B
Message log	Trace Set	*J2EE-server-working-directory-(ejb.public.directory)*`/ejb/`*server-*	--	Y	--	A	A

Type of data	File to be collected	Default collection destination	Data			How to collect	
			Primary	Secondary	Definition	A	B
		name/logs/CSCADP/MQADP/maintenance/*/*					
Definition information	Definition information for the MQ adapter	*Installation-directory-of-Cosminexus*/CSC/custom-adapter/MQ/config/*	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

Table A–31: Collection target related to the information of HCSC server (Message Queue adapter) (In UNIX)

Type of data	File to be collected	Default collection destination	Data			How to collect	
			Primary	Secondary	Definition	A	B
Message log	Trace set	*J2EE-server-working-directory-(ejb.public.directory)*/ejb/*server-name*/logs/CSCADP/MQADP/maintenance/*/*	--	Y	--	A	A
Definition information	Definition information for the MQ adapter	*Installation-directory-of-Cosminexus*/CSC/custom-adapter/MQ/config/*	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

A.16 HCSC server (FTP adapter)

The following table describes the logs to be collected related to HCSC server (FTP adapter) information.

Table A–32: Collection target related to the information of HCSC server (FTP adapter) (In Window)

Type of data	File to be collected	Default collection destination	Data			How to collect	
			Primary	Secondary	Definition	A	B
Communication trace	FTP adapter protocol trace	*J2EE-server-working-directory-(ejb.public.directory)*`/ejb/`*server-name*`/logs/CSCADP/FTPADP/*/*`	--	Y	--	A	C
Maintenance log	FTP adapter maintenance log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/CSCADP/FTPADP/maintenance/*/*`	--	Y	--	A	C
Message log	Trace set	*J2EE-server-working-directory-(ejb.public.directory)*`/ejb/`*server-name*`/logs/CSCADP/TP1ADP/maintenance/*/*`	--	Y	--	C	C
Definition information	Definition information for the FTP adapter	*Installation-directory-of-Cosminexus*`/CSC/custom-adapter/FTP/config/*`	Y	--	Y	Y	Y
	Common definition information for the FTP adapter	*Installation-directory-of-Cosminexus*`/CSC/custom-adapter/FTP/config/common/*`	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

Table A–33: Collection target related to the information of HCSC server (FTP adapter) (In UNIX)

Type of data	File to be collected	Collection target	Data			How to collect	
			Primary	Secondary	Definition	A	B
Communication trace	FTP adapter protocol trace	*J2EE-server-working-directory-(ejb.public.directory)*`/ejb/`*server-name*`/logs/CSCADP/FTPADP/*/*`	--	Y	--	A	C
Maintenance log	FTP adapter maintenance log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/CSCADP/FTPADP/maintenance/*/*`	--	Y	--	A	C

Type of data	File to be collected	Collection target	Data			How to collect	
			Primary	Secondary	Definition	A	B
Message log	Trace set	*J2EE-server-working-directory-(ejb.public.directory)*/ejb/*server-name*/logs/CSCADP/TP1ADP/maintenance/*/*	--	Y	--	C	C
Definition information	Definition information for the FTP adapter	*Installation-directory-of-Cosminexus*/CSC/custom-adapter/FTP/config/*	Y	--	Y	Y	Y
	Common definition information for the FTP adapter	*Installation-directory-of-Cosminexus*/CSC/custom-adapter/FTP/config/common/*	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

A.17 HCSC server (SFTP adapter)

The following table describes the logs to be collected related to HCSC server (SFTP adapter) information.

Table A–34: Collection target related to the information of HCSC server (SFTP adapter) (In Window)

Type of data	File to be collected	Collection target	Data			How to collect	
			Primary	Secondary	Definition	A	B
Communication trace	SFTP adapter protocol trace	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CSCADP/SFTPADP/*/*	--	Y	--	A	A
Maintenance log	SFTP adapter maintenance log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CSCADP/SFTPADP/maintenance/*/*	--	Y	--	A	A
Definition information	Definition information for the SFTP adapter	*Installation-directory-of-Cosminexus*/CSC/custom-adapter/SFTP/config/*	Y	--	Y	Y	Y
	Common definition information for the SFTP adapter	*Installation-directory-of-Cosminexus*/CSC/custom-adapter/SFTP/config/common/*	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

Table A–35: Collection target related to the information of HCSC server (SFTP adapter) (In UNIX)

Type of data	File to be collected	Collection target	Data			How to collect	
			Primary	Secondary	Definition	A	B
Communication trace	SFTP adapter protocol trace	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/`CSCADP/SFTPADP/*/*`	--	Y	--	A	A
Maintenance log	SFTP adapter maintenance log	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/`CSCADP/SFTPADP/maintenance/*/*`	--	Y	--	A	A
Definition information	Definition information for the SFTP adapter	*Installation-directory-of-Cosminexus*`/CSC/custom-adapter/SFTP/config/*`	Y	--	Y	Y	Y
	Common definition information for the SFTP adapter	*Installation-directory-of-Cosminexus*`/CSC/custom-adapter/SFTP/config/common/*`	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

A.18 HCSC server (file operation adapter)

The following table describes the logs to be collected for the HCSC server (file operation adapter) information.

Table A–36: Logs to be collected for the HCSC server (file operation adapter) information (in Windows)

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
Maintenance log	Maintenance log for the file operation adapter	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/CSCADP/ADPFOP/maintenance/*/cscadpfopmnt_*.log`	Y	--	--	A	C
Exception log	Exception log for the file operation adapter	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/CSCADP/ADPFOP/maintenance/*/cscadpfopexp_*.log`	Y	--	--	A	C
Definition information	Definition information for the file operation adapter	*Application-Server-installation-directory*`/CSC/config/adpfop/*`	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For the collection methods

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

Table A–37: Logs to be collected for the HCSC server (file operation adapter) information (in UNIX)

Type of data	File to be collected	Collection target	Data			Collection method	
			Primary	Secondary	Definition	A	B
Maintenance log	Maintenance log for the file operation adapter	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/CSCADP/ADPFOP/maintenance/*/cscadpfopmnt_*.log`	Y	--	--	A	C
Exception log	Exception log for the file operation adapter	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/CSCADP/ADPFOP/maintenance/*/cscadpfopexp_*.log`	Y	--	--	A	C
Definition information	Definition information for the file operation adapter	*Application-Server-installation-directory*`/CSC/config/adpfop/*`	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data
Definition: Definition sending data
A: `mngsvrutil collect snapshot` command
B: `snapshotlog` command

- For data
  Y: Collected
  --: Not collected
- For the collection methods
  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

A.19 HCSC server (FTP inbound adapter)

The following table describes the logs to be collected for the HCSC server (FTP inbound adapter) information.

Table A–38: Logs to be collected for the HCSC server (FTP inbound adapter) information (in Windows)

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
Message log	Message log for the FTP inbound adapter	*J2EE-server-working-directory-(ejb.public.directory)*`/ejb/`*server-name*`/logs/csc/inbound-adapter/ftp/`*resource-adapter-name*`/*`	Y	--	--	A	A
	Message log for the operation commands of the FTP inbound adapter	*Application-Server-installation-directory*`/CSC/inbound-adapter/ftp/logs/`*resource-adapter-name*`/*`	Y	--	--	Y	Y
Definition information	FTP inbound adapter definition file	*Application-Server-installation-directory*`/CSC/inbound-adapter/ftp/config/`*resource-adapter-name*`/`*server-name*`/*.xml`	Y	--	Y	Y	Y

Legend:
Primary: Primary delivery data
Secondary: Secondary delivery data
Definition: Definition sending data
A: `mngsvrutil collect snapshot` command
B: `snapshotlog` command

- For data
  Y: Collected
  --: Not collected
- For the collection methods
  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

Table A–39: Logs to be collected for the HCSC server (FTP inbound adapter) information (in UNIX)

Type of data	File to be collected	Collection target	Data			Collection method	
			Primary	Secondary	Definition	A	B
Message log	Message log for the FTP inbound adapter	*J2EE-server-working-directory-(ejb.public.directory)*`/ejb/`*server-name*`/logs/csc/inbound-adapter/ftp/`*resource-adapter-name*`/*`	Y	--	--	A	A
	Message log for the operation commands of the FTP inbound adapter	*Application-Server-installation-directory*`/CSC/inbound-adapter/ftp/logs/`*resource-adapter-name*`/*`	Y	--	--	Y	Y
Definition information	FTP inbound adapter definition file	*Application-Server-installation-directory*`/CSC/inbound-adapter/ftp/config/`*resource-adapter-name*`/*.xml`	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For the collection methods

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

A.20 HCSC server (mail adapter)

The following table describes the logs to be collected for the HCSC server (mail adapter) information.

Table A–40: Logs to be collected for the HCSC server (mail adapter) information (in Windows)

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
Message log	Message log for the mail adapter	*Message-log-output-directory-for-the-HCSC-mail-adapter-operation-commands-*	Y	--	--	C	C

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
	operation commands	*(mailadp.command.messagelog.filepath)*`/*`					
Maintenance log	Maintenance log for the mail adapter	*Maintenance-log-output-directory-for-the-HCSC-mail-adapter-(mailadp.methodtrace.filepath)*`/*/*`	--	Y	--	A	C
Definition information	Definition information for the mail adapter	*Application-Server-installation-directory*`/CSC/config/mail/*`	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For the collection methods

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

Table A–41: Logs to be collected for the HCSC server (mail adapter) information (in UNIX)

Type of data	File to be collected	Collection target	Data			Collection method	
			Primary	Secondary	Definition	A	B
Message log	Message log for the mail adapter operation commands	*Message-log-output-directory-for-the-HCSC-mail-adapter-operation-commands-(mailadp.command.messagelog.filepath)*`/*`	Y	--	--	C	C
Maintenance log	Maintenance log for the mail adapter	*Maintenance-log-output-directory-for-the-HCSC-mail-adapter-(mailadp.methodtrace.filepath)*`/*/*`	--	Y	--	A	C
Definition information	Definition information for the mail adapter	*Application-Server-installation-directory*`/CSC/config/mail/*`	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

--: Not collected

- For the collection methods
  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

A.21 HCSC server (HTTP adapter)

The following table describes the logs to be collected for the HCSC server (HTTP adapter) information.

Table A–42: Logs to be collected for the HCSC server (HTTP adapter) information (in Windows)

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
Maintenance log	Maintenance log for the HTTP adapter	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/CSCADP/ADPHTTP/maintenance/*/cscadphttpmnt_*.log`	--	Y	--	A	C
Exception log	Exception log for the HTTP adapter	*J2EE-server-log-output-directory-(ejb.server.log.directory)*`/CSCADP/ADPHTTP/maintenance/*/cscadphttpexp_*.log`	Y	--	--	A	C
Definition information	Definition information for the HTTP adapter	*Application-Server-installation-directory*`/CSC/custom-adapter/HTTP/config/**`	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data
  Y: Collected
  --: Not collected
- For the collection methods
  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

Table A–43: Logs to be collected for the HCSC server (HTTP adapter) information (in UNIX)

Type of data	File to be collected	Collection target	Data			Collection method	
			Primary	Secondary	Definition	A	B
Maintenance log	Maintenance log for the HTTP adapter	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CSCADP/ADPHTTP/maintenance/*/cscadphttpmnt_*.log	--	Y	--	A	C
Exception log	Exception log for the HTTP adapter	*J2EE-server-log-output-directory-(ejb.server.log.directory)*/CSCADP/ADPHTTP/maintenance/*/cscadphttpexp_*.log	Y	--	--	A	C
Definition information	Definition information for the HTTP adapter	*Application-Server-installation-directory*/CSC/custom-adapter/HTTP/config/**	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For the collection methods

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

A.22 HCSC server (command adapter)

The following table describes the logs to be collected for the information about the HCSC server (command adapter).

Table A–44: Logs to be collected for the information about the HCSC server (command adapter) (in Windows)

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
Maintenance log	Maintenance log for the command adapter	*J2EE-server-working-directory-(ejb.public.directory)*/ejb/*server-name*/logs/CSCADP/ADPCMD/maintenance/*/cscadpcmdmnt_*.log	--	Y	--	A	A

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
Exception log	Exception log for the command adapter	*J2EE-server-working-directory-(ejb.public.directory)*`/ejb/`*server-name*`/logs/CSCADP/ADPCMD/maintenance/*/cscadpcmdexp_*.log`	Y	--	--	A	A
Definition information	Definition information for the command adapter	*Application-Server-installation-directory*`/CSC/custom-adapter/Command/config/*`	Y	--	Y	Y	Y
		Application-Server-installation-directory`/CSC/custom-adapter/Command/config/common/*`	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For the collection methods

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

Table A–45: Logs to be collected for the information about the HCSC server (command adapter) (in UNIX)

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
Maintenance log	Maintenance log for the command adapter	*J2EE-server-working-directory-(ejb.public.directory)*`/ejb/`*server-name*`/logs/CSCADP/ADPCMD/maintenance/*/cscadpcmdmnt_*.log`	--	Y	--	A	A
Exception log	Exception log for the command adapter	*J2EE-server-working-directory-(ejb.public.directory)*`/ejb/`*server-name*`/logs/CSCADP/ADPCMD/maintenance/*/cscadpcmdexp_*.log`	Y	--	--	A	A
Definition information	Definition information for the command adapter	*Application-Server-installation-directory*`/CSC/custom-adapter/Command/config/*`	Y	--	Y	Y	Y

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
		Application-Server-installation-directory/CSC/custom-adapter/Command/config/common/*	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For the collection methods

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

A.23 HCSC server (file event reception)

The following table describes the logs to be collected for the information about the HCSC server (file event reception).

Table A–46: Logs to be collected for the information about the HCSC server (file event reception) (in Windows)

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
Message log	File event trace	*J2EE-server-working-directory-(ejb.public.directory)*/ejb/*server-name*/logs/csc/rcp/fileevent/*/fileeventtrace_*.log	--	Y	--	A	A
Definition information	Definition information for file event reception	*Application-Server-installation-directory*/CSC/custom-reception/fileevent/config/*	Y	--	Y	Y	Y
		Application-Server-installation-directory/CSC/custom-reception/fileevent/config/common/*	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For the collection methods

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

Table A–47: Logs to be collected for the information about the HCSC server (file event reception) (in UNIX)

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
Message log	File event trace	*J2EE-server-working-directory-(ejb.public.directory)*`/ejb/`*server-name*`/logs/csc/rcp/fileevent/*/fileeventtrace_*.log`	--	Y	--	A	A
Definition information	Definition information for file event reception	*Application-Server-installation-directory*`/CSC/custom-reception/fileevent/config/*`	Y	--	Y	Y	Y
		Application-Server-installation-directory`/CSC/custom-reception/fileevent/config/common/*`	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For the collection methods

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

A.24 Audit log

The following table describes the logs to be collected in relation to audit log information.

Table A–48: Collection target related to audit log information (In Windows)

Category	Type of data	File to be collected	Default collection destination	Data			How to collect	
				Primary	Secondary	Definition	A	B
None	Message log	Message log audit log	*Audit-log-message-output-directory-(auditlog.raslog.message.directory)*`/rasmessage*.log`	Y	--	--	C	C
	Other logs	Audit log	*Audit-log-output-directory-(auditlog.directory)*`/*`	--	Y	--	C	C
		Audit log exception information	*Audit-log-exception-output-directory-(auditlog.raslog.exception.directory)*`/rasexception*.log`	Y	--	--	C	C
	Maintenance log	Other logs	*Audit-log-message-output-directory-(auditlog.raslog.message.directory)*`/*/*`	Y	--	--	C	C
			Audit-log-exception-output-directory-(auditlog.raslog.exception.directory)`/*/*`	Y	--	--	C	C
	Definition information	Audit log definition file	*Cosminexus-installation-directory*`/common/conf/auditlog.properties`	Y	--	Y	C	C
Naming service	Others	Log showing naming service start, stop or abnormal termination	*Windows-event-log-(EventLog)*	Y	--	--	D	D
	Dump	Thread dump	*Cosminexus-installation-directory*`/TPB/logj/javacore*`	Y	--	--	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

Table A–49: Collection target related to audit log information (In UNIX)

Category	Type of data	File to be collected	Collection target	Data			How to collect	
				Primary	Secondary	Definition	A	B
None	Message log	Message log of audit log	*Audit-log-message-output-directory-(auditlog.raslog.message.directory)*`/rasmessage*.log`	Y	--	--	C	C
	Other logs	Audit log	*Audit-log-output-directory-(auditlog.directory)*`/*`	--	Y	--	C	C
		Audit log exception information	*Audit-log-exception-output-directory-(auditlog.raslog.exception.directory)*`/rasexception*.log`	Y	--	--	C	C
	Maintenance log	Other logs	*Audit-log-message-output-directory-(auditlog.raslog.message.directory)*`/*/*`	Y	--	--	C	C
			Audit-log-exception-output-directory-(auditlog.raslog.exception.directory)`/*/*`	Y	--	--	C	C
	Definition information	Audit log definition file	*Cosminexus-installation-directory*`/common/conf/auditlog.properties`	Y	--	Y	C	C
Naming service	Others	Log showing naming service start, stop or abnormal termination	*UNIX-syslog-(syslog)*	Y	--	--	C	C
	Dump	Thread dump	*Cosminexus-installation-directory*`/TPB/logj/javacore*`	Y	--	--	Y	Y
		core dump	*Cosminexus-installation-directory*`/TPB/logj/core*`	--	Y	--	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1 (4) Rules for the coding to be collected*.

A.25 Other information

The following table describes the logs to be collected in relation to other information.

Table A–50: Logs to be collected for the other information (in Windows)

Type of data	File to be collected	Default collection destination	Data			Collection method	
			Primary	Secondary	Definition	A	B
Message log	`cosmienv` command log	*Application-Server-installation-directory*`/env/log/*`	Y	--	--	Y	Y
Definition information	Host definition file (for Windows)	*System-root-directory-(systemroot)*`/system32/drivers/etc/hosts`	Y	--	Y	Y	Y
Other	Program product information	`/etc/.hitachi/pplistd/pplistd`	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For the collection methods

  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details on the collection method A and collection method B, see *Appendix A.1(2) Method of collecting snapshot log*. For the meanings of the directories and files to be collected and described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

Table A–51: Collection target related to other information (In UNIX)

Type of data	File to be collected	Default collection destination	Data			How to collect	
			Primary	Secondary	Definition	A	B
Message log	`cosmienv` command log	*Cosminexus-installation-directory*`/env/log/*`	Y	--	--	Y	Y
Definition information	Host Definition log (for UNIX)	`/etc/hosts`	Y	--	Y	Y	Y
Others	Program product information	`/etc/.hitachi/pplistd/pplistd`	Y	--	Y	Y	Y

Legend:

Primary: Primary delivery data

Secondary: Secondary delivery data

Definition: Definition sending data

A: `mngsvrutil collect snapshot` command

B: `snapshotlog` command

- For data

  Y: Collected

  --: Not collected

- For methods of collection
  See *Appendix A.1(3) Availability of snapshot log collection and changes in settings related to collection*.

Note: For details about collection method A and collection method B, see *Appendix A.1 (2) Method of collecting snapshot log*. For details about the file significance and directory to be collected described in the table, see *Appendix A.1(4) Rules for the coding to be collected*.

B. Identifying the Connection in Which an Error Has Occurred When Connecting to a Database

To handle a database-related error, it is important to identify the connection used to connect to the database.

This section describes how to identify a connection in which trouble occurred while connecting to the database (HiRDB or Oracle) using the information output from the applications server (trace based performance analysis and logs) and information output from the database (trace information and execution results of the `pdls` command, and contents of dynamic performance view).

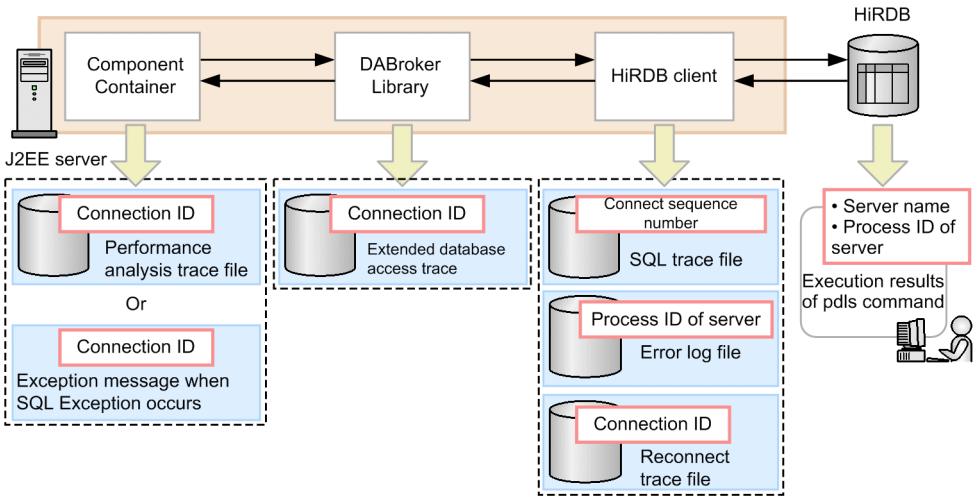
With the application server, the connection information (connection ID) for unique identification of the connection you use in connecting to HiRDB and Oracle is output to a trace based performance analysis. In a series of processing from the J2EE server to the database, and until the processing results are returned from the database to the J2EE server, a connection ID and a connect serial number assigned by the database server are output to the log and trace information of the related configuration software. By comparing and checking this information, you can identify the connection in which an error has occurred.

An overview of output of connection IDs when you use HiRDB, and when you use Oracle, and the information used to identify the connection where an error has occurred is described below:

- **When you use HiRDB**

  An overview of output of connection IDs is described below:

  Figure B–1: Output overview of connection ID (In HiRDB)



  The following table describes the information used to identify the connection where an error has occurred.

  Table B–1: Information used to identify the connection where an error has occurred (in HiRDB)

No.	Output source	Information type	Reference
1	Component Container	Trace based performance analysis file	*Appendix B.1*
2		Exception message when SQLException occurs	
3	DABroker Library	Extended database access trace	*Appendix B.2*
4	HiRDB client	SQL trace file	*Appendix B.3*
5		Error log information	
6		Reconnection trace	

No.	Output source	Information type	Reference
7	HiRDB server	Execution results of the `pdls` command	*Appendix B.4*

The connection ID includes the following information:

- *server-name*
  The front-end server name (when using HiRDB or a Parallel Server) or single server name (when using HiRDB or a Single Server) is displayed.
- *connect-serial-number*
  A connect serial number assigned by the HiRDB server that is displayed in server name is displayed.
- *server-process-ID*
  The process ID of the HiRDB server that is displayed in the server name is displayed.

The output format and an output example of a connection ID is described below:

Output format of a connection ID (In HiRDB)

```
server-name:connect-serial-number:server-process-ID
```

Output example of a connection ID (In HiRDB)

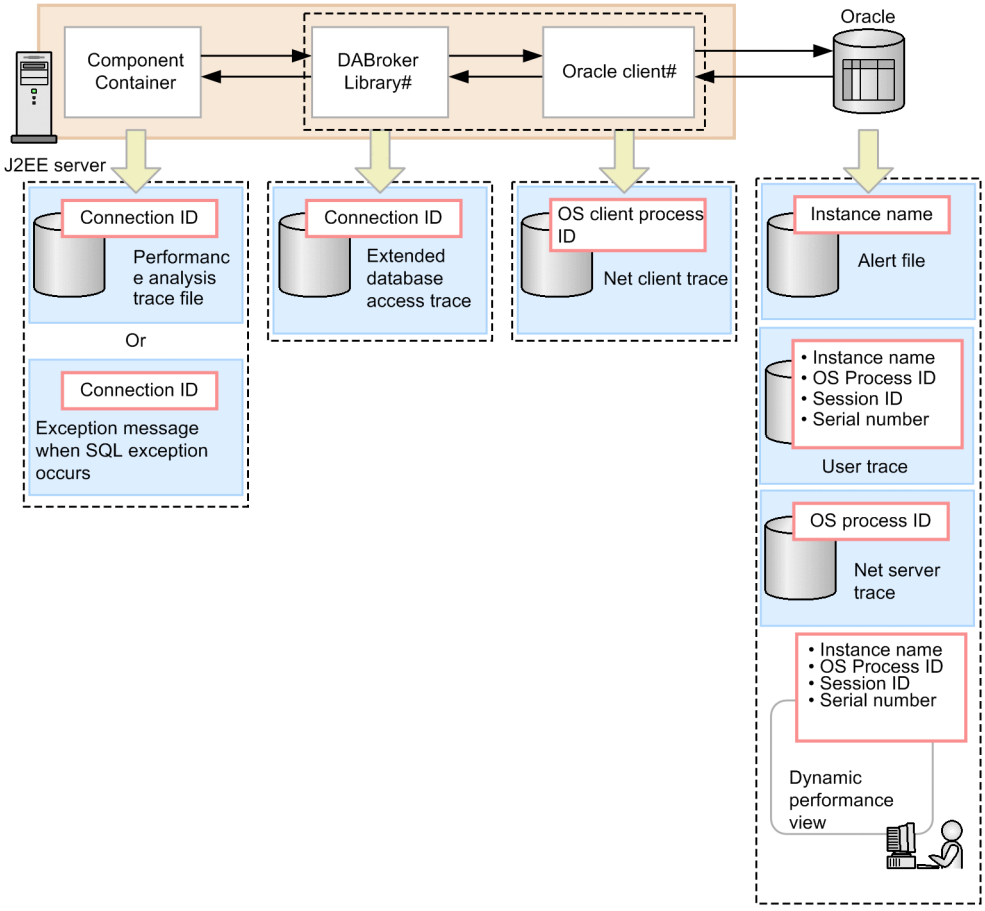
```
fes01:15:2351
```

Reference note

During the processing of a global transaction, if a connection gets broken due to an error, the connection is automatically re-established. This reconnection is, however, not output to the reconnection trace. In such a case, a discrepancy occurs in the connection ID output to the reconnection trace and the actual connection information. Basically, however, a connection does not get broken while processing a global transaction.

- **When you use Oracle**
  An overview of output of connection IDs is described below:

Figure B–2: Output overview of Connection ID (In Oracle)



When using Oracle JDBC Thin Driver, do not use Cosminexus DABroker Library and Oracle client.

The following table describes the information used to identify the connection where an error has occurred.

Table B–2: Information used to identify the connection where an error has occurred (In Oracle)

No.	Output source	Information type	Reference
1	Component Container	Trace based performance analysis file	*Appendix B.1*
2		Exception message when SQLException occurs	
3	DABroker Library	Extended database access trace	*Appendix B.2*
4	Oracle client	Net client trace	*Appendix B.5*
5	Oracle server	Alert file	*Appendix B.6*
6		User trace	
7		Net server trace	
8		Dynamic performance view	

The connection ID includes the following information:

- *instance-name*
  The instance name of the Oracle server is displayed.
- *session-ID*
  The session ID assigned by the Oracle server is displayed.

- *session-serial-number*
  The session serial number assigned by the Oracle server is displayed.
- OS-process-ID
  The OS process ID is displayed.

Output format of a connection ID (In Oracle)

```
instance-name:session-ID:session-serial-number:OS-process-name
```

Output example of a connection ID (In Oracle)

```
ORCL:17:5:920
```

The dynamic performance view of Oracle is used to generate the connection ID, therefore, a user connecting to Oracle must have permission to view the dynamic performance view. Use either of the following methods to grant viewing permission to a user connecting to Oracle:

- Execute `GRANT SELECT_CATALOG_ROLE TO user-name;`.
- Execute `GRANT SELECT ON V_$INSTANCE TO user-name;`, `GRANT SELECT ON V_$PROCESS TO user-name;`, and `GRANT SELECT ON V_$SESSION TO user-name;`.

Also, when using Oracle, if you set true in the value of the property item ConnectionIDUpdate, in property definitions of a DB Connector, you can generate a connection ID when acquiring connection. However, the SQL for generating connection ID is issued when you acquire a connection, as a result, the performance may be affected. Set false in an environment where you do not wish to perform a reconnection. For details about how to set property definitions of a DB Connector, see *4.2.2 Property definition of DB Connector* in the *uCosminexus Application Server Application Setup Guide*.

Important note

When you use transparent application failover of Oracle, the connection ID output to the PRF trace and the actual connection ID may be different. This is because the connection is re-established in Oracle. When the connection ID output to the PRF trace and the actual connection ID are different, you cannot track the trace information of Oracle with the connection ID.

B.1 Cosminexus Component Container

The output destination of connection IDs output from Cosminexus Component Container differs depending on the status of transaction processing (normal or abnormal).

In the case of normal processing

Output to a trace based performance analysis file.

In the case of abnormal processing

Output in an exception message when SQLException occurs.

In the trace based performance analysis file, connection IDs are output at the following three timings:

- When you acquire a connection to the database
  When using HiRDB, output the connection ID of the acquired connection immediately before the `DataSource.getConnection()` method or `DataSource.getConnection(String username, String password)` method terminates.

When using Oracle, output the connection ID of the acquired connection immediately before the `DataSource.getConnection()` method terminates.

- When you release the database connection

  Immediately after starting the `Connection.close()` method, output the connection ID of the connection acquired with the `getConnection` method that corresponds to the `Connection.close()` method.

- When you replace a connection to the database with the association function

  When you invoke the `ManagedConnection.associateConnection()` method, the connection ID of the destination connection is output when you replace with the association function.

The connection IDs acquired at the above timings are output to the interface name of the trace information of event IDs described in the following table.

Table B–3: Trace based performance analysis where connection IDs are output

Event ID	Description
0x8C01	Information output during the processing immediately before the termination of the `DataSource.getConnection()` method.
0x8C03	Information output during the processing immediately before the termination of the `DataSource.getConnection(String username, String password)` method.
0x8C20	Information output during the processing immediately after starting the `Connection.close()` method.
0x8C40	Information output during the invocation of the `ManagedConnection.associateConnection()` method.

For details about trace collection points and the trace information that you can acquire, see *8.11 Trace collection points of a DB Connector and JCA container*.

How to acquire a trace based performance analysis, the exception message when SQLException occurs, and their output format is described below:

(1) Trace based performance analysis file

This is a file that edits and outputs trace information, that is output during a series of processing of requests from the client to an EIS such as databases and until the processing results are returned to the client, to a PRF trace file. The trace information is output in CSV format.

Conditions for output of a connection ID

If all the below-mentioned conditions are fulfilled, the connection ID is output to a trace based performance analysis file:

- The database used is either of the following:

  HiRDB

  Oracle

- Logical performance tracer is starting
- A DB Connector is used as the resource adapter.

(a) Points to be noted

The points to be noted when you view a trace based performance analysis are described below:

- Points to be noted for releasing a connection

When you invoke the `Connection.close()` method multiple times for the same connection, the trace based performance analysis is output only for that many number of times.

- Points to be noted when you use the association functionality

  When you replace a connection to the database with the association functionality, the connection to the database is established with a physical connection different from the connection ID output with the `getConnection()` method. In such a case, when you invoke the `ManagedConnection.associateConnection()` method, the connection ID corresponding to the destination physical connection when you replace with the association functionality, is output to a trace based performance analysis. As a result, to identify the connection ID that is actually connected to the database, you need to track even the connection ID output to a trace based performance analysis with the event ID 0x8C40.

  Note that the connection may not be replaced even if you invoke the `ManagedConnection.associateConnection()` method. In such a case, the connection ID is not output to a trace based performance analysis.

  Reference note

  The association functionality replaces normal 1:1 correspondence between a logical connection and a physical connection by sharing one physical connection among multiple logical connections.

- Points to be noted when you use the automatic reconnection function

  When you use the automatic reconnection function of HiRDB, the connection ID output to a trace based performance analysis and the connection ID of the actual connection may be different. In such a case, you also need to reference the reconnection trace of the HiRDB client.

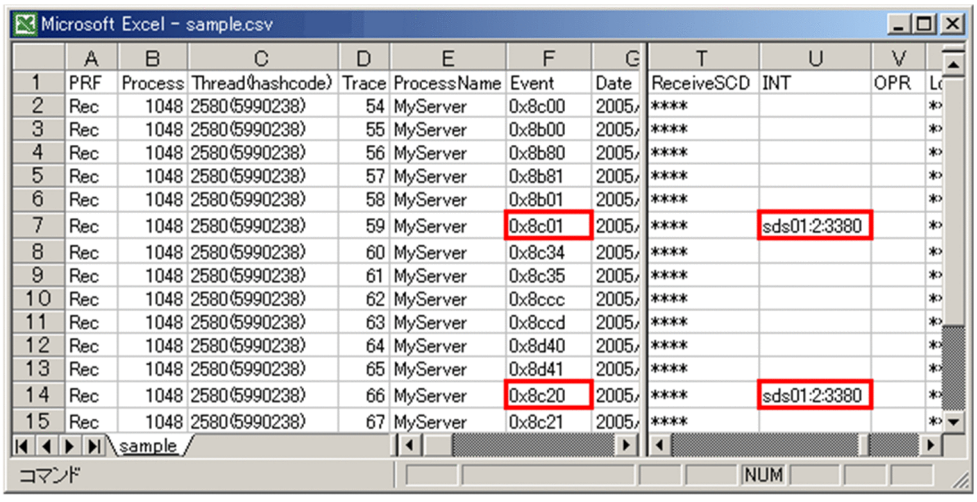
(b) How to acquire

Acquire the trace based performance analysis file by executing the management command (`mngsvrutil`). The trace based performance analysis file is output to *Manager-log-output-directory*`\prf` (in Windows), or to *Manager-log-output-directory*`/prf` (in UNIX). For details about how to acquire a trace based performance analysis file, see *7.3.1 How to collect a trace based performance analysis file*.

(c) Output format

An output example of a trace based performance analysis is described below. The connection ID is output in the interface name (INT column).

Figure B–3: Trace based performance analysis output example



	A	B	C	D	E	F	G	T	U	V
1	PRF	Process	Thread(hashcode)	Trace	ProcessName	Event	Date	ReceiveSCD	INT	OPR
2	Rec	1048	2580(5990238)	54	MyServer	0x8c00	2005/	****		
3	Rec	1048	2580(5990238)	55	MyServer	0x8b00	2005/	****		
4	Rec	1048	2580(5990238)	56	MyServer	0x8b80	2005/	****		
5	Rec	1048	2580(5990238)	57	MyServer	0x8b81	2005/	****		
6	Rec	1048	2580(5990238)	58	MyServer	0x8b01	2005/	****		
7	Rec	1048	2580(5990238)	59	MyServer	0x8c01	2005/	****	sds01:2:3380	
8	Rec	1048	2580(5990238)	60	MyServer	0x8c34	2005/	****		
9	Rec	1048	2580(5990238)	61	MyServer	0x8c35	2005/	****		
10	Rec	1048	2580(5990238)	62	MyServer	0x8ccc	2005/	****		
11	Rec	1048	2580(5990238)	63	MyServer	0x8ccd	2005/	****		
12	Rec	1048	2580(5990238)	64	MyServer	0x8d40	2005/	****		
13	Rec	1048	2580(5990238)	65	MyServer	0x8d41	2005/	****		
14	Rec	1048	2580(5990238)	66	MyServer	0x8c20	2005/	****	sds01:2:3380	
15	Rec	1048	2580(5990238)	67	MyServer	0x8c21	2005/	****		

(2) Exception message when SQLException occurs

This is a message that indicates that SQLException is thrown as an exception, when an error occurs in the database access, Cosminexus DABroker Library, or JDBC driver.

Conditions for output of a connection ID

If the database in use is either of the following, the connection ID is output in the exception message when SQLException occurs:

- HiRDB
- Oracle

(a) Points to be noted

The points to be noted when you view the exception message when SQLException occurs, are described below:

- Points to be noted for the output connection ID

  The latest connection ID is always output to the exception message when SQLException occurs. When reconnected by the automatic reconnecting functionality of HiRDB, the connection ID after being reconnected is output.

(b) How to acquire

An exception message is output to the following log files when SQLException occurs:

- In Windows
  - *working-directory*\ejb*server-name*\logs\cjexception[*n*].log
  - *working-directory*\ejb*server-name*\logs\connectors*display-name-of-resource-adapter*[*n*].log
- In UNIX
  - *working-directory*/ejb/*server-name*/logs/cjexception[*n*].log
  - *working-directory*/ejb/*server-name*/logs/connectors/*display-name-of-resource-adapter*[*n*].log

In the part of log file name [*n*], the file number (From 1 to the total number of files (maximum 16)) is added.

(c) Output format

HiRDB

The connection ID is output at the end of ErrMsg of the message KFDJ00001-E. The example of output is described below. The *bold*, highlighted part is the connection ID.

```
JP.co.Hitachi.soft.DBPSV_Driver.SQLException: KFDJ00001-E Error occurred a
t server.
[JdbcDbpsvResultSQLExecute.SQLExecute]
OperationType : 2002
ReturnCode    : -100
ErrorCode     : -404
WarningInfo   : 0
ErrorMsg     : KFPA11404-E Input data too long for column or assignment ta
rget in variable 1 [HiRDB_CONNECTION_ID(sds01:7:2988)]
```

In the case of Oracle

The connection ID is output at the end of ErrMsg of the message ORA-00942. The example of output is described below. The *bold*, highlighted part is the connection ID.

```
JP.co.Hitachi.soft.DBPSV_Driver.SQLException: KFDJ00001-E Error occurred a
t server.
[JdbcDbpsvResultSQLExecute.SQLPrepare]
OperationType : 2002
ReturnCode    : -200
ErrorCode     : 942
WarningInfo   : 0
ErrorMsg      : ORA-00942: Table or view does not exist. [ORACLE_CONNECTIO
N_ID(ORCL:17:5:920)]
PreparedSQL   : selectSectionID   : 2
```

B.2 Cosminexus DABroker Library

In Cosminexus DABroker Library, the connection ID is output to extended database access trace.

This section describes the how to acquire the extended database access trace and output format of the trace.

(1) Extended database access trace

The Extended database access trace is a trace that outputs the access information from the time of database connection until disconnection. The extended database access trace is output each time a connection is established with the database.

The following conditions must be satisfied to output the connection ID to the extended database access trace:

Conditions for output of the connection ID (In HiRDB)

- The version of HiRDB client must be 07-01 or later.
- 0 or a higher value must be specified in `DABEXSQL_TRC_LINE` of the Cosminexus DABroker Library operation environment definition file.

For details about setting the operation environment of Cosminexus DABroker Library in Windows, see *Chapter 18* in the *uCosminexus Application Server Compatibility Guide*, and for UNIX, see *Chapter 18* in the *uCosminexus Application Server Compatibility Guide*. For details about the Cosminexus DABroker Library operation environment definition file, see *Chapter 18* in the *uCosminexus Application Server Compatibility Guide*.

Conditions for output of the connection ID (In Oracle)

- You must be using Oracle9i or Oracle10g.
- -1, 0, or a value in the range of 1024 to 32767 must be specified in `DABEXSQL_TRC_LINE E` of the Cosminexus DABroker Library operation environment definition file.

In Windows, use the environment settings utility to set the Cosminexus DABroker Library operation environment definition file. For details about setting the operation environment of Cosminexus DABroker Library using the environment settings utility in Windows, see *Chapter 18* in the *uCosminexus Application Server Compatibility Guide*, and for UNIX, see *Chapter 18* in the *uCosminexus Application Server Compatibility Guide*. For details about the Cosminexus DABroker Library operation environment definition file, see *Chapter 18* in the *uCosminexus Application Server Compatibility Guide*.

(a) How to acquire

The storage location of extended database access trace is as follows:

- In Windows

  Files under the *Cosminexus-DABroker-Library-operation-directory*`\spool\db_access` directory

- In UNIX
  Files under the `/opt/DABroker/spool/db_access` directory

(b) Output format

The output format of extended database access trace is as follows:

For details about the output format of extended database access trace, see *Chapter 18* in the *uCosminexus Application Server Compatibility Guide*.

In HiRDB

The connection ID is output to the `HiRDB_CONNECTION_ID` column.

- In Windows

```
*-------------------------------------------------------*
*- DataBase Access Information (DRV)                    -*
*- Date YYYY/MM/DD hh:mm:ss.nnnnnn                      -*
*-------------------------------------------------------*
DABroker Connect ID  : AAAA(BBBBB)
Process ID           : CCCCC
UserID               : DDDD
Client Name          : EEEEE
(information-output-for-each-DB)
Lang Mode            : GGGGG

THREAD-ID CID  EVT  START-TIME  END-TIME  RETCODE BLOCKCNT (Windows Que
ryPerformance Counter) HiRDB_CONNECTION_ID
(trace-information)
(SQL)SS...SS
(trace-information)
```

- In UNIX

```
*-------------------------------------------------------*
*- DataBase Access Information (DRV)                    -*
*- Date YYYY/MM/DD hh:mm:ss.nnnnnn                      -*
*-------------------------------------------------------*
DABroker Connect ID  : AAAAA(BBBBB)
Process ID           : CCCCC
UserID               : DDDDD
Client Name          : EEEEE
(information-output-for-each-DB)
Lang Mode            : GGGGG

THREAD-ID CID  EVT  START-TIME  END-TIME  RETCODE
BLOCKCNT  HiRDB_CONNECTION_ID
(trace-information)
(SQL)SS...SS
(trace-information)
```

In Oracle

The connection ID is output to the `ORACLE_CONNECTION_ID` column.

```
*-------------------------------------------------------------------*
*- DataBase Access Information (ORACLE8i Driver)                    -*
*- Date YYYY/MM/DD hh:mm:ss.nnnnnn                                  -*
*-------------------------------------------------------------------*
```

```
DABroker Connect ID  : AAAAAAAA(BBBBB)
Process ID           : CCCC
UserID               : DDDDDD
Client Name          : EEEEEE
SQLNET               : FFFFFF
Lang Mode            : GGGG
*-------------------------------------------------------------------------*
 THREAD-ID  CID EVT     START-TIME          END-TIME            RETCODE  B
LOCKCNT ORACLE_CONNECTION_ID
(trace-information)
(SQL)SS...SS
(trace-information)
```

B.3 HiRDB Client

In a HiRDB client, information such as the connection ID and connect serial number is output to the SQL trace file, error log file, and reconnection trace file.

How to acquire the SQL trace file, error log file, and reconnection trace file, and the output format of each one is described below:

(1) SQL trace file

This is a trace file that outputs the SQL trace information of an executed UAP. This trace file is output when the execution of an SQL terminates.

If an SQL error occurs while you are executing a UAP, you can identify the SQL responsible for the error by referencing the SQL trace file.

Conditions for output of a connection ID

If the below-mentioned condition is fulfilled, the connection ID is output to the SQL trace file:

- A value is specified in the client environment definitions PDCLTPATH and PDSQLTRACE.

For details about how to set client environment definitions, see the *HIRDB UAP Development Guide*.

(a) How to acquire

The SQL trace file is stored in the directory specified in the client environment definition PDCLTPATH.

(b) Output format

The server name, connect serial number, and server process ID are output in the format are as follows:

```
(Omitted)
 ...
CONNECTION STATUS:
 CURHOST(connection-destination-host-name) CURPORT(connect-port-number) SRVN
AME(server-name)
 CNCTNO(connect-serial-number) SVRPID(server-process-ID) CLTPID(UAP-process-
ID) CLTTID(UAP-thread-serial-number)
 ...
(Omitted)
```

The SQL statement, SQL execution time, and values set in the variables within the SQL statement are also output to the SQL trace file. For details about the output format, see the *HIRDB UAP Development Guide*.

(2) Error log file

This is a log file that outputs the error information when an error occurs during the communication between a client and HiRDB server, or in the XA interface defined in X/Open. This log file is output when an error occurs during the execution of an SQL, communication processing, or execution of XA interface functions defined in X/Open.

Conditions for output of a connection ID

If the below-mentioned condition is fulfilled, the connection ID is output to the error log file:

- A value is specified in the client environment definitions PDCLTPATH and PDUAPERLOG.

For details about how to set, see the *HIRDB UAP Development Guide*.

(a) How to acquire

The SQL trace file is stored in the directory specified in the client environment definition PDCLTPATH.

(b) Output format

The server process ID is output to the SQL trace in the format described below:

```
Error-log-identifier (>> or >) UAP-process-ID UAP-thread-serial-number Serve
r-process-ID Error-log-counter (Rest omitted)
```

For details about the output format, see the *HIRDB UAP Development Guide*.

In addition to the connection information, the time of acquiring the error, SQLCODE, and SQL operation code where an error occurred is output to the error log information.

(3) Reconnection trace file

When you reconnect with the automatic reconnection function of HiRDB client, then this trace file outputs the value of the connection handle, connection information before reconnection, connection information after reconnection, and the time of reconnection that HiRDB manages internally. This trace file is output when a connection is executed automatically with the automatic reconnection function.

Conditions for output of a connection ID

If the below-mentioned conditions are fulfilled, the connection ID is output to the reconnection trace file:

- A value is specified in the client environment definitions PDCLTPATH and PDSQLTRACE.
- The automatic reconnection function of HiRDB is used.

For details about how to set up, see the *HIRDB UAP Development Guide*.

(a) How to acquire

The reconnection trace file is stored in the directory specified in the client environment definition PDCLTPATH. The stored file names are pdrcnct1.trc and pdrcnct2.trc.

(b) Output format

The reconnection trace is output in the format described below:

```
Connection-handle-value Reconnection-result(S or F) Reconnection-start-date
-and-time -
Reconnection-end-date-and-time Connection-information-before-reconnection =
> Connection-information-after-reconnection
```

The connection ID is output as the connection information after reconnection.

For details about the output format, see the *HIRDB UAP Development Guide*.

An output example of the reconnection trace file is described below. The *bold*, highlighted part is the connection ID.

```
40004250 S 2004/04/12 11:10:36.766 - 2004/04/12 11:10:41.846 sds:9:23763 =>
sds:10:23750
40004250 S 2004/04/12 11:11:07.491 - 2004/04/12 11:11:12.547 sds:10:23750 =
> sds:11:23765
40004850 F 2004/04/12 11:17:58.285 - 2004/04/12 11:18:23.395 sds:14:23751
=>
40005050 S 2004/04/12 11:27:35.098 - 2004/04/12 11:27:40.152 sds:1:24414 =>
sds:2:24418
```

B.4 HiRDB Server

In HiRDB server, check the server status by referencing the results of executing the `pdls` command.

Check the status of the connected HiRDB server by comparing the server name and server process ID displayed in the `pdls` command execution results with the information included in the connection ID.

The execution format of `pdls` command and the output format of the execution results are described below:

(1) Execution results of the pdls command

This is a command that enables the display of exclusive control status, process status, and communication control information of a HiRDB server.

(a) Execution format

Execute the following `pdls` commands, and check the server status.

- To display the exclusive control status of the server

  ```
  pdls -d lck
  ```

- To display the server process status

  ```
  pdls -d prc
  ```

- To display the server communication control information

  ```
  pdls -d rpc
  ```

For details about the `pdls` command, see the *HiRDB Command Reference*.

(b) Output format

The server name is output to the SVID column, while the server process ID is output to the PID column of the execution results.

For details about the output format, see the *HiRDB Command Reference*.

B.5 Oracle Client

In Oracle client, the details of the executed network events are output to the Net client trace file. The OS client process ID of the client is included in the file name of the Net client trace file.

A Net client trace file is output to the following locations:

- In Windows

  location-specified-by-sqlnet.ora-file#`\CLI_`*OS-client-process-ID*`.trc`
- In UNIX

  l*ocation-specified-by-sqlnet.ora-file*#`/CLI_`*OS-client-process-ID*`.trc`

The output destination of the Net client trace file is set with the Oracle sqlnet.ora file. The coding format of the sqlnet.ora file is described below. For details about the sqlnet.ora file, see the documentation on Oracle.

```
TRACE_LEVEL_CLIENT=16
TRACE_DIRECTORY_CLIENT=directory-name
TRACE_UNIQUE_CLIENT=ON
TRACE_TIMESTAMP_CLIENT=ON
```

To reference a Net client trace file, you need to specify the desired Net client trace file from the output Net client trace files. To specify the desired Net client trace file:

1. Execute the following SQL, and acquire the OS client process ID.

   ```
   SELECT PROCESS FROM V$SESSION WHERE (SID = session-ID) AND (SERIAL# = sess
   ion-serial-number);
   ```

2. Compare the acquired OS client process ID and the OS client process ID included in the file name of the Net client trace file, and specify the desired Net client trace file.

For details about the output contents of the Net client trace file, see the documentation on Oracle.

Important note

The Net client trace consumes a large amount of disk space, as a result, it may lead to a decline in the system performance. Reference the Net client trace only when required.

B.6 Oracle Server

How to output the error information of the Oracle server, and the server status, in an Oracle server are described below:

- Acquiring alert files
- Acquiring user traces
- Acquiring Net server trace files
- Using dynamic performance view

(1) Alert file

In an alert file, you can confirm the following information with an instance name of a trace based performance analysis as the key.

- All internal errors, block corruption errors, and dead lock errors that have occurred
- Management and operation of CREATE, ALTER, or DROP statements, STARTUP or SHUTDOWN statements, and ARCHIVELOG statement
- Messages and errors related to the shared server and dispatcher process functionality
- Errors that occurred during automatic refresh of the dataized view
- Values of all initialization parameters during invocation of a database and instances

Alert files are output to the following locations:

- In Windows
  location-specified-with-BACKGROUND_DUMP_DEST#\ALERT_*instance-name*.LOG
- In UNIX
  location-specified-with-BACKGROUND_DUMP_DEST#/ALERT_*instance-name*.LOG

Set with the initialization parameter USER_DUMP_DEST of Oracle. For details about USER_DUMP_DEST, see the documentation on Oracle.

Of the output alert files, specify the desired alert file by comparing the instance name output to a trace based performance analysis, and the instance name included in the alert file name.

(2) User trace

In the user trace, you can check the following information with an instance name of a trace based performance analysis, OS process ID, session ID, and session serial number as the key:

- Information of error that occurred in a server process
- Execution plan and the statistics of an SQL

(3) Net server trace file

In Oracle server, the details of the executed network events are output to a Net server trace file. The OS process ID of the server is included in the file name of the Net server trace file.

Net server trace files are output to the following locations:

- In Windows
  location-specified-by-sqlnet.ora-file#\CLI_*OS-process-ID*.TRC
- In UNIX

location-specified-by-sqlnet.ora-file# /CLI_*OS-process-ID*.TRC

The output destination of the Net server trace file is set with the Oracle sqlnet.ora file. The coding format of the sqlnet.ora file is described below. For details about the sqlnet.ora file, see the documentation on Oracle.

```
TRACE_LEVEL_CLIENT=16
TRACE_DIRECTORY_CLIENT=directory-name
TRACE_UNIQUE_CLIENT=ON
TRACE_TIMESTAMP_CLIENT=ON
```

Of the output Net server trace files, specify the desired Net server trace file by comparing the OS process ID of a trace based performance analysis and the OS process ID name included in the Net server trace file name.

For details about the output contents of the Net server trace file, see the documentation on Oracle.

Important note

The Net server trace file consumes a large amount of disk space, as a result, it may lead to a decline in the system performance. Reference the Net client trace only when required.

(4) Dynamic performance view

You can reference the details of a process by studying the dynamic performance view or identifying a session. Study the dynamic performance view by comparing it with the connection IDs output to trace based performance analysis files.

The following table describes the relationship between the connection IDs of the dynamic performance view and trace based performance analysis.

Table B–4: Relationship between the connection IDs of dynamic performance view and trace based performance analysis

Item No.	Item displayed in the dynamic performance view	Connection ID item of trace based performance analysis
1	V$INSTANCE INSTANCE_NAME	instance-name
2	V$SESSION SID	session-ID
3	V$SESSION SERIAL#	session-serial-number
4	V$PROCESS SPID	OS-process-ID

For details about the dynamic performance view, see the documentation on Oracle.

C. Recovering Tables for a CMR When an Error Occurs

This section describes how to recover tables for a CMR that you were using before the occurrence of a failure in a J2EE server, after you deploy an application containing a CMR. For details about the table for CMR, see *9.6.3 Mapping CMP2.x and the database* in the *uCosminexus Application Server Application Setup Guide*. When you deploy an application containing CMR, and a failure occurs while you start or terminate a J2EE server in a running state for the purpose of maintenance, the application containing the CMR does not start in the state in which you deployed it. If you resolve the error and make an attempt to deploy the application containing a CMR, you cannot deploy if a table with the same name that you want to create already exists in the database (to prevent sharing of tables between applications).

Here, if you regenerate the SQL, an SQL using new table names for a CMR will be generated, and you can deploy the application to enable the use of tables for a new CMR. If, however, you want to use the same relationship that you were using before terminating the J2EE server, you need to deploy in such a way so as to enable the use of tables remaining in the database.

You can specify the table in `ejbserver.ejb.cmp20.cmr.use.existing_table` parameter in `<configuration>` tag of the logical J2EE server (j2ee-server) of the Easy Setup definition file. `ejbserver.ejb.cmp20.cmr.use.existing_table` is an option to recover the related information used till an error occurred when starting the application. If you specify `true`, you can deploy such that the existing table in the data base can be used. Use this option to use the existing tables:

1. Use an SQL to confirm that the tables for CMR that you were using before the failure occurred are remaining in the database.
2. Terminate the J2EE server (take an action against the cause of failure to start the application in the deployed state).
3. Specify the following in `ejbserver.ejb.cmp20.cmr.use.existing_table` parameter in `<configuration>` tag of the logical J2EE server (j2ee-server) of the Easy Setup definition file.

   `<param-name>` tag

   ```
   ejbserver.ejb.cmp20.cmr.use.existing_table
   ```

   `<param-value>` tag

   ```
   true
   ```

4. Start the J2EE server.
5. Re-deploy the application containing a CMR that failed to start in the deployed state.

   Important note

   Here, do not regenerate the SQL. If you regenerate, an SQL with new table names will be generated, and you will not be able to use the previous tables.

6. Stop the J2EE server.
7. Whether to specify `false` in the `ejbserver.ejb.cmp20.cmr.use.existing_table` parameter in `<configuration>` tag of the logical J2EE server (j2ee-server) of the Easy Setup definition file or return to the state where this option is not set.
8. Re-start the J2EE server.

D. Main Functionality Changes in Each Version

This section describes the main functionality changes in Application Server versions prior to version 11-00 and the purpose of each change. For details on the main functionality changes in version 11-00, see *1.4 Main functionality changes in Application Server 11-00*.

The contents described in this section are as follows:

- This section gives an overview and describes the main changes in the functionality of various Application Server versions. For details on the functionality, see the description in the *Section* column of the *Reference* column. The main locations where the functionality is described in the 11-00 manual are described in the *Reference* and *Section* columns.

uCosminexus Application Server is omitted from the manual names mentioned in the *Reference* column.

D.1 Main functionality changes in 09-87

(1) Supporting standard and existing functionality

The following table describes the items that are changed to support the standard and existing functionality.

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

Item	Overview of changes	Reference manual	Reference
Support for Java SE 11	The Java SE 11 functionality can be now used.	This manual	*Chapter 9*

D.2 Main functionality changes in 09-80

(1) Supporting standard and existing functionality

he following table describes the items that are changed to support the standard and existing functionality.

Table D–2: Changes made for supporting the standard and existing functionality

Item	Overview of changes	Reference manual	Reference
Use of lambda expressions for the JAX-RS functionality	The classes included in the packages and their sub packages specified for the servlet initialization parameter in `web.xml` can now use lambda expressions.	*Web Service Development Guide*	*11.2*
Support for Java SE 9	The Java SE 9 functionality can be now used.	This manual	*Chapter 9*

(2) Maintaining and improving reliability

The following table describes the items that are changed for maintaining and improving reliability.

Table D–3: Changes made for maintaining and improving reliability

Item	Overview of changes	Reference manual	Reference
Support for Apache 2.4 by the Web server	Apache 2.4 is now supported as the base version of the Web server.	*HTTP Server User Guide*	*Chapter 6*, *Appendix G*
Use of elliptic curve cryptography for SSL communication	SSL communication that uses elliptic curve cryptography can be now used.	*HTTP Server User Guide*	*Chapter 5*, *Appendix G*
Change of the SSL library	The SSL library that provides the SSL functionality was changed to OpenSSL.	*HTTP Server User Guide*	*Chapter 5*, *Appendix G*

D.3 Main functionality changes in 09-70

(1) Supporting standard and existing functionality

he following table describes the items that are changed to support the standard and existing functionality.

Table D–4: Changes made for supporting the standard and existing functionality

Item	Overview of changes	Reference manual	Reference
Addition of JSP compilation versions 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 by JDK 8	The option for the Permanent area used to start JavaVM is changed 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*, *Appendix A.2*
Support for SHA-2 for user authentication by integrated user management	SHA-224, SHA-256, SHA-384, and SHA-512 were added as the hash algorithms for user authentication by integrated user management.	*Security Management Guide*	*5.3.1*, *5.3.9*, *5.10.7*, *12.4.3*, *12.5.3*, *13.2*, *14.2.2*
Addition of automatic start, automatic restart, and automatic stop procedures in Red Hat Enterprise Linux Server 7	The procedures for automatically starting, restarting, and stopping Management Server and Administration Agent in Red Hat Enterprise Linux Server 7 were added.	*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 describes the items that are changed with the purpose of maintaining and improving operability.

Table D–5: Changes made for maintaining and improving operability

Item	Overview of changes	Reference manual	Reference
Support for upgrading to V9.7	The procedure for changing the option for the Permanent area used to start JavaVM during a version upgrade to the option for the Metaspace area was added.	This manual	*10.3.1*, *10.3.2*, *10.3.4*
Operations using WAR files	The WAR applications configured only with WAR files can now be deployed on the J2EE servers.	*Web Container Functionality Guide*	*2.2.1*
		Common Container Functionality Guide	*15.9*
		Command Reference Guide	*cjimportwar (Import a WAR application)*
Forced release of the Explicit memory block of the Explicit Memory Management functionality	The processing to release the Explicit memory block can now be executed by using the `javagc` command at any timing.	*Expansion Guide*	*7.6.1*, *7.9*
		Command Reference Guide	*javagc (forced execution of garbage collection)*

(3) Other purposes

The following table describes the items that are changed for other purposes.

Table D–6: Changes due to other purposes

Item	Overview of changes	Reference manual	Reference
Snapshot log data to be collected	JavaVM event log data and Management Server thread dump were added as snapshot log data to be collected.	This manual	*Appendix A.2*
Output of the `cjenvsetup` command log data	Information about the execution of Component Container Administrator setup (`cjenvsetup` command) is now output to the message log.	*System Setup and Operation Guide*	*4.1.4*
		Maintenance and Migration Guide	*4.20*
		Command Reference Guide	*cjenvsetup (set up Component Container Administrator)*
Output of the CPU time to the event log of the Explicit Memory Management functionality	The CPU time taken to perform processing to release the Explicit memory block is now output to the event log of the Explicit Memory Management functionality.	This manual	*5.11.3*

Item	Overview of changes	Reference manual	Reference
Enhancement of the user-extended trace based performance analysis	The following functions were added to the user-extended trace based performance analysis: • Trace targets can now be specified in units of packages or classes in addition to being specified in units of methods, which is the usual specification method. • The range of available event IDs was expanded. • The restrictions on the number of lines that can be specified in the user-extended trace based performance analysis configuration file were relaxed. • The trace collection level can now be specified in the user-extended trace based performance analysis configuration file.	This manual	*7.5.2*, *7.5.3*, *8.23.1*
Improvement of the analysis of information when the asynchronous Session Bean invocation is used	The root application information of the PRF trace can now be used to compare the requests of the invocation source and the invocation destination.	*EJB Container Functionality Guide*	*2.17.3*

D.4 Main functionality changes in 09-60

(1) Supporting standard and existing functionality

The following table describes the items that are changed to support the standard and existing functionality.

Table D–7: Changes made for supporting the standard and existing functionality

Item	Overview of changes	Reference manual	Reference
Support for G1 GC	G1 GC 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.	This manual	*9.18*

(2) Maintaining and improving reliability

The following table describes the items that are changed for maintaining and improving reliability.

Table D–8: Changes made for maintaining and improving reliability

Item	Overview of changes	Reference manual	Reference
Addition of the finalize-retention resolution function	The retention of the finalization processing can now be resolved and the occurrence of delays in the release of OS resources can now be suppressed.	This manual	*9.16*

(3) Other purposes

The following table describes the items that are changed for other purposes.

Table D–9: Changes due to other purposes

Item	Overview of changes	Reference manual	Reference
Addition of the asynchronous output function for log files	Log data can now be asynchronously output to files.	*Definition Reference Guide*	*14.2*

D.5 Main functionality changes in 09-50

(1) Improvement of development productivity

The following table describes the items changed with the purpose of improving the development productivity.

Table D–10: Changes made with the purpose of improving the development productivity

Item	Overview of changes	Reference manual	Reference
Simplification of Eclipse setup	The GUI can now be used to set up the Eclipse environment.	*Application Development Guide*	*1.1.5*, *2.4*
Support for debugging by using the user-extended trace based performance analysis	The user-extended trace based performance analysis configuration file can now be created in the development environment.	*Application Development Guide*	*1.1.3, 6.4*

(2) Simplifying implementation and setup

The following table describes the items that are changed to simplify installation and setup.

Table D–11: Changes made for simplifying implementation and setup

Item	Overview of changes	Reference manual	Reference
Expansion of the system configuration patterns in a virtual environment	The number of tier types that can be used in a virtual environment (`http-tier`, `j2ee-tier`, and `ctm-tier`) increased. This makes it possible to set up the following system configuration patterns: • Pattern that allocates the Web server and the J2EE server to different hosts • Pattern that separately allocates the front end (servlets or JSPs) and the back end (EJBs) • Pattern that uses the CTM	*Virtual System Setup and Operation Guide*	*1.1.2*

(3) Supporting standard and existing functionality

The following table describes the items that are changed to support the standard and existing functionality.

Table D–12: Changes made for supporting the standard and existing functionality

Item	Overview of changes	Reference manual	Reference
Support for the JDBC 4.0 specifications	DB Connector now supports HiRDB Type4 JDBC Driver and SQL Server JDBC drivers complying with the JDBC 4.0 specifications.	*Common Container Functionality Guide*	*3.6.3*

Item	Overview of changes	Reference manual	Reference
Relaxation of the naming rules for the Portable Global JNDI names	Characters that can be used for Portable Global JNDI names were added.	*Common Container Functionality Guide*	*2.4.3*
Support for the Servlet 3.0 specifications	The names of HTTP cookies and URL path parameters for Servlet 3.0 can now be changed in Servlet 2.5 or earlier.	*Web Container Functionality Guide*	*2.7*
Extended use of applications that can be integrated with Bean Validation	Bean Validation can now be used for validation in the CDI and user applications as well.	*Common Container Functionality Guide*	*Chapter 9*
Support for JavaMail	The email sending and receiving functionality that uses APIs compliant with JavaMail 1.4 can now be used.	*Common Container Functionality Guide*	*Chapter 7*
Expanded scope of OSs that can use the `javacore` command	The `javacore` command can now be used to obtain the Windows thread dump.	*Command Reference Guide*	*javacore (Acquiring the thread dump/in Windows)*

(4) Maintaining and improving reliability

The following table describes the items that are changed for maintaining and improving reliability.

Table D–13: Changes made for maintaining and improving reliability

Item	Overview of changes	Reference manual	Reference
Avoidance of the depletion of the code cache area	The size of the code cache area used in the system can now be checked, and before the area is depleted, the threshold value can be changed to avoid the depletion of the area.	*System Design Guide*	*7.2.6*
		This manual	*5.7.2*, *5.7.3*
		Definition Reference Guide	*14.1*, *14.2*, *14.4*
Support for the efficient application of the Explicit Memory Management functionality	Functionality that can control the objects transferred to the explicit heap was added as functionality for reducing the automatic release processing time and for efficiently applying the Explicit Memory Management functionality. • Functionality that controls the transfer of objects to the Explicit memory block • Functionality that specifies the classes to be excluded from those for which the Explicit Memory Management functionality is applied • Output of the information about the object release rate to the 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*
		This manual	*5.5*
Expansion of the output range of statistical information for each class	The `static` field-based reference relationship can now be output to the extended thread dump that includes statistical information for each class.	This manual	*9.6*

(5) Maintaining and improving operability

The following table describes the items that are changed with the purpose of maintaining and improving operability.

Table D–14: Changes made for maintaining and improving operability

Item	Overview of changes	Reference manual	Reference
Support for the EADs session failover functionality	The EADs session failover functionality that implements the session failover functionality by integrating with the EADs is now supported.	*Expansion Guide*	*Chapter 5*
Operations by using WAR files	A WAR application configured by using only WAR files can now be deployed to a J2EE server.	*Web Container Functionality Guide*	*2.2.1*
		Common Container Functionality Guide	*15.9*
		Command Reference Guide	*cjimportwar (Import a WAR application)*
Start or stop of the management functionality by using synchronous execution	A synchronous execution option for starting or stopping the management functionality (Management Server and Administration Agent) was added.	*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 the Explicit memory block of the Explicit Memory Management functionality	The processing to release the Explicit memory block can now be executed by using the `javagc` command at any timing.	*Expansion Guide*	*7.6.1*, *7.9*
		Command Reference Guide	*javagc (forced execution of garbage collection)*

(6) Other purposes

The following table describes the items that are changed for other purposes.

Table D–15: Changes due to other purposes

Item	Overview of changes	Reference manual	Reference
Acquisition of definition information	The `snapshotlog` (collect snapshot logs) command can now be used to collect only the definition files.	This manual	*2.3*
		Command Reference Guide	*snapshotlog (collect snapshot logs)*
Output of the `cjenvsetup` command log data	Information about the execution of Component Container Administrator setup (`cjenvsetup` command) is not output to the message log.	*System Setup and Operation Guide*	*4.1.4*
		This manual	*4.20*
		Command Reference Guide	*cjenvsetup (set up Component Container Administrator)*
Support for BIG-IP v11	BIG-IP v11 was added to the available load balancer types.	*System Setup and Operation Guide*	*4.7.2*
		Virtual System Setup and Operation Guide	*2.1*
Output of the CPU time to the event log of the Explicit Memory Management functionality	The CPU time taken to perform the processing to release the Explicit memory block is now output to the event log of the Explicit Memory Management functionality.	This manual	*5.11.3*
Enhancement of the user-extended trace based performance analysis	The following functions were added to the user-extended trace based performance analysis: • Trace targets can now be specified in units of packages or classes in addition to being specified in units of methods, which is the usual specification method. • The range of available event IDs was expanded. • The restrictions on the number of lines that can be specified in the user-extended trace based performance analysis configuration file were relaxed. • The trace collection level can now be specified in the user-extended trace based performance analysis configuration file.	This manual	*7.5.2*, *7.5.3*, *8.23.1*
Improvement of the analysis of information when the asynchronous Session Bean invocation is used	The root application information of the PRF trace can now be used to compare the requests of the invocation source and the invocation destination.	*EJB Container Functionality Guide*	*2.17.3*

D.6 Main functionality changes in 09-00

(1) Simplifying implementation and setup

The following table describes the items that are changed to simplify installation and setup.

Table D–16: Changes made for simplifying implementation and setup

Item	Overview of changes	Reference manual	Reference
Changing the units to be set up and operated 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 use the file in which the virtual server group is defined, and register multiple virtual servers to a management unit in a batch.	*Virtual System Setup and Operation Guide*	*1.1.2*
Cancelling the restrictions on the environment setup by using Setup Wizard	The restrictions on the environments that can be set up with Setup Wizard have been removed. Even if an environment has been set up with another functionality, you can now unset up the environment and use Setup Wizard for setup.	*System Setup and Operation Guide*	*2.2.7*
Simplifying the procedure for deleting the setup environment	The deletion procedure has now been simplified by adding the functionality to delete a system environment setup with Management Server (`mngunsetup` command).	*System Setup and Operation Guide*	*4.1.37*
		Management Portal User Guide	*3.6*, *5.4*
		Command Reference Guide	*mngunsetup (Deleting the Management Server configuration environment)*

(2) Supporting standard and existing functionality

The following table describes the items that are changed to support the standard and existing functionality.

Table D–17: Changes made for supporting the standard and existing functionality

Item	Overview of changes	Reference manual	Reference
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*,

Item	Overview of changes	Reference manual	Reference
			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*
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 describes the items that are changed for maintaining and improving reliability.

Table D–18: Changes made for maintaining and improving reliability

Item	Overview of changes	Reference manual	Reference
Using TLSv 1.2 for SSL/TLS communication	You can now use RSA BSAFE SSL-J to execute the SSL/TLS communication with a security protocol containing TLSv 1.2.	*Security Management Guide*	*7.3*

(4) Maintaining and improving operability

The following table describes the items that are changed with the purpose of maintaining and improving operability.

Table D–19: Changes made for maintaining and improving operability

Item	Overview of changes	Reference manual	Reference
Monitoring the total pending queues of the entire Web container	You can now output the total pending queues of the entire Web container in the operation information and monitor the number of queues.	*Operation, Monitoring, and Linkage Guide*	*Chapter 3*
Output of 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.	This manual	*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 Preference Information window • Start windows for the Web server, J2EE server, and SFO server • Batch start, batch restart, and start windows for Web server cluster and J2EE server cluster	*Management Portal User Guide*	*10.10.1*, *11.9.2*, *11.10.2*, *11.10.4*, *11.10.6*, *11.11.2*, *11.12.2*, *11.12.4*, *11.2.6*

Item	Overview of changes	Reference manual	Reference
Adding the restart functionality in the management functionality	You can now specify automatic restart for the management functionality (Management Server and Administration Agent), and continue operations even when an error occurs in the management functionality. The procedure for automatic start settings 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 setup of autostart and autorestart)*

(5) Other purposes

The following table describes the items that are changed for other purposes.

Table D–20: Changes due to other purposes

Item	Overview of changes	Reference manual	Reference
Changing the file switching units when log is output	The output destination files are now switched by date when the log is output.	This manual	*3.2.1*
Changing the Web server name	The name of the Web server included in Application Server is changed to Cosminexus HTTP Server.	*HTTP Server User Guide*	--
Supporting direct connection by using the BIG-IP APIs (SOAP architecture)	Direct connection is now supported by using APIs (SOAP architecture) in BIG-IP (load balancer). Also, 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.

D.7 Main functionality changes in 08-70

(1) Simplifying implementation and setup

The following table describes the items that are changed to simplify installation and setup.

Table D–21: Changes made for simplifying implementation and setup

Item	Overview of changes	Reference	Section
Improving the Management portal	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, J2EE applications (ear file and zip file) can now be uploaded on Management Server using the Management portal window.	*First Step Guide*	*3.5*
		Management Portal User Guide	--
Adding functionality for implicitly importing the	The functionality for implicitly importing the import property of the `page/tag` directive can now be used.	*Web Container Functionality Guide*	*2.3.7*

Item	Overview of changes	Reference	Section
import property for the `page/tag` directive			
Support for automating the environment settings corresponding to JP1 products in a virtual environment	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 by using 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. 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 standard and existing functionality

The following table describes the items that are changed to support standard and existing functionality.

Table D–22: Changes made for supporting standard and existing functionality

Item	Overview of changes	Reference	Section
Adding items to be specified in `ejb-jar.xml`	A class level interceptor and a method level interceptor can now be specified in `ejb-jar.xml`.	*EJB Container Functionality Guide*	*2.15*
Supporting the parallel copy garbage collection	The parallel copy garbage collection can now be selected.	*Definition Reference Guide*	*14.5*
Supporting the global transaction of the Inbound resource adapter conforming to Connector 1.5 specifications	`Transacted Delivery` can now be used in the resource adapters conforming to Connector 1.5 specifications. This enables participation of EIS invoking the Message-driven Bean in the global transaction.	*Common Container Functionality Guide*	*3.16.3*
Supporting MHP of a TP1 inbound adapter	MHP can now be used 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 describes the items that are changed for maintaining and improving reliability.

Table D–23: Changes for maintaining and improving reliability

Item	Overview of changes	Reference	Section
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.	This manual	*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 describes the items that are changed with the purpose of maintaining and improving operability.

Table D–24: Changes with the purpose of maintaining and improving operability

Item	Overview of changes	Reference	Section
Supporting 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 describes the items that are changed for other purposes.

Table D–25: Changes due to other purposes

Item	Overview of changes	Reference	Section
Supporting the direct connection using APIs (REST Architecture) to the load balancing functionality	Direct connection using APIs (REST architecture) is now supported as a method to connect to the Load balancing functionality. ACOS (AX2500) is 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 snapshot log collection (timeout) at a specified time. The contents collected as primary delivery data have been changed.	This manual	*Appendix A*

D.8 Main functionality changes in 08-53

(1) Simplifying implementation and setup

The following table describes the items that are changed to simplify installation and setup.

Table D–26: Changes made for simplifying implementation and setup

Item	Overview of changes	Reference	Section
Setting up a virtual environment supporting various hypervisors	Application Server can now be set up 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 standard and existing functionality

The following table describes the items that are changed to support standard and existing functionality.

Table D–27: Changes made for supporting standard and existing functionality

Item	Overview of changes	Reference	Section
Invocation from OpenTP1 supporting transaction integration	Transactions can now be integrated 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 describes the items that are changed for maintaining and improving reliability.

Table D–28: Changes made for maintaining and improving reliability

Item	Overview of changes	Reference	Section
Enhancing the JavaVM troubleshooting functionality	The following functionality can now be used as the JavaVM troubleshooting functionality: • The operations when OutOfMemoryError occurs can now be changed. • You can now set up an upper limit for the amount of C heap allocated during JIT compilation. • You can now set up the maximum thread count. • The output items of the extended verbosegc information have been extended.	This manual	*Chapter 4*, *Chapter 5*, *Chapter 9*

(4) Maintaining and improving operability

The following table describes the items that are changed with the purpose of maintaining and improving operability.

Table D–29: Changes made for maintaining and improving operability

Item	Overview of changes	Reference	Section
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 describes the items that are changed for other purposes.

Table D–30: Changes due to other purposes

Item	Overview of changes	Reference	Section
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.

D.9 Main functionality changes in 08-50

(1) Simplifying implementation and setup

The following table describes the items that are changed to simplify installation and setup.

Table D–31: Changes made for simplifying implementation and setup

Item	Overview of changes	Reference	Section
Changing the mandatory tag for specifying web.xml in the Web service provider	The specification of the listener tag, servlet tag, and servlet-mapping tag was changed from mandatory to optional in web.xml in the Web service provider.	*Definition Reference Guide*	*2.2.3*
Using the network resources of the logical server	The functionality for accessing the network resources and network drive existing on another host from the J2EE applications was added.	*Operation, Monitoring, and Linkage Guide*	*1.2.3*, *5.2*, *5.7*
Simplifying the execution procedure of the sample program	The procedure of executing the sample program was simplified by providing a part of the sample program in the EAR format.	*First Step Guide*	*3.5*
		System Setup and Operation Guide	*Appendix L*
Improving the operations in the management portal window	The default update interval of the window was changed from "Do not update" to "3 seconds".	*Management Portal User Guide*	*7.4.1*
Improving the Setup Wizard completion window	The Easy Setup definition file and Connector property file used for setup are now displayed on the window for Setup Wizard completion.	*System Setup and Operation Guide*	*2.2.6*
Setting up the virtual environment	The procedure of setting up Application Server on the virtual server, implemented by using a hypervisor, has been added.#	*Virtual System Setup and Operation Guide*	*Chapter 3*, *Chapter 5*

#
To set up in the 08-50 mode, see *Appendix D Settings to use the virtual server manager in the 08-50 mode* in the manual *uCosminexus Application Server Virtual System Setup and Operation Guide*.

(2) Supporting standard and existing functionality

The following table describes the items that are changed to support standard and existing functionality.

Table D–32: Changes made for supporting standard and existing functionality

Item	Overview of changes	Reference	Section
Supporting invocation from OpenTP1	The Message-driven Bean running on Application Server can now be invoked from OpenTP1.	*Common Container Functionality Guide*	*Chapter 4*
Supporting JMS	The Cosminexus JMS Provider functionality conforming to the JMS 1.1 specifications can now be used.	*Common Container Functionality Guide*	*Chapter 6*
Supporting Java SE 6	The Java SE 6 functionality can now be used.	This manual	*5.5*, *5.8.1*
Supporting the use of generics	Generics can now be used with EJB.	*EJB Container Functionality Guide*	*4.2.18*

(3) Maintaining and improving reliability

The following table describes the items that are changed for maintaining and improving reliability.

Table D–33: Changes made for maintaining and improving reliability

Item	Overview of changes	Reference	Section
Improving the usability of the Explicit Memory Management functionality	The Explicit Memory Management functionality can now be used easily by using the automatic allocation configuration file.	*System Design Guide*	*7.2*, *7.7.3*, *7.11.4*, *7.12.1*
		Expansion Guide	*Chapter 7*
Controlling the database session failover functionality for URIs	When using the database session failover functionality, you can now specify the requests that will be outside the scope of the functionality for URIs.	*Expansion Guide*	*5.6.1*
Monitoring errors in the virtual environment	In a virtual system, the virtual server is now monitored and the occurrence of errors can be detected.	*Virtual System Setup and Operation Guide*	*Appendix D*

(4) Maintaining and improving operability

The following table describes the items that are changed with the purpose of maintaining and improving operability.

Table D–34: Changes made for maintaining and improving operability

Item	Overview of changes	Reference	Section
Omitting the management user account	The login ID and password of the user can now be omitted in the management portal, Management Server commands, or Smart Composer functionality commands.	*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/ setting up Management Server)*, *mngsvrutil (Management Server management commands)*, *8.3*, *cmx_admin_passwd (Setting up the management user account of Management Server)*

Item	Overview of changes	Reference	Section
Virtual environment operations	The procedure of executing batch start, batch stop, scale in, and scale out for multiple virtual servers has been added in the virtual system. #	*Virtual System Setup and Operation Guide*	*Chapter 4*, *Chapter 6*

#

To operate in the 08-50 mode, see *Appendix D Settings to use the virtual server manager in the 08-50 mode* in the manual *uCosminexus Application Server Virtual System Setup and Operation Guide*.

(5) Other purposes

The following table describes the items that are changed for other purposes.

Table D–35: Changes due to other purposes

Item	Overview of changes	Reference	Section
Unused objects statistical functionality in the Tenured area	Only the unused objects can now be identified in the Tenured area.	This manual	*9.8*
Base object list output functionality for Tenured augmentation factors	The information of the objects that form the base of unused objects, identified by using the unused objects statistical functionality in the Tenured area, can now be output.		*9.9*
Class-wise statistical information analysis functionality	The class-wise statistical information can now be output in the CSV format.		*9.10*
Cluster node switching based on the detection of the exceeding of the auto-restart frequency of the logical server	Node switching can now be executed when the logical server is in an abnormally stopped state (state in which the auto-restart count is exceeded, or state in which an error is detected when the auto-restart count is set to 0) in a cluster configuration wherein Management Server is to be monitored for node switching.	*Operation, Monitoring, and Linkage Guide*	*16.2.2*, *16.3.3*, *16.3.4*, *18.4.3*, *18.5.3*
Node switching system for the host unit management model	Node switching can now be executed for the host unit management model in the system operations integrated with the cluster software.		*Chapter 16*
Supporting ACOS (AX2000, BS320)	ACOS (AX2000, BS320) was added to the types of load balancers that are available.	*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*
Transaction attributes that can be specified in the Stateful Session Bean (SessionSynchronization) when transactions are managed with CMT, have been added	`Supports`, `NotSupported`, and `Never` can now be specified as the transaction attributes in the Stateful Session Bean (SessionSynchronization), when transactions are managed with CMT.	*EJB Container Functionality Guide*	*2.7.3*
Forcibly terminating Administration Agent when OutOfMemoryError occurs	Administration Agent is now forcibly terminated when OutOfMemoryError occurs in JavaVM.	This manual	*2.5.5*

Item	Overview of changes	Reference	Section
Asynchronous parallel processing of threads	Asynchronous timer processing and asynchronous thread processing can now be implemented by using `TimerManager` and `WorkManager`.	*Expansion Guide*	--

D.10 Main functionality changes in 08-00

(1) Improvement of development productivity

The following table describes the items changed with the purpose of improving the development productivity.

Table D–36: Changes made with the purpose of improving the development productivity

Item	Overview of changes	Reference	Section
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) Support to standard functionality

The following table describes the items changed with the purpose of supporting the standard functionality.

Table D–37: Changes made with the purpose of supporting the standard functionality

Item	Overview of changes	Reference	Section
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*

Item	Overview of changes	Reference	Section
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 09-00 and later versions, you can use the JSP debug functionality in the development environment using WTP.

(3) Maintenance and improvement of reliability

The following table describes the items changed with the purpose of maintaining or improving reliability.

Table D–38: Changes made with the purpose of maintaining or improving reliability

Item	Overview of changes	Reference	Section
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 09-00.

(4) Maintenance and improvement of operability

The following table describes the items that are changed with the purpose of maintaining and improving operability.

Table D–39: Changes made with the purpose of maintaining and improving operability

Item	Overview of changes	Reference	Section
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 describes the items changed with some other purpose.

Table D–40: Changes made with other purposes

Item	Overview of changes	Reference	Section
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*
snapshot log	Changed the collection contents of the snapshot log.	This manual	*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 GC selection functionality	Enabled the selection of whether or not to execute a GC before the output of class-wise statistical information.	This manual	*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 JavaVM log file.	This manual	*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 has been deleted in 09-00.

E. 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

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W