HITACHI
Inspire the Next

In-Memory Data Grid

Hitachi Elastic Application Data Store

User's Guide

3020-3-V22-01(E)

Notices

■ Relevant program products

For Red Hat Enterprise Linux Server 6 (64-bit x86_64)

P-9W43-8A41 Hitachi Elastic Application Data Store 04-00

P-9W43-8B41 Hitachi Elastic Application Data Store Client for Java 04-00

P-9W43-8C41 Hitachi Elastic Application Data Store Client for C 04-00

■ Trademarks

HITACHI, Cosminexus, JP1, uCosminexusare either trademarks or registered trademarks of Hitachi, Ltd. in Japan and other countries.

AMD is a trademark of Advanced Micro Devices, Inc.

Intel is a trademark of Intel Corporation in the U.S. and/or other countries.

Linux$^{(R)}$ is the registered trademark of Linus Torvalds in the U.S. and other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

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

RSA and BSAFE are registered trademarks or trademarks of EMC Corporation in the United States and other countries.

This product includes RSA BSAFE$^{(R)}$ Cryptographic software of EMC Corporation.

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

This product includes software developed by IAIK of Graz University of Technology.

This product includes software developed by Ben Laurie for use in the Apache-SSL HTTP server project.

Portions of this software were developed at the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign.

This product includes software developed by the University of California, Berkeley and its contributors.

This software contains code derived from the RSA Data Security Inc. MD5 Message-Digest Algorithm, including various modifications by Spyglass Inc., Carnegie Mellon University, and Bell Communications Research, Inc (Bellcore).

Regular expression support is provided by the PCRE library package, which is open source software, written by Philip Hazel, and copyright by the University of Cambridge, England. The original software is available from ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/

This product includes software developed by Ralf S. Engelschall <rse@engelschall.com> for use in the mod_ssl project (http://www.modssl.org/).

This product includes software developed by the Java Apache Project for use in the Apache JServ

servlet engine project (http://java.apache.org/).

This product includes software developed by Daisuke Okajima and Kohsuke Kawaguchi (http://relaxngcc.sf.net/).

This product includes software developed by Andy Clark.

Other product and company names mentioned in this document may be the trademarks of their respective owners. Throughout this document Hitachi has attempted to distinguish trademarks from descriptive terms by writing the name with the capitalization used by the manufacturer, or by writing the name with initial capital letters. Hitachi cannot attest to the accuracy of this information. Use of a trademark in this document should not be regarded as affecting the validity of the trademark.

This product includes *uCosminexus Primary Server Base* as a mass market encryption program.

For details about how to use this product, see the following manual:

Hitachi software manual: Online manual Cosminexus

http://itdoc.hitachi.co.jp/Pages/document_list/manuals/cosmiv9_en.html

■ Restrictions

Information in this document is subject to change without notice and does not represent a commitment on the part of Hitachi. The software described in this manual is furnished according to a license agreement with Hitachi. The license agreement contains all of the terms and conditions governing your use of the software and documentation, including all warranty rights, limitations of liability, and disclaimers of warranty.

Material contained in this document may describe Hitachi products not available or features not available in your country.

No part of this material may be reproduced in any form or by any means without permission in writing from the publisher.

■ Issued

Aug. 2015: 3020-3-V22-01(E)

■ Copyright

Preface

This manual explains how to design, set up, and operate Hitachi Elastic Application Data Store, and how to develop Hitachi Elastic Application Data Store application programs.

■ Intended readers

This manual is intended for readers who design, set up, or operate a system in which Hitachi Elastic Application Data Store is deployed, and for developers of Hitachi Elastic Application Data Store application programs.

Readers of this manual must have:

- A basic knowledge of in-memory data grids
- Knowledge of Linux
- A basic knowledge of program development using the Java or C programming language (application program developers)

■ Conventions: Abbreviations for product names

This manual uses the following abbreviations for product names and Java-related terms:

Abbreviation			Full name or meaning
EADS	EADS server		Hitachi Elastic Application Data Store
	EADS client	EADS client (Java)	Hitachi Elastic Application Data Store Client for Java
		EADS client (C)	Hitachi Elastic Application Data Store Client for C
JavaVM			Java Virtual Machine
JDK			Java Developer's Kit

■ Conventions: Acronyms

This manual also uses the following acronyms:

Acronym	Full name or meaning
API	Application Programming Interface
KVS	Key-Value store

■ Conventions: Fonts and symbols

The following table explains the text formatting conventions used in this manual:

Text formatting	Convention
Bold	Bold characters indicate text in a window, other than the window title. Such text includes menus, menu options, buttons, radio box options, or explanatory labels. For example: • From the **File** menu, choose **Open**.

Text formatting	Convention
	• Click the **Cancel** button. • In the **Enter name** entry box, type your name.
Italic	Italic characters indicate a placeholder for some actual text to be provided by the user or system. For example: • Write the command as follows: `copy` *source-file target-file* • The following message appears: `A file was not found. (file =` *file-name*`)` Italic characters are also used for emphasis. For example: • Do *not* delete the configuration file.
`Monospace`	Monospace characters indicate text that the user enters without change, or text (such as messages) output by the system. For example: • At the prompt, enter `dir`. • Use the `send` command to send mail. • The following message is displayed: `The password is incorrect.`

The following table explains the symbols used in this manual:

Symbol	Convention
|	In syntax explanations, a vertical bar separates multiple items, and has the meaning of OR. For example: `A|B|C` means `A`, or `B`, or `C`.
`{ }`	In syntax explanations, curly brackets indicate that only one of the enclosed items is to be selected. For example: `{A|B|C}` means only one of `A`, or `B`, or `C`.
`[ ]`	In syntax explanations, square brackets indicate that the enclosed item or items are optional. For example: `[A]` means that you can specify `A` or nothing. `[B|C]` means that you can specify `B`, or `C`, or nothing.
`...`	In coding, an ellipsis (`...`) indicates that one or more lines of coding have been omitted. In syntax explanations, an ellipsis indicates that the immediately preceding item can be repeated as many times as necessary. For example: `A, B, B, ...` means that, after you specify `A, B,` you can specify `B` as many times as necessary.

■ Conventions: KB, MB, GB, and TB

This manual uses the following conventions:

- 1 KB (kilobyte) is 1,024 bytes.
- 1 MB (megabyte) is $1,024^2$ bytes.
- 1 GB (gigabyte) is $1,024^3$ bytes.
- 1 TB (terabyte) is $1,024^4$ bytes.

■ Conventions: Version numbers

The version numbers of Hitachi program products are usually written as two sets of two digits each, separated by a hyphen. For example:

- Version 1.00 (or 1.0) is written as 01-00.
- Version 2.05 is written as 02-05.
- Version 2.50 (or 2.5) is written as 02-50.
- Version 12.25 is written as 12-25.

The version number might be shown on the spine of a manual as *Ver. 2.00*, but the same version number would be written in the program as *02-00*.

Contents

Part 2: Design and Configuration

Part 4: Application Program Development

Part 5: Useful Lists and Messages

1

About Hitachi Elastic Application Data Store

Hitachi Elastic Application Data Store (EADS) is a software product that allows you to configure an in-memory data grid. This chapter provides an overview of Hitachi Elastic Application Data Store and describes its features.

1.1 In-memory data grid for efficient processing of large volumes of data in a manner appropriate to the characteristics of your IT system

Hitachi Elastic Application Data Store (abbreviated hereafter to *EADS*) is a data processing infrastructure product for configuring an in-memory data grid. An in-memory data grid is a data processing infrastructure that distributes a large volume of data to multiple servers' memory for the purpose of efficient processing of that data.

Today's social environment is undergoing rapid changes that are being driven primarily by advances in IT technology - especially in the development of Internet technology and in the expansion of services being deployed on the Internet. The widespread use of high-performance mobile devices, such as mobile phones and smartphones, has made these changes happen at unprecedented speeds.

With the evolution of the social environment, the amount of data handled by IT systems has increased exponentially. Users desire fast response times even during peak access periods, and they might wish for servers to be added temporarily in response to fluctuations in the volume of data and the number of accesses.

It has become difficult to satisfy such demands with IT systems that use only a relational database management system (RDBMS) to manage data.

Data processing infrastructure software products based on a different concept from conventional RDBMSs are now attracting attention, such as in-memory data grids. These software products, which are often called NoSQL (Not Only SQL) databases, complement RDBMS.

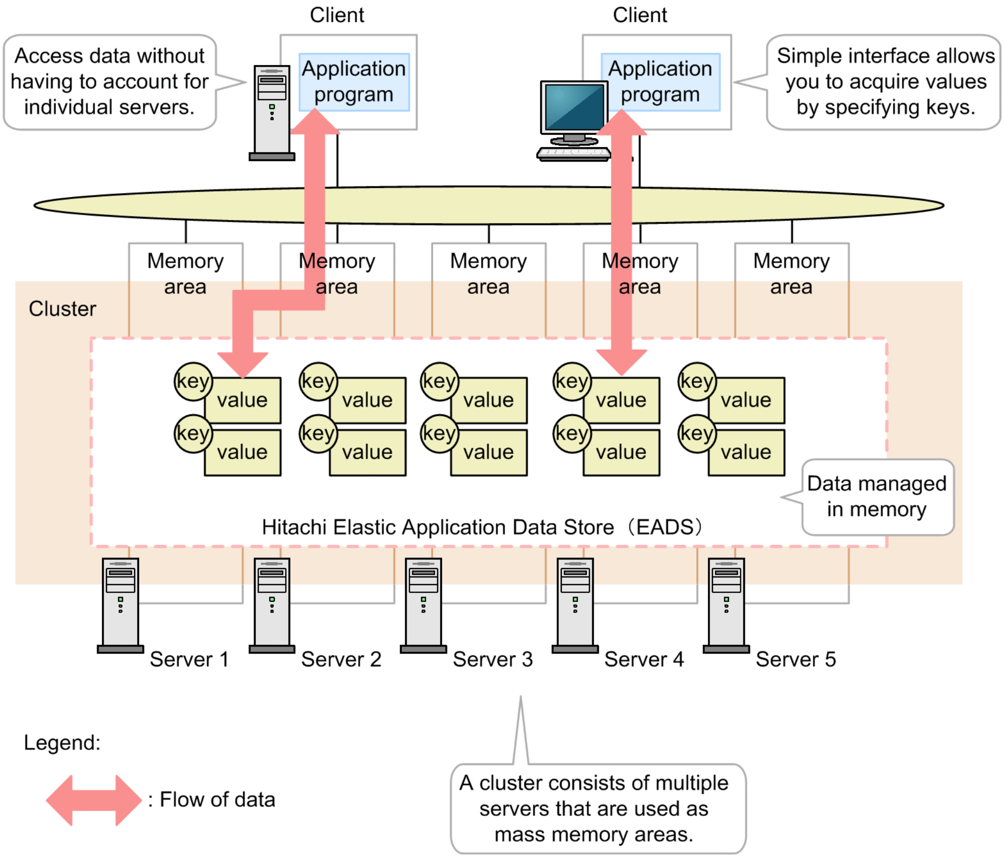
With the advent of NoSQL, you can now satisfy various needs by combining an RDBMS and complementary software products, rather than by relying solely on an RDBMS for all data management tasks.

EADS provides a data processing infrastructure that complements an RDBMS by configuring an in-memory data grid that provides both high scalability and high reliability.

1.2 Overview of EADS

This section provides an overview of EADS.

Figure 1–1: Overview of EADS


Client
Client
Access data without having to account for individual servers.
Application program
Application program
Simple interface allows you to acquire values by specifying keys.
Memory area
Memory area
Memory area
Memory area
Memory area
Cluster
key
value
Data managed in memory
Hitachi Elastic Application Data Store（EADS）
Server 1
Server 2
Server 3
Server 4
Server 5
Legend:
: Flow of data
A cluster consists of multiple servers that are used as mass memory areas.

1.2.1 Easy data management and simple interface

EADS employs a distributed in-memory KVS as one of the in-memory data grid implementation methods. KVS manages data in a simple format that consists of data (values) and keys that uniquely identify the data. Its simple interface, which prompts you to specify a key to obtain a value, makes developing application programs easy.

1.2.2 Managing data in memory

EADS manages data in memory areas. This method eliminates the overhead of disk access processing.

1.2.3 Mass memory areas consisting of multiple servers

EADS manages multiple servers as a group. This group is called a cluster. EADS treats the memory areas of the servers in the cluster as a single memory area. This method can process a large volume of data that exceeds the memory capacity of the individual servers.

Also, clients can access the data without needing to be aware of individual servers.

1.3 Features of EADS

EADS has the following features:

- Ability to achieve high scalability from a small starting point
- Data redundancy to achieve a fault-tolerant system with high availability
- High-speed data processing for faster response times
- Flexible data manipulation to better match user needs
- Management of persistent data using disk areas

The following explains each of these features.

1.3.1 Ability to achieve high scalability from a small starting point

EADS makes it easy for you to change the configuration of your system, such as by adding or removing servers. When you change the system configuration, you can scale in and out flexibly by relocating data. This feature gives you high scalability.

Because it is easy to change the system configuration after operations have begun, you can minimize initial costs (by starting small), and then expand your system as your business grows.

1.3.2 Data redundancy to achieve a fault-tolerant system with high availability

EADS can create redundant copies of data by automatically copying it to multiple servers. If one of the servers fails, the user can continue processing by using the data that was copied to another server.

By having redundant copies of data, you avoid losing data in memory in the event of a failure.

1.3.3 High-speed data processing for faster response times

EADS places all of the data to be processed in memory. This eliminates disk access overhead, resulting in high-speed data processing.

The time required to access data is reduced, although complex queries (such as in SQL) cannot be performed due to the simple data structure, which consists of keys and values.

This feature gives you fast response times.

1.3.4 Flexible data operations to better match user needs

EADS provides a set of API functions for basic data operations, and a set of API functions for performing flexible data operations according to user needs. By using the API functions provided by EADS, the user can execute on servers a program in which a series of data operations (user processing) has been defined, such as for totaling and analyzing data.

1.3.5 Management of persistent data using disk areas

EADS manages data in memory areas. To make data persistent, EADS can manage data by using both memory areas and disk areas or by using only disk areas.

Note that when disk areas are used for data management, response performance is poorer than when memory areas are used because of the overhead required for disk access processing.

1.4 Examples of EADS applications

This section presents an example of EADS deployment that benefits the user, and an example that does not benefit the user.

- An example of EADS deployment that benefits the user

  EADS is suitable for standard applications, such as high-speed processing of large amounts of data.

  Because EADS creates redundant copies of data to achieve high-availability and fault-tolerant systems, it is also suitable for enterprise systems that must provide high reliability.

  1.4.1 Using EADS as a write buffer shows an example of using EADS as a write buffer, and *1.4.2 Using EADS as a read cache* shows an example of using EADS as a read cache.

- An example of EADS deployment that does not benefit the user

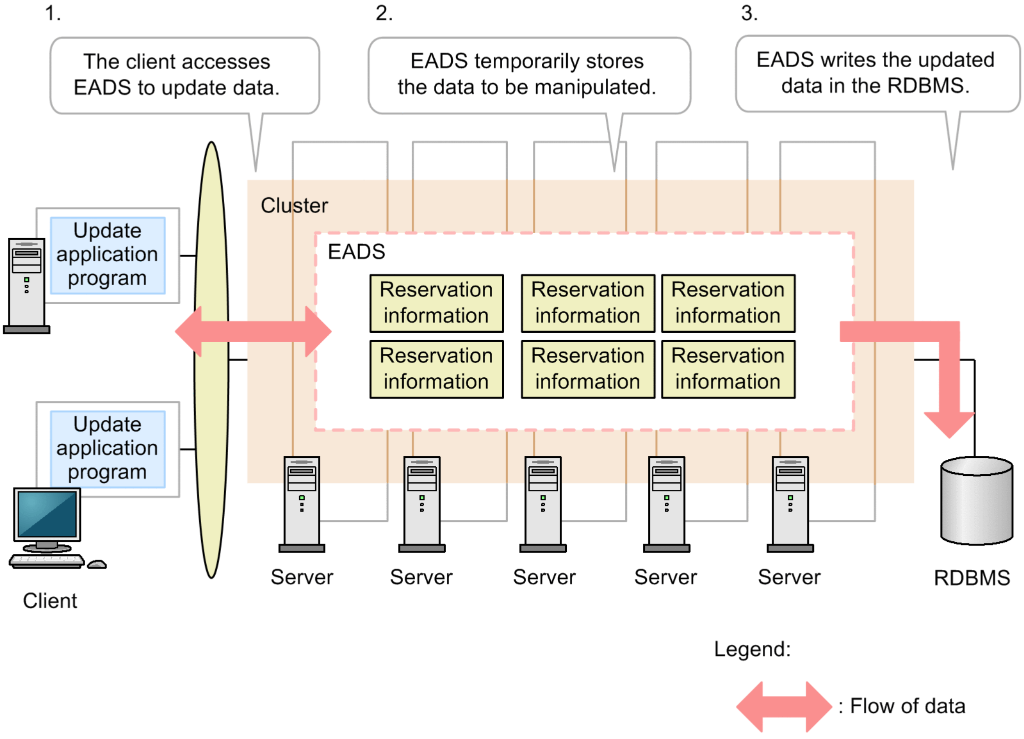
  EADS is not suitable for atypical applications that require complex queries, such as in SQL, because EADS manages data in a simple format consisting of keys and values.

1.4.1 Using EADS as a write buffer

You can use EADS as a write buffer in application systems. A write buffer is memory for storing data temporarily to eliminate disk access overhead during data write operations.

The figure below shows an example of an update application program that uses EADS as a write buffer. This example might be for a system that handles a large volume and wide variety of data, such as a reservations management system or an online trading system.

Figure 1–2: Using EADS as a write buffer



1. A client accesses EADS to update data such as reservation information.
2. EADS temporarily stores the data to be manipulated by the client (using EADS as a write buffer).

3. EADS writes the updated data in the RDBMS.

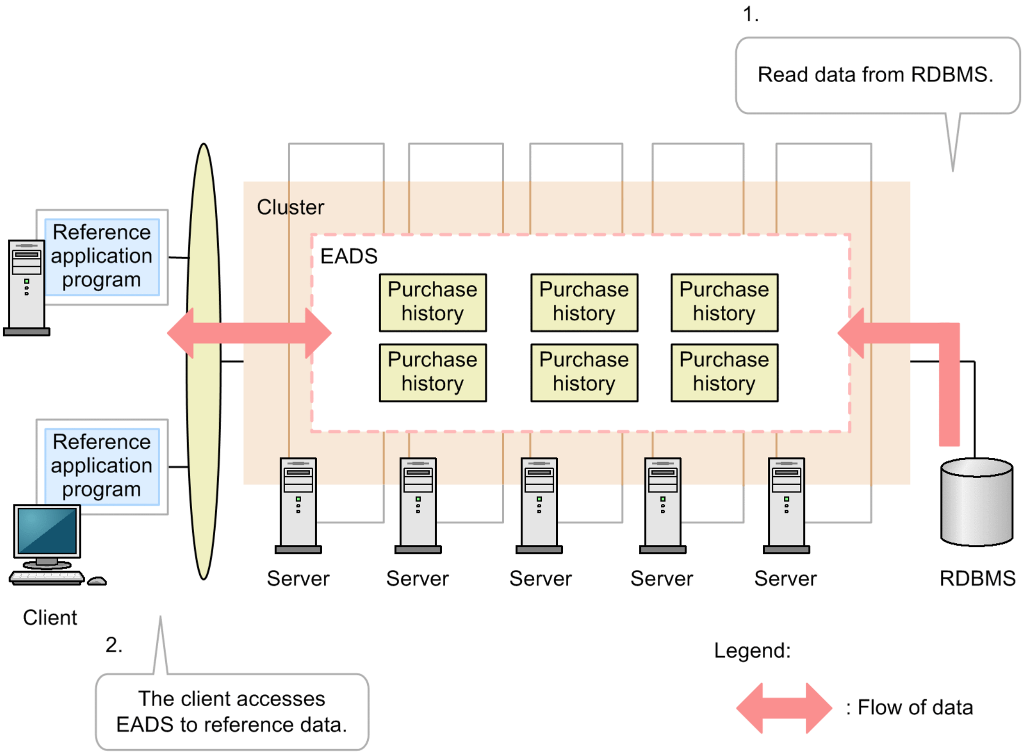
You can eliminate disk access overhead and achieve high-speed data processing by using EADS as a write buffer, without writing data directly in the database.

1.4.2 Using EADS as a read cache

You can use EADS as a read cache in application systems. A read cache is memory for storing data temporarily to achieve high-speed data read operations.

The figure below shows an example of a reference application program that uses EADS as a read cache. This example might be for a system that displays customer information and customer purchase histories, such as for an online order fulfillment system.

Figure 1–3: Using EADS as a read cache



1. First load the required data, such as a purchase history, from the RDBMS to EADS (read cache).
2. The client accesses EADS and references data.

You can eliminate disk access overhead and achieve high-speed data processing by using EADS as a read cache, without reading data directly from the RDBMS.

1.5 User tasks and corresponding parts of the manual

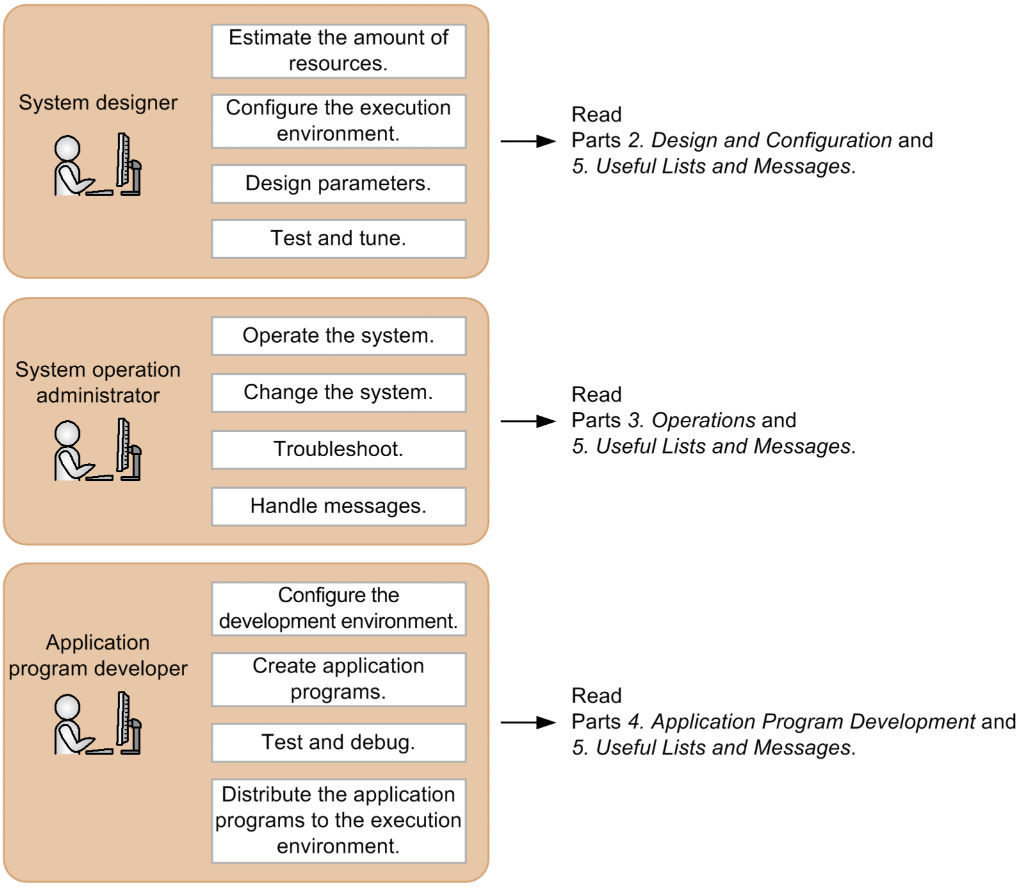
This manual is intended for the following readers:

- System designers
- System operation administrators
- Application program developers

This manual consists of the following five parts:

- *Part 1 Description*
- *Part 2 Design and Configuration*
- *Part 3 Operations*
- *Part 4 Application Program Development*
- *Part 5 Useful Lists and Messages*

The following figure shows tasks that the user performs, and the parts of the manual that describe those tasks.


System designer
Estimate the amount of resources.
Configure the execution environment.
Design parameters.
Test and tune.
Read
Parts 2. *Design and Configuration* and
5. *Useful Lists and Messages*.
System operation administrator
Operate the system.
Change the system.
Troubleshoot.
Handle messages.
Read
Parts 3. *Operations* and
5. *Useful Lists and Messages*.
Application program developer
Configure the development environment.
Create application programs.
Test and debug.
Distribute the application programs to the execution environment.
Read
Parts 4. *Application Program Development* and
5. *Useful Lists and Messages*.

2 Architecture

This chapter explains the configuration and architecture of EADS.

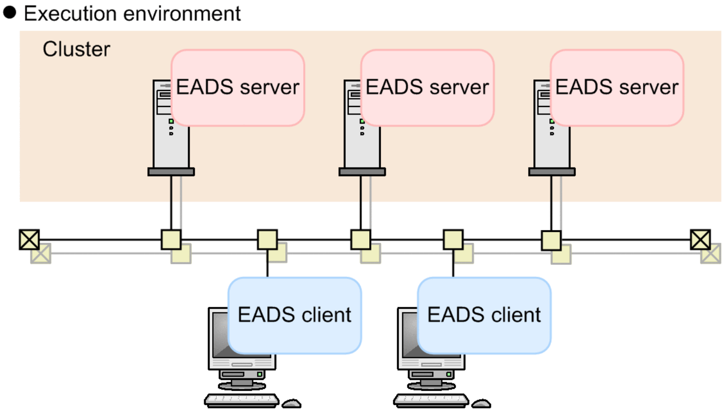
2.1 Configuration of EADS

This section explains how to configure a system in which EADS is deployed, and how to configure processes.

2.1.1 System configuration

The following figure shows a system configuration.

Figure 2–1: System configuration



(1) Execution environment

This is the environment you need in order to use the application programs that access the distributed KVS, as well as the distributed KVS itself.

You must use a reliable network. To improve reliability, we recommend that you use dual networks between EADS servers.

(a) EADS server

EADS server refers to a server process that manages data consisting of keys and values.

The following is the program product used to configure EADS servers:

- Hitachi Elastic Application Data Store

(b) Cluster

Normally, a system consists of multiple EADS servers. A group of EADS servers is called a cluster. EADS clients recognize the cluster as a single unit of storage.

A cluster consists of a group of EADS servers that have the same multicast address and port number within the same segment.

Important note

The number of EADS servers that make up the cluster must be at least the data multiplicity (the number of redundant copies of data plus the original) × 2 - 1.

For details about the number of redundant copies of data, see *2.8 Creating redundant copies of data*.

(c) EADS client

EADS clients refer to user programs that use client libraries provided by EADS to connect to EADS servers.

There are two types of EADS clients, corresponding to the programming languages used to create application programs (Java or C):

- EADS client (Java)
  Hitachi Elastic Application Data Store Client for Java
- EADS client (C)
  Hitachi Elastic Application Data Store Client for C

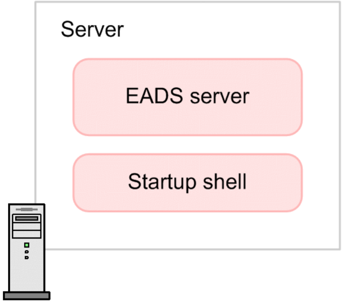
Important note

EADS does not support a system configuration that involves conversion of IP addresses or port numbers for communication between EADS clients and EADS servers.

2.1.2 Configuration of processes

A server with EADS deployed consists of the processes shown in the following figure.

Figure 2–2: Configuration of processes



(1) EADS server

An EADS server refers to a server process that manages data consisting of keys and values.

(2) Startup shell

The startup shell starts the EADS server.

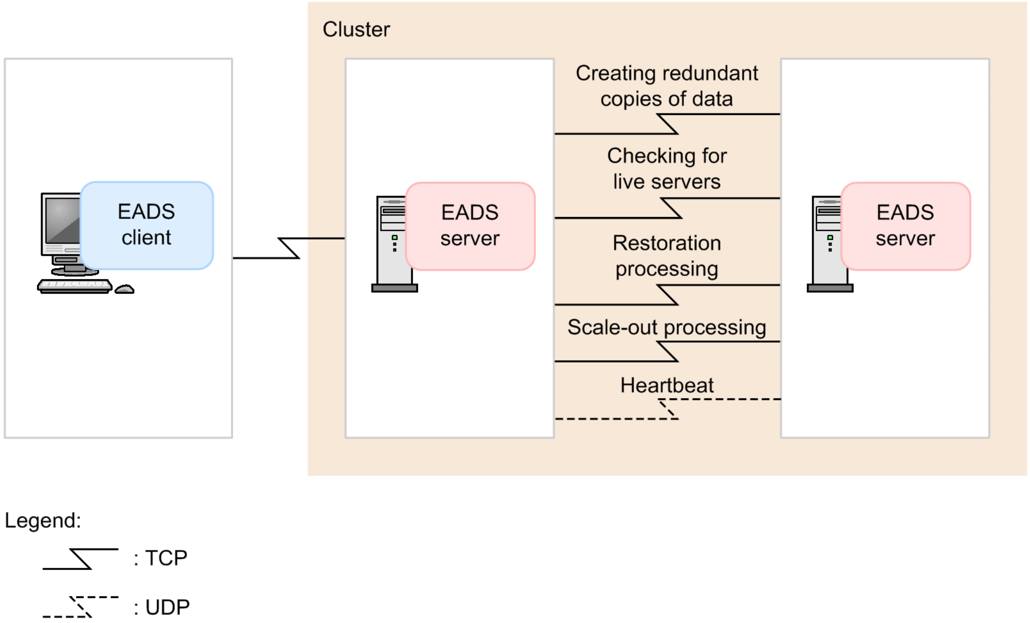
2.2 Mechanisms of EADS communication processing

This section explains the mechanisms of EADS communication processing.

2.2.1 Protocols used for communication

The following figure shows the protocols used for communication between EADS clients and EADS servers, and among EADS servers.

Figure 2–3: Protocols used for communication



(1) Communication between an EADS client and an EADS server

Communication between EADS clients and EADS servers is performed using TCP (Transmission Control Protocol).

(2) Communication between EADS servers

Communication between EADS servers is performed using TCP and UDP (User Datagram Protocol).

TCP is used for the following communication between EADS servers:

- Creating redundant copies of data
- Checking for live servers#
- Restoration processing
- Scale-out processing (adding EADS servers)

UDP is used for the following communication between EADS servers:

- Heartbeat#

#

Heartbeats are multicast within a cluster.

Heartbeats consist of packets distributed in a cluster to report that the EADS servers are running normally.

A check conducted by one EADS server to determine whether another EADS server that is not sending heartbeats has gone down is called a check for live servers.

For details, see *2.9 Monitoring a cluster*.

2.2.2 Buffer used for communication

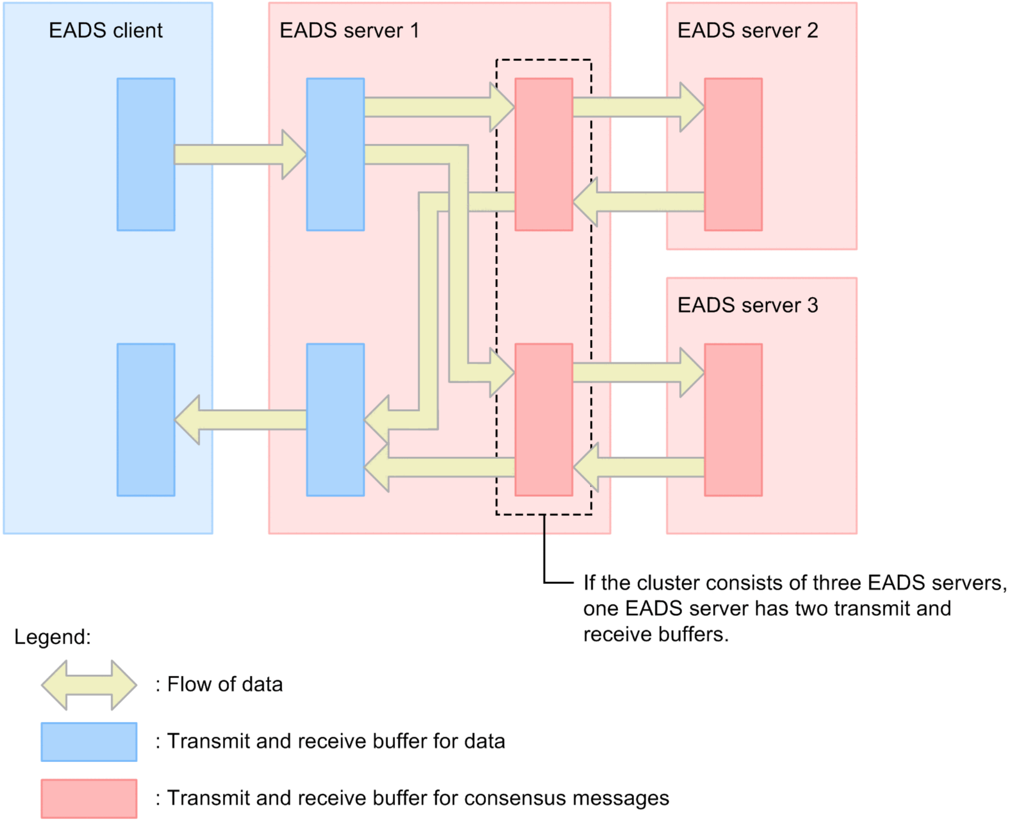
The following types of buffers are used to send and receive data during communication between an EADS client and an EADS server, as well as between EADS servers that use TCP:

- Buffer for transmitting and receiving data
- Buffer for transmitting and receiving consensus messages

You can improve communication efficiency by adjusting the buffer size according to the volume of data to be handled. For details, see *9.1.2 Specifying the buffer size*.

The following figure shows the buffers used for communication.

Figure 2–4: Overview of the buffers used for communication


EADS client
EADS server 1
EADS server 2
EADS server 3
Legend:
: Flow of data
: Transmit and receive buffer for data
: Transmit and receive buffer for consensus messages

If the cluster consists of three EADS servers, one EADS server has two transmit and receive buffers.

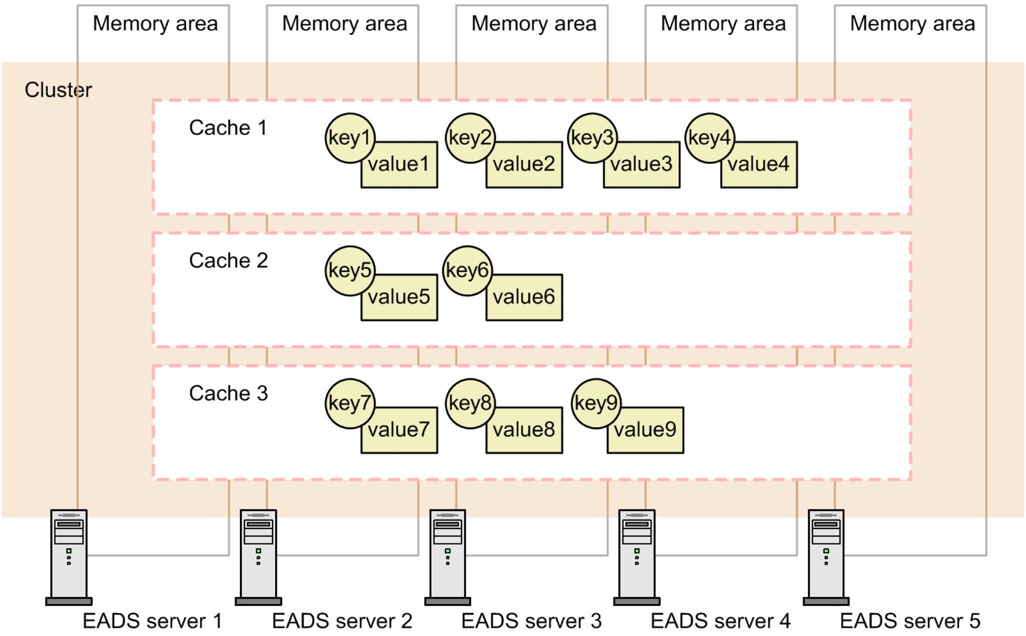
For details about consensus messages, see *2.4.9 General procedure for data access*.

2.3 Areas storing keys and values

Keys and values are stored in an area called a *cache*.

A cache is a logical data storage area that is created across multiple EADS servers. A group of EADS servers that share a cache and make up a single logical KVS is called a *cluster*.

Figure 2–5: Overview of caches



To create caches, use the `eztool createcache` command. You can create a maximum of 16 caches in a cluster.

For details about how to create caches, see *10.2 Starting the EADS servers (and creating caches)* or *11.5 Adding and deleting caches*.

2.3.1 Cache types

EADS supports three types of caches, as explained below. You use the cache type that is appropriate to your operation methods.

Table 2–1: Cache types and their characteristics

No.	Cache type	Data storage	Characteristics
1	Memory cache	Memory area	• Data can be referenced and updated at high speed. • Data loss might occur in the event of a failure because data is managed in memory.
2	Disk cache	Disk area	• Data can be made persistent by using files. • Data referencing and data updating create disk access overhead. • A volume of data greater in size than the size of the physical memory can be stored. • If redundant copies of data are created, data can be restored even when more EADS servers fail than there are data copies.
3	Two-way cache	Memory area and disk area	• Data can be referenced at high speed.

No.	Cache type	Data storage	Characteristics
			• Data can be made persistent by using files. • Data updating creates disk access overhead. • If redundant copies of data are created, data can be restored even when more EADS servers fail than there are data copies.

2.4 Data access

The following are the types of cache data operations that are available:

Data update operations

- put (storing data)
- create (storing new data)
- update (updating data)
- replace (replacing data)
- remove (deleting data)

Data reference operation

- get (acquiring data)

Depending on the type of data operation, you can perform batch operations on multiple data items in a cache.

You can use API functions and commands to manipulate data in a cache. The following table shows the use of the API functions and the commands for data operations.

Table 2–2: Use of the API functions and commands for data operations

Data operation	API function	Command
Data update operations	• put • create • update • replace • remove	• eztool put command • eztool remove command
Data update operations (batch operation)	• putAll • removeAll	eztool removeall command
Data reference operation	get	eztool get command
Data reference operation (batch operation)	getAll	There is no applicable command.

Important note

In terms of the data types and sizes that can be specified, there is not complete compatibility between the API functions and the commands.

Differences in data update operations

The data update operation you can perform depends on whether you use an API function (put, create, update, replace, putAll) or the eztool put command. The following table shows the data operations supported by the API functions and the command:

API function or command	Storing new data	Updating existing data
• Put • put All • eztool put command	Y	Y
create	Y	N

API function or command	Storing new data	Updating existing data
`update`	N	Y
`replace`	N	Y#

Legend:

Y: Can be executed.

N: Cannot be executed.

#

Replaces only values that match the specified values.

2.4.1 Storing data (using put)

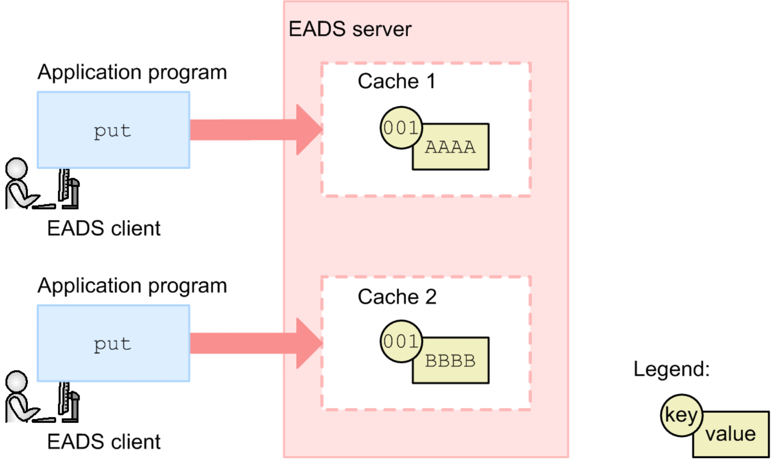
You use `put` to store data in a cache.

First, a key is associated with a value, and then the key-value pair is stored (using `put`). If the specified key already exists in the cache, the value is updated (using `put`) unconditionally.

For the key, specify a value that is unique in the cache. Data with a duplicated key can be stored if the storage cache name is different.

The following figure shows an example in which data having the same key is stored in caches 1 and 2.

Figure 2–6: Overview of storing data (using put)



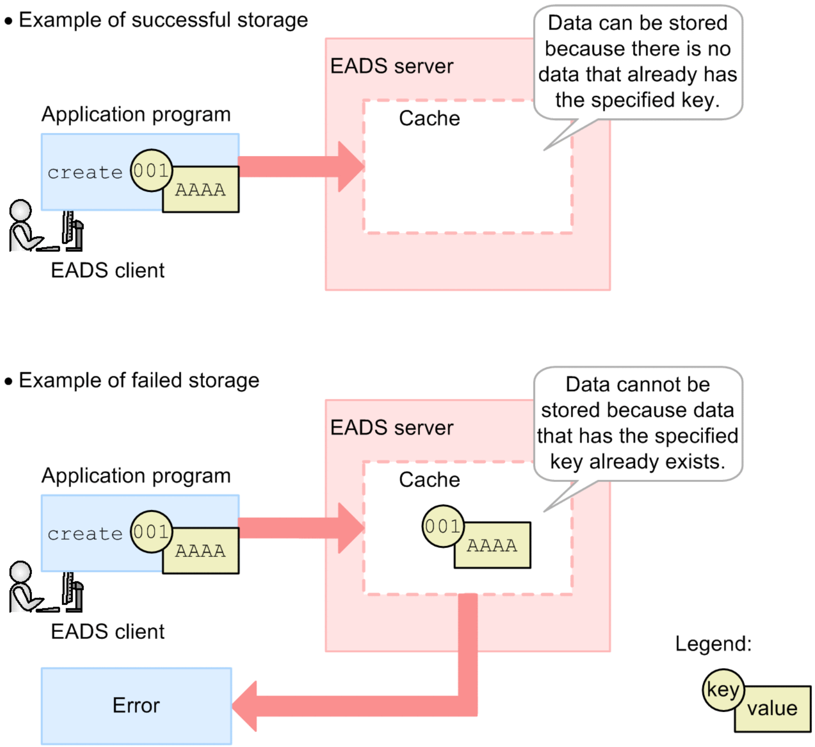
2.4.2 Storing new data (using create)

You use `create` to store new data in a cache.

Only when a new key is stored, a key is associated with a value, and then the key-value pair is stored (using `create`).

`create` only stores new data. If a specified key already exists in the cache, an error results.

Figure 2–7: Overview of storing new data (using create)



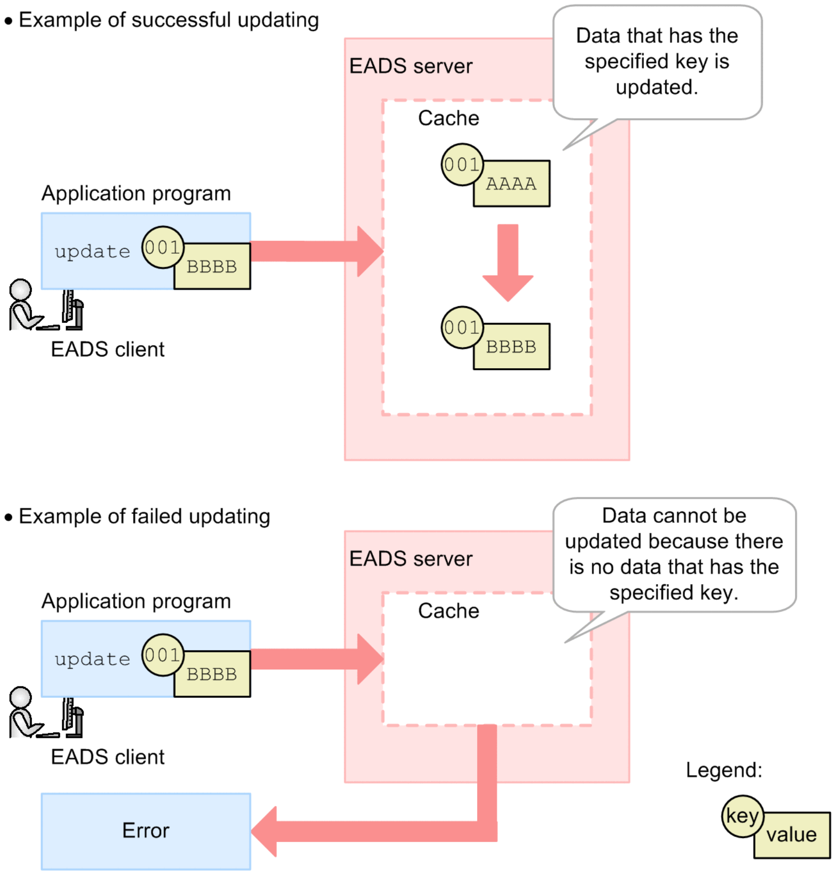
• Example of successful storage
EADS server
Data can be stored because there is no data that already has the specified key.
Application program
Cache
create
001
AAAA
EADS client
• Example of failed storage
EADS server
Data cannot be stored because data that has the specified key already exists.
Application program
Cache
create
001
AAAA
001
AAAA
EADS client
Error
Legend:
key
value

2.4.3 Updating data (using update)

You use `update` to update the data stored in a cache.

Only if the specified key is stored in a cache, `update` associates a value with the key, and then updates the data. If the specified key is not stored in a cache, an error results.

Figure 2–8: Overview of updating data (using update)


• Example of successful updating
EADS server
Data that has the specified key is updated.
Cache
001
AAAA
Application program
update
001
BBBB
001
BBBB
EADS client
• Example of failed updating
EADS server
Data cannot be updated because there is no data that has the specified key.
Application program
Cache
update
001
BBBB
EADS client
Error
Legend:
key
value

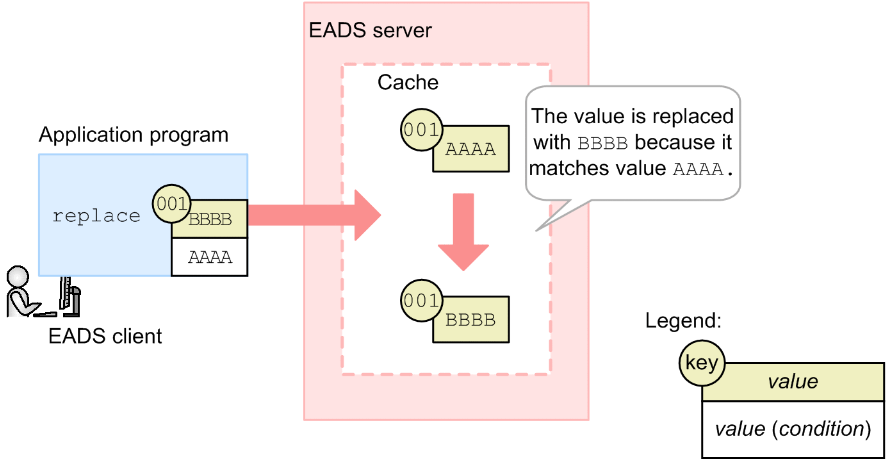
2.4.4 Replacing data (using replace)

You use `replace` to replace data in a cache.

While `put` unconditionally updates values, `replace` replaces values only when they match the specified values.

The following figure shows an example in which a value whose key is `001` is replaced with `BBBB`.

Figure 2–9: Overview of replacing data (using replace)



This example replaces the value with `BBBB` because it matches the specified value (`AAAA`).

Reference note

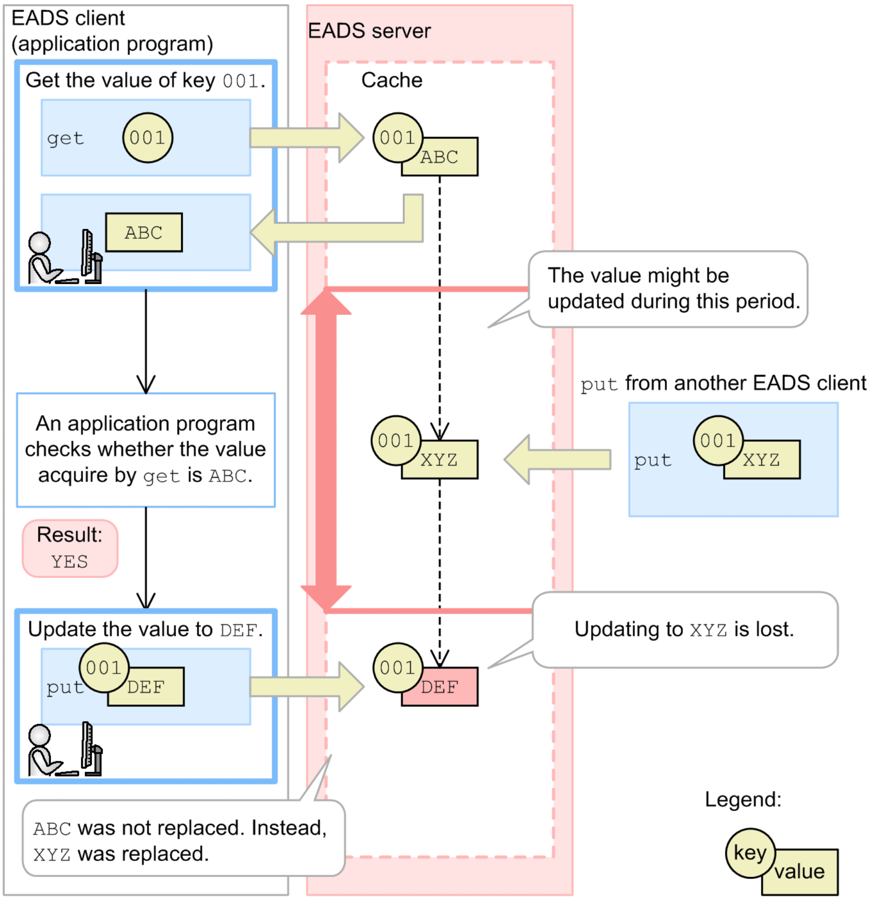
Difference between put and replace

To check the contents of a stored value and then update the value, you might use get to obtain a value, an application program checks the value, and then you execute put. However, the value might be updated by another application program between the get and put processes. You can avoid this problem by using replace.

The following figure shows the difference between put and replace by means of an example that replaces ABC with DEF.

Figure 2–10: Difference between put and replace

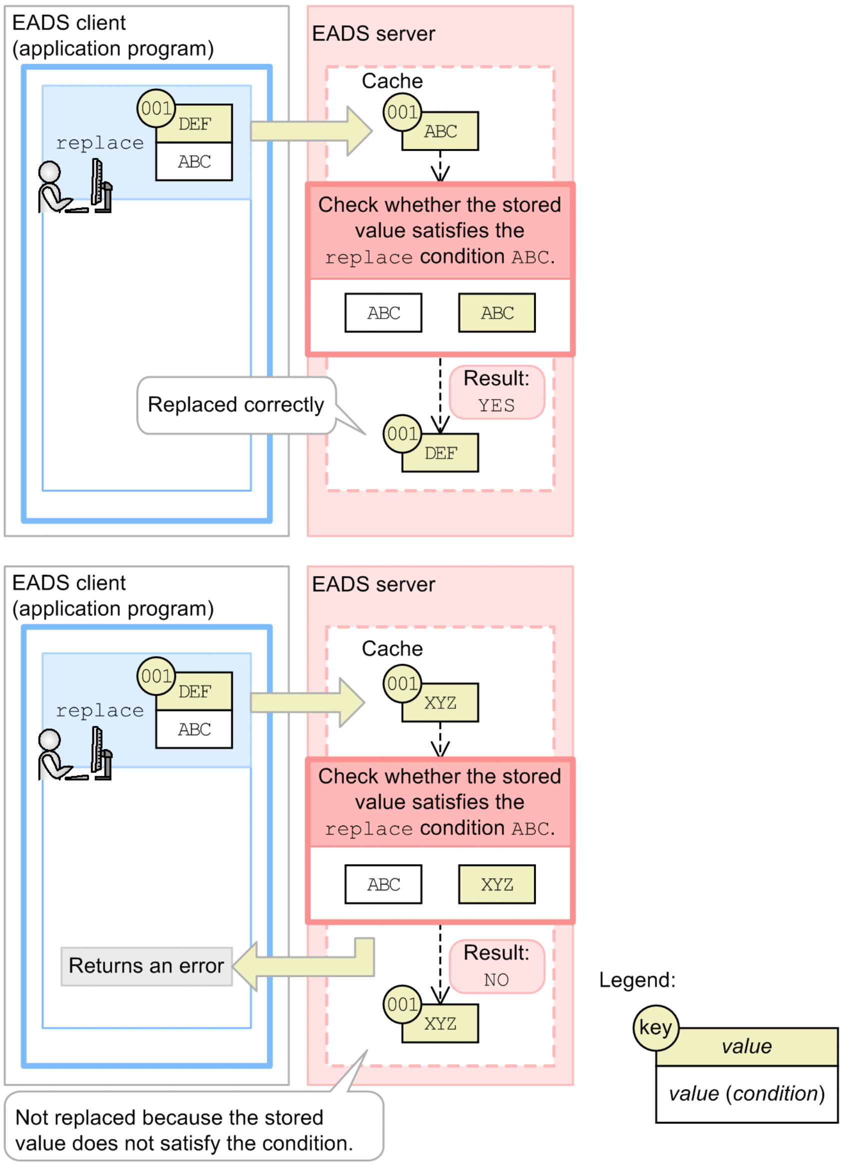
- Replacing ABC with DEF (using get and put)


EADS client
(application program)
EADS server
Cache
Get the value of key 001.
get
001
001
ABC
ABC
The value might be updated during this period.
put from another EADS client
An application program checks whether the value acquire by get is ABC.
001
XYZ
put
001
XYZ
Result:
YES
Update the value to DEF.
001
put
DEF
001
DEF
Updating to XYZ is lost.
ABC was not replaced. Instead, XYZ was replaced.
Legend:
key
value

Explanation:

If get and put are used for replace processing, data consistency might not be maintained because an interrupt might occur during the processing, as in this example.

- Replacing ABC with DEF (using replace)



Explanation:

replace can perform replace processing while maintaining data consistency.

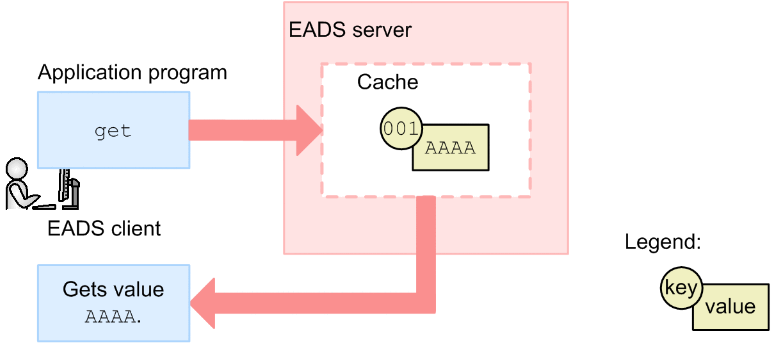
2.4.5 Acquiring data (using get)

You use get to acquire data from a cache.

get specifies a key associated with the value to be acquired, and then acquires (gets) the value.

The following figure shows an example of getting the value whose key is 001.

Figure 2–11: Overview of acquiring data (using get)



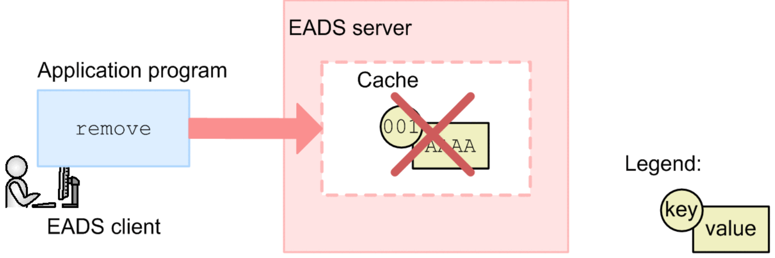
2.4.6 Deleting data (remove)

You use `remove` to delete data from a cache.

`remove` specifies a key associated with the value to be deleted, and then deletes (removes) the key and value.

The following figure shows an example of deleting data whose key is `001`.

Figure 2–12: Overview of deleting data (using remove)

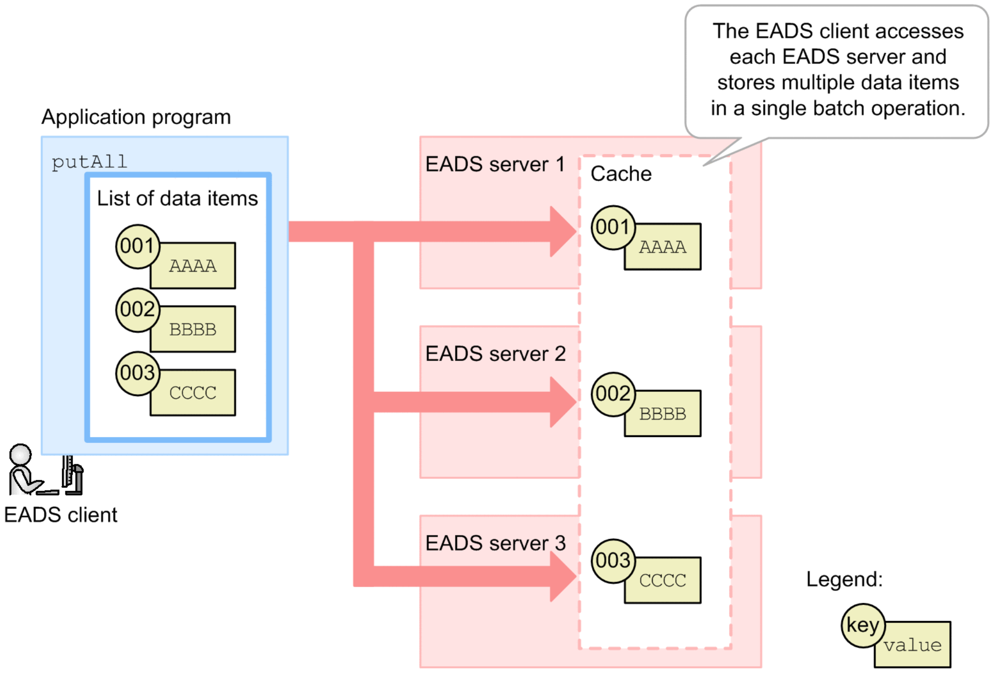


2.4.7 Performing batch operations on a cache

You can perform batch operations on multiple data items in a cache. The following types of batch operations are supported:

- Batch data storage (`putAll`)
- Batch data acquisition (`getAll`)
- Batch data deletion (`removeAll`)

Figure 2–13: Overview of batch operations performed on a cache (example of putAll)



You can use the following methods to specify the data that is to be subject to a batch operation:

- Specify multiple keys or data items in a single batch operation.
- Specify the name of the group to which the target keys belong.
- Specify the EADS server at the data storage destination.

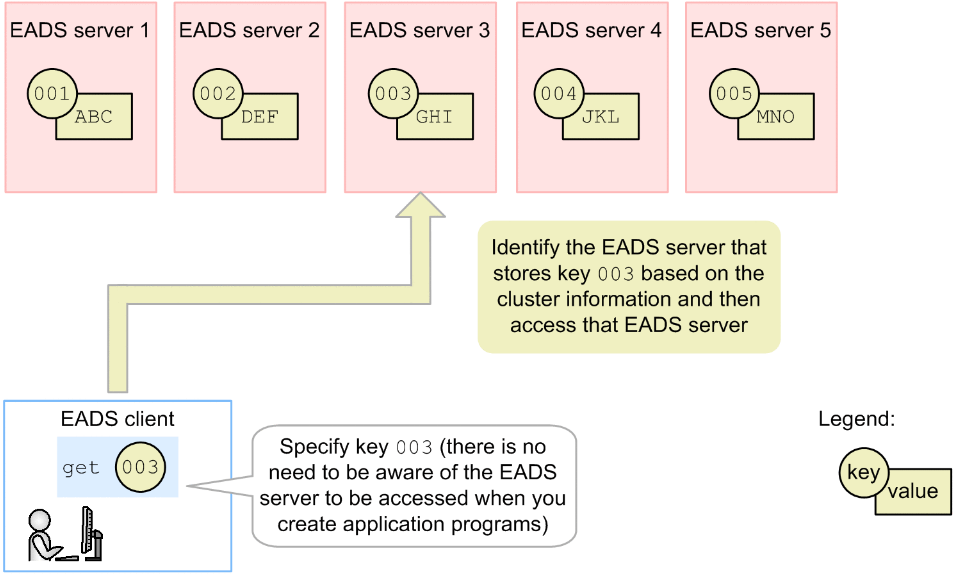
For details about group names (grouping keys), see *2.6 Placing data on a specific EADS server (grouping keys)*.

2.4.8 Data access without having to be aware of the locations of individual EADS servers

EADS uses an algorithm called *consistent hashing* to distribute data to the EADS servers in the cluster.

To access distributed data, an EADS client identifies the EADS server that stores the data based on connection target information (cluster information) that the EADS client maintains. Therefore, when you create application programs, there is no need to know the physical locations of the EADS servers that store data.

Figure 2–14: Overview of accessing EADS servers


EADS server 1
001
ABC
EADS server 2
002
DEF
EADS server 3
003
GHI
EADS server 4
004
JKL
EADS server 5
005
MNO
Identify the EADS server that stores key 003 based on the cluster information and then access that EADS server
EADS client
get
003
Specify key 003 (there is no need to be aware of the EADS server to be accessed when you create application programs)
Legend:
key
value

If there are redundant copies of the data, an EADS server storing the data is accessed. For details about how redundant copies of data are created, see *2.8 Creating redundant copies of data*.

For an overview of how cluster information is monitored, see *2.9.1 Overview of monitoring a cluster by sending heartbeats* .

For an overview of data distribution, see *2.5 Data distribution by consistent hashing*.

2.4.9 General procedure for data access

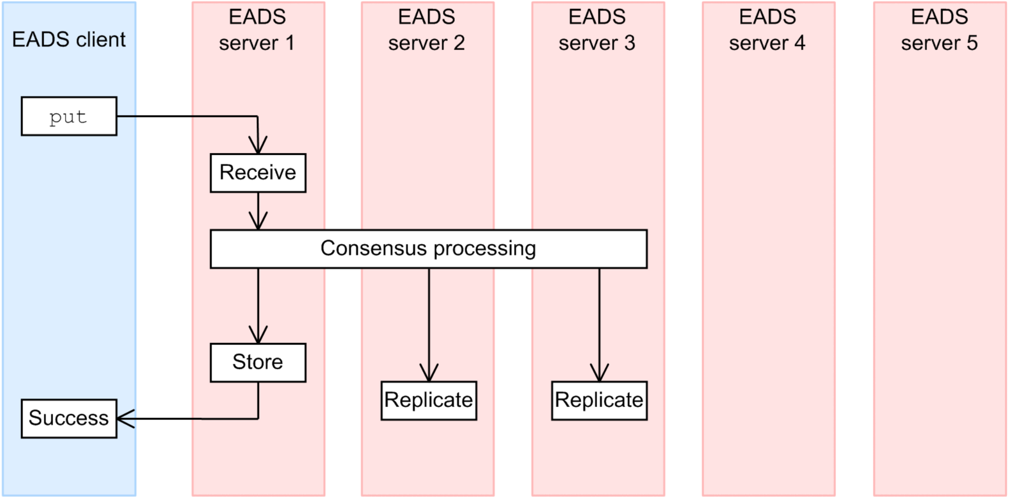
This subsection explains the general procedure for data access.

(1) Data update operation

The EADS client identifies the EADS server that stores the data based on the cluster information that the EADS client maintains and then updates the data.

The following figure shows the general procedure for data access by means of an example of `put` processing that sets the multiplicity to `3`.

Figure 2–15: General procedure for data access



The EADS client sends a `put` processing request.

When EADS server 1 receives the request, it sends a consensus message to the EADS servers to which the data is to be copied to obtain consensus for performing the `put` processing.

EADS server 1 performs the `put` processing if it receives as many consensus messages from individual EADS servers as there are data copies plus the original. This insures data consistency when redundant copies of the data are created. This example needs consensus from three EADS servers to perform `put` processing because multiplicity has been set to `3`.

If consensus processing is not completed within a specified period of time (the default is 0.8 second), a timeout occurs and the consensus processing is performed again.

If the EADS server receives three consensus responses including one from itself, it stores and creates redundant copies of the data. Because the multiplicity is set to `3`, this example copies data to EADS servers 2 and 3. The processing is performed asynchronously at the EADS servers.

When the data has been stored in EADS server 1, which is the EADS server that received the request, the processing results are returned to the EADS client.

The following explains the handling of data storage and redundancy errors.

The possible causes of a data storage or redundancy error are as follows:

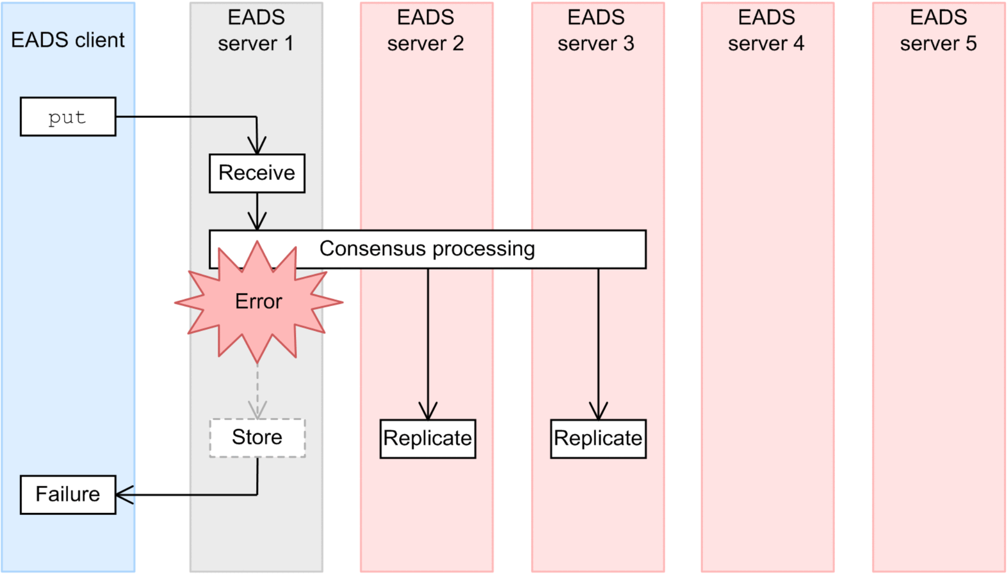
- The connection-target EADS server or network has failed.
- The area for storing the value part of a key-value pair is inadequate.#
- There is more data than an EADS server can store.#

#

When the total data restriction function is enabled and a shortage of data storage capacity is foreseen, an error in the corresponding processing can be set and the EADS server can be prevented from becoming isolated. For this reason, we recommend that you enable the total data restriction function.

(a) When data storage fails

This subsection explains the handling of data storage errors by means of an example of `put` processing that sets multiplicity to `3`.



In this example, data storage in EADS server 1 failed and an error is returned to the EADS client.

Redundant copies of the data are created.

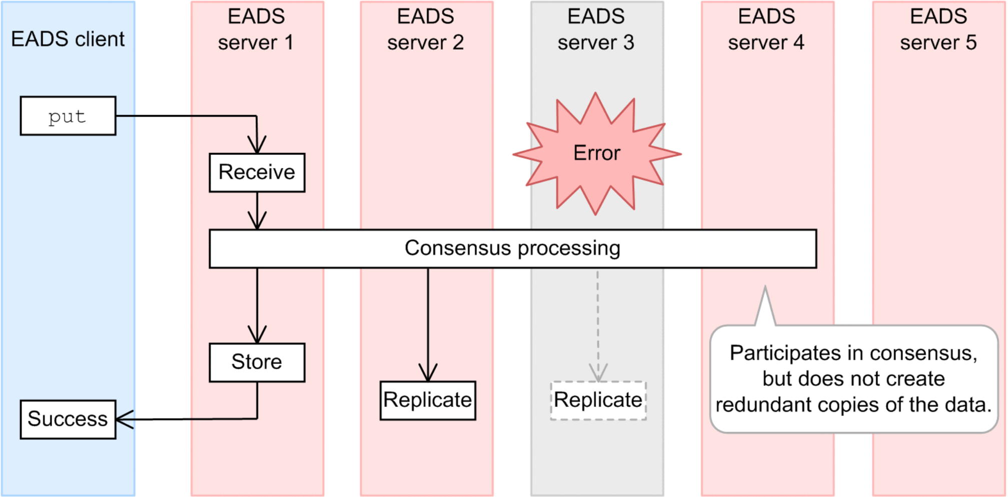
The EADS server where storage of the data failed is isolated,# but the cluster continues operating. In this status, the data multiplicity remains low. To restore the data multiplicity, eliminate the problem that isolated the EADS server and then restore the EADS server.

#

An isolated EADS server no longer accepts requests from EADS clients.

(b) When data storage is successful, but creation of redundant copies of the data fails

This subsection explains the handling when data storage was successful, but creation of redundant copies of the data fails, by way of an example of `put` processing that sets multiplicity to `3`.



To perform `put` processing, this example requires consensus from three EADS servers. However, there is no response from EADS server 3, one of the targets to which the data is to be copied. The example obtains consensus from another EADS server (EADS server 4), and then performs the `put` processing. Note that EADS server 4 does not create redundant copies of the data.

The EADS server in which creation of a redundant copy of the data failed is isolated, but the cluster continues operating. In this status, the data multiplicity remains low. To restore the data multiplicity, eliminate the problem that isolated the EADS server and then restore the EADS server.

(2) Data reference operation

The EADS client identifies the EADS server where the desired data was stored based on the cluster information that the EADS client maintains, and then references the data.

If the target EADS server in which the data was stored is down but redundant copies of the data had been created, the EADS client accesses an EADS server containing a copy of the data and references the data there.

2.4.10 Locking during data access

If data were to be manipulated by multiple EADS clients and commands simultaneously, data consistency might be lost. To prevent this, EADS servers lock data in units of ranges during update operations.

Range refers to data storage areas in which data in a cache is separated according to the location of the EADS servers. For details, see *2.5.1 Overview of data distribution*.

The following table shows whether the same range of data can be accessed by multiple EADS clients and commands simultaneously.

Table 2–3: Whether the same range of data can be accessed by multiple EADS clients and commands simultaneously

Processing underway	Processing to be executed simultaneously	
	Data update operation#1	Data reference operation#2
Data update operation#1	N	Y
Data reference operation#2	Y	Y

Legend:

Y: Can be performed simultaneously. The processing is performed without waiting for completion of the current processing.

N: Cannot be performed simultaneously. The processing is performed after the current processing has been completed.

#1

Data update operation means the following API functions or commands:

- API functions (`put`, `putAll`, `create`, `update`, `replace`, `remove`, `removeAll`)
- `eztool put` command
- `eztool remove` command
- `eztool removeall` command

The API functions include API functions executed within user functions.

#2

Data reference operation means the following API functions or command:

- API functions (`get`, `getAll`)
- The following `Iterator` methods of the `Group` interface:
  - `keyIterator()`

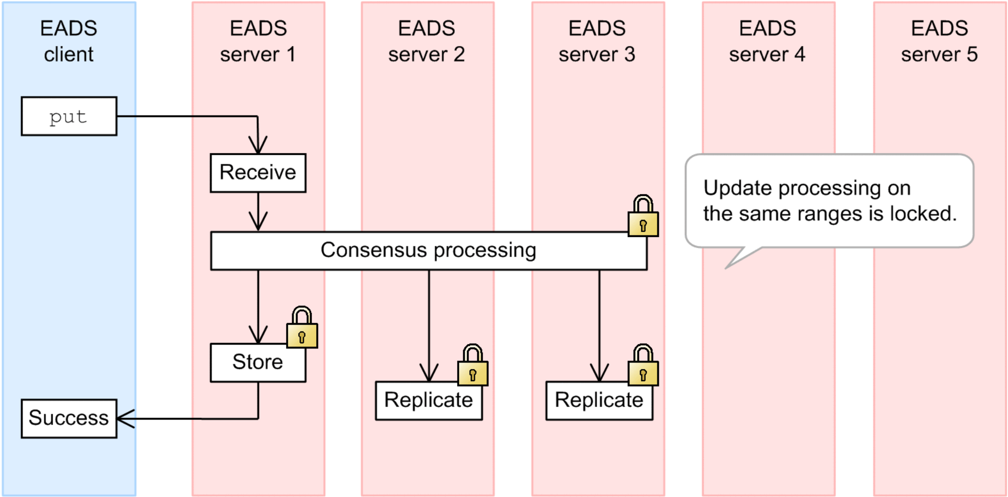
- descendingKeyIterator()
- higherKeyIterator()
- lowerDescendingKeyIterator()

- eztool get command

The API functions include API functions executed within user functions.

The following figure shows the scope of locking, using put as an example.

Figure 2–16: Scope of locking



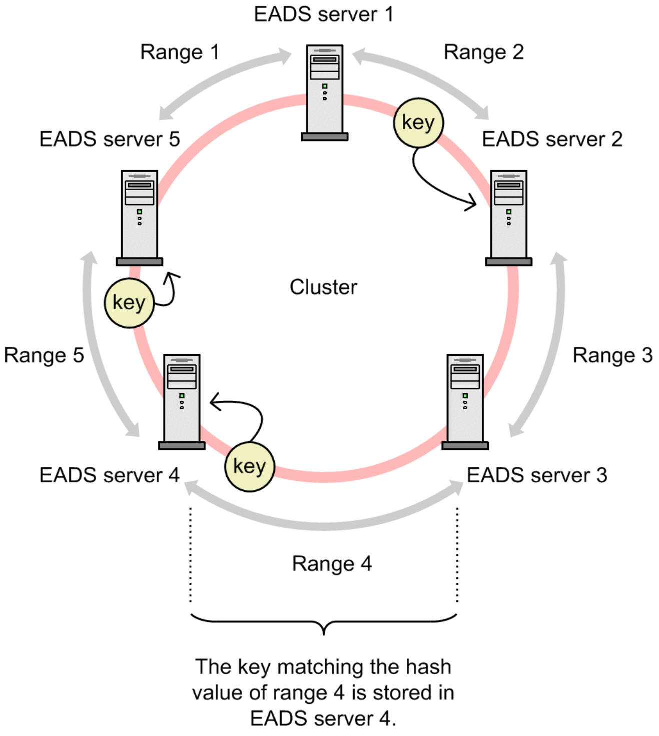
2.5 Data distribution by consistent hashing

EADS uses an algorithm called *consistent hashing* to distribute data to the EADS servers in the cluster.

2.5.1 Overview of data distribution

This subsection provides an overview of data distribution.

Figure 2–17: Data distribution by consistent hashing



In consistent hashing, the servers and keys are treated as being placed on the same circumference. Sequential integers are assigned in ascending order to this circumference in a counterclockwise direction.

In this figure, first, the hash values of the EADS servers (from 1 through 5) and the keys for the data to be stored are obtained. Then, the servers and keys are placed on the circumference according to their hash values.

The key for each data item is then stored on the first EADS server located clockwise from the location at which the key was placed.

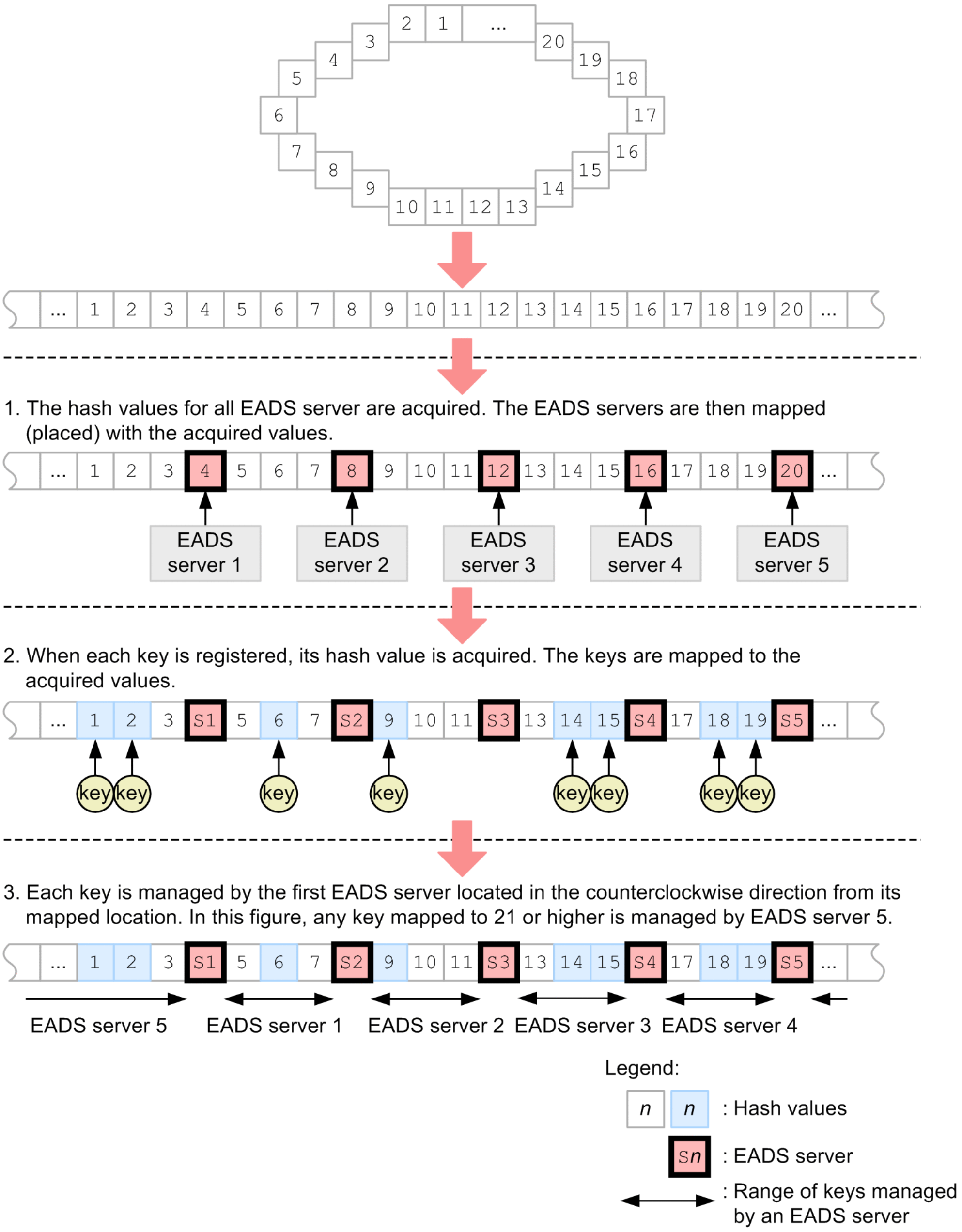
A range of hash values obtained by separating the consistent hashing circumference in a cache by the location of each EADS server is called a *range*. Ranges are managed by range IDs. In this figure, a key placed in range 4 is stored in EADS server 4.

This circle has nothing to do with the physical placement of the EADS servers.

2.5.2 Details about data distribution

This subsection provides details about data distribution.

In consistent hashing, servers and keys are treated as being placed on a circumference. However, in this example, they are represented on a straight line for convenience.



1. The hash values for all EADS server are acquired. The EADS servers are then mapped (placed) with the acquired values.

2. When each key is registered, its hash value is acquired. The keys are mapped to the acquired values.

3. Each key is managed by the first EADS server located in the counterclockwise direction from its mapped location. In this figure, any key mapped to 21 or higher is managed by EADS server 5.

2.5.3 Adding EADS servers (scaling out) and distributing data

EADS enables you to add new EADS servers to a cluster without having to stop the cluster. This is called *scale-out processing*. You can add an EADS server at any desired location on the consistent hashing circumference.

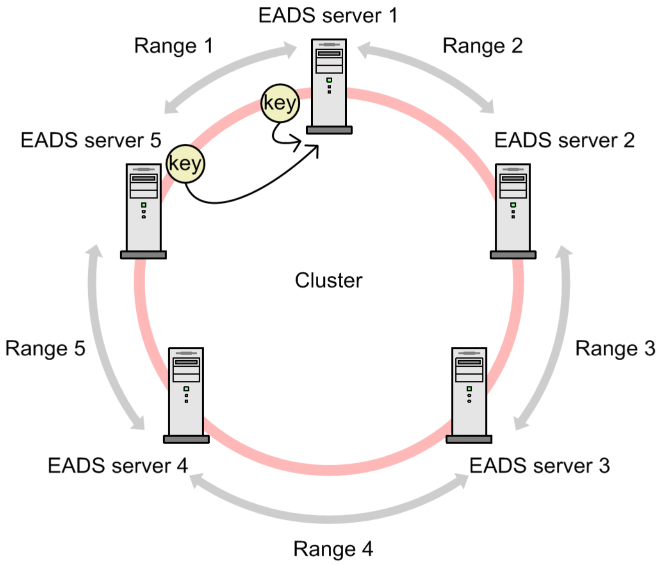
When scale-out processing is performed, the range within which a new EADS server is added is divided. The range containing values that are greater than the value at the location of the added EADS server (hash value) becomes the new range. The EADS server that had been managing the previous range transfers the newly added range of data to the added EADS server.

Adding new EADS servers to a cluster enables you to increase the overall size of the physical memory and improve performance in the entire cluster. By dividing a range, you can reduce the memory usage by the EADS server that had been managing the previous range and reduce its workload.

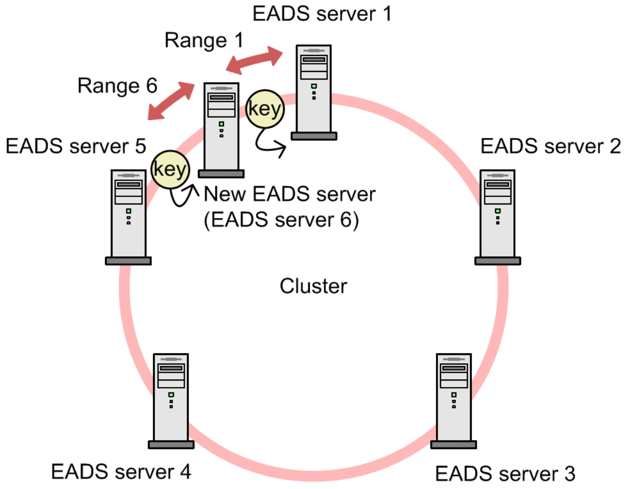
The following figure provides an overview of adding one new EADS server to a cluster.

Figure 2–18: Overview of scale-out processing (adding an EADS server)

■ Before adding a new EADS server



■ After adding a new EADS server (after scale-out processing)



2.6 Placing data on a specific EADS server (grouping keys)

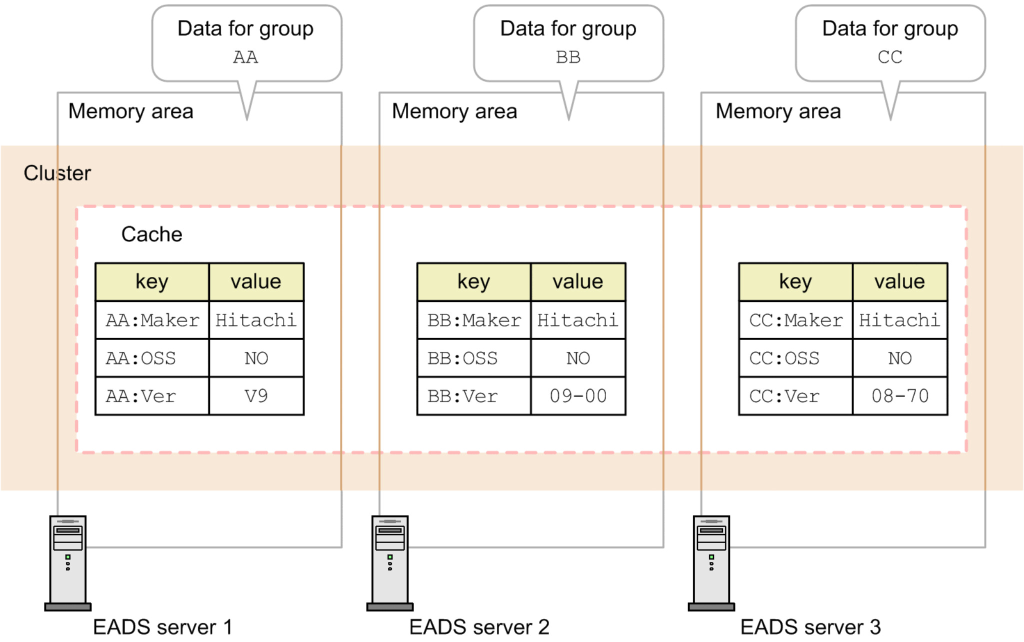
EADS allows you to group multiple keys together so that related data resides on the same EADS server. This is called grouping keys.

If keys are not grouped, the data is distributed among the EADS servers in the cluster. When data needs to be manipulated for purposes such as obtaining totals, data distributed throughout the cluster must be accessed. Communication is required each time data is acquired from each EADS server.

If you know the keys that will need to be processed and group those keys together, you can place related data on the same EADS server. This enables processing using user functions to be performed on groups, thus improving the efficiency of data processing.

For details about user functions, see *2.7 Efficient data processing using user functions*.

Figure 2–19: Placing data on a specific EADS server (grouping keys)



key	value
AA:Maker	Hitachi
AA:OSS	NO
AA:Ver	V9

key	value
BB:Maker	Hitachi
BB:OSS	NO
BB:Ver	09-00

key	value
CC:Maker	Hitachi
CC:OSS	NO
CC:Ver	08-70

2.6.1 Grouping keys

You group keys by assigning a group name to the keys. This subsection explains group names, element names, and group hierarchies.

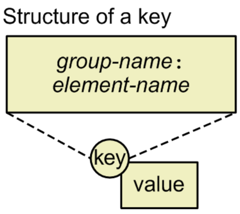
(1) Group names and element names

You group keys by specifying a group name for the keys.

Define a key in the format *group-name* **:** *element-name*. Separate the group name and the element name with a colon (**:**).

The following figure shows the structure of keys when keys are grouped.

Figure 2–20: Structure of keys



Group name

This is a name of a group of specific keys.

Element name

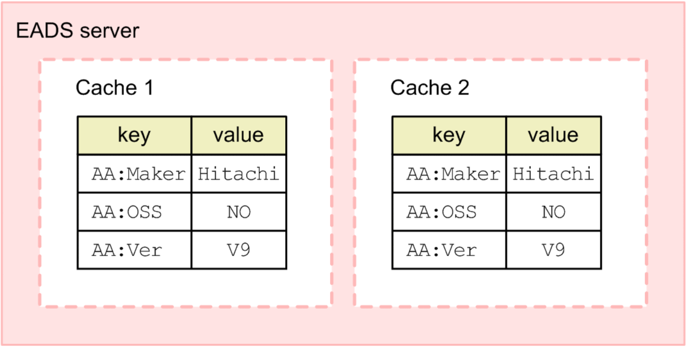
This is a name used to uniquely identify a value.

If keys are not grouped, this becomes the key.

(2) Defining group and element names

Each key must be unique within the cache. Therefore, multiple keys that have the same group and element names cannot be defined in the same cache.

As shown in the following figure, keys with identical group and element names can be defined as long as their storage caches are different:



key	value
AA:Maker	Hitachi
AA:OSS	NO
AA:Ver	V9

key	value
AA:Maker	Hitachi
AA:OSS	NO
AA:Ver	V9

(3) Group hierarchies

You can arrange groups hierarchically by defining multiple group names, such as `Group1:Group2:Group3:ElementA`. The name of each group is called the group hierarchy name.

By using group hierarchies, you can perform more detailed data manipulation involving a specific hierarchy, such as totaling the data belonging to `Group2` in the data in `Group1`.

When groups are configured in a hierarchy, a group name consists of the highest group hierarchy name through the corresponding group hierarchy name. For example, in `Group1:Group2:Group3:ElementA`, group hierarchy names are `Group1`, `Group2`, and `Group3`, and group names are `Group1`, `Group1:Group2`, and `Group1:Group2:Group3`.

2.6.2 Data distribution with grouped keys

If a data item has a defined group name, its storage EADS server is determined by the hash value of that group name.

If groups have hierarchies, the storage EADS server is determined by the hash value of the first group hierarchy name. Multiple data items with the same first group hierarchy name are stored on the same EADS server because their hash values are the same.

For example, the data items with the following keys are stored on the same EADS server:

- `Group1:Group2:ElementA`
- `Group1:Group3:ElementB`
- `Group1:ElementC`

For an overview of data distribution, see *2.5 Data distribution by consistent hashing*.

> **Tip**
>
> You can also group keys by specifying the EADS server ID of the storage EADS server. Such a group is called an EADS server ID specified group. If you use EADS server ID specified groups to group keys, specify the EADS server ID enclosed in square brackets (`[]`) at the beginning of the keys (at the beginning of the highest group hierarchy name).
>
> For example, the key data shown below belongs to group `[1]Group1` and is stored in the EADS server whose EADS server ID is `1` (any character string following the EADS server ID, such as `Group1`, is ignored and the storage EADS server is determined only by the specified EADS server ID).
>
> - `[1]Group1:Group2:ElementA`
> - `[1]Group1:Group3:ElementB`
> - `[1]Group1:ElementC`

2.7 Efficient data processing using user functions

A user function is a program that defines a series of data operations (user processing) on specific data in a cache, such as totaling and analyzing data.

The user can first create a user function and place it on EADS servers, and then execute that user function by calling it from the EADS client.

A user function accesses only the data stored on the EADS server on which the user function was executed. Therefore, you can reduce communication overhead and achieve efficient data processing by using user functions instead of by using the EADS client.

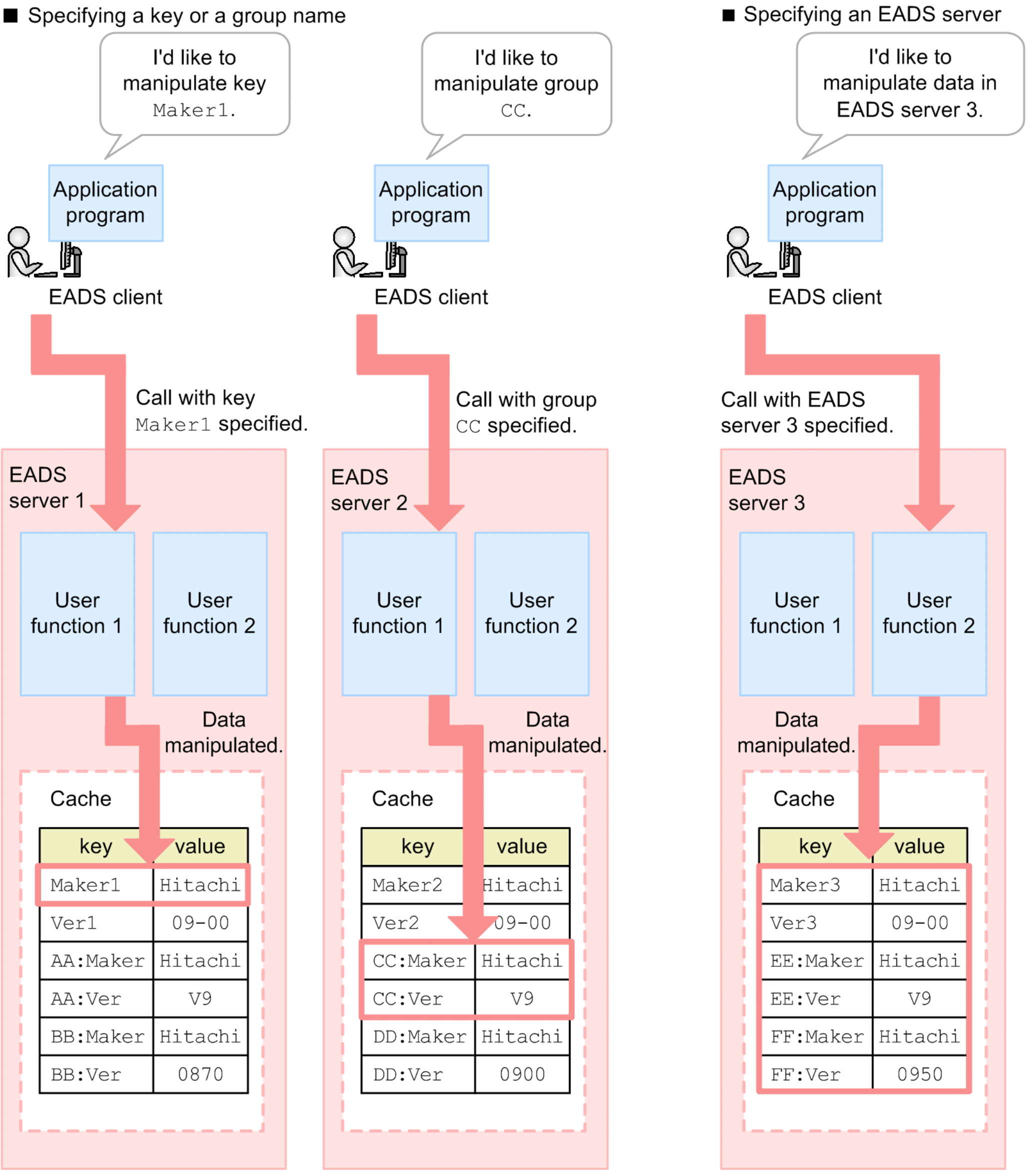
Use Java to create user functions.

2.7.1 Mechanism of user functions

There are two ways to execute user functions:

- By specifying a key or a group
- By specifying an EADS server

Figure 2–21: Mechanism of user functions


■ Specifying a key or a group name
I'd like to manipulate key Maker1.
I'd like to manipulate group CC.
Application program
EADS client
Call with key Maker1 specified.
Call with group CC specified.
EADS server 1
EADS server 2
User function 1
User function 2
Data manipulated.
Cache
■ Specifying an EADS server
I'd like to manipulate data in EADS server 3.
Application program
EADS client
Call with EADS server 3 specified.
EADS server 3
User function 1
User function 2
Data manipulated.
Cache

key	value
Maker1	Hitachi
Ver1	09-00
AA:Maker	Hitachi
AA:Ver	V9
BB:Maker	Hitachi
BB:Ver	0870

key	value
Maker2	Hitachi
Ver2	09-00
CC:Maker	Hitachi
CC:Ver	V9
DD:Maker	Hitachi
DD:Ver	0900

key	value
Maker3	Hitachi
Ver3	09-00
EE:Maker	Hitachi
EE:Ver	V9
FF:Maker	Hitachi
FF:Ver	0950

To use user functions, first create them and then place them on all EADS servers in the cluster.

The following explains the mechanism of user functions for each execution method.

(1) Execution by specifying a key or a group

Call the user function from the EADS client by specifying the key or group name and user function name. The EADS server to be accessed is determined from the hash value of the specified key or group name.

(2) Execution by specifying an EADS server

Call the user function from the EADS client by specifying the EADS server and user function name. The EADS server to be accessed is determined from the specified EADS server's address information (IP address and port number).

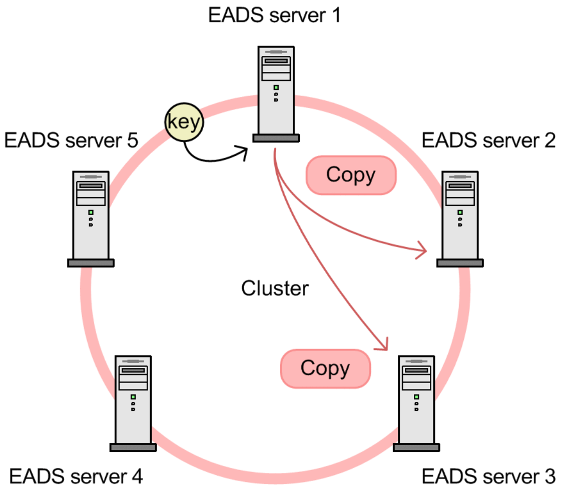
2.8 Creating redundant copies of data

EADS can create redundant copies of data by automatically copying data to multiple EADS servers. If one of the EADS servers fails, you can continue processing by using the data copied to the other EADS servers. By having redundant copies of data, you can also avoid losing data from memory in the event of a failure.

The following provides an overview of how EADS creates redundant copies of data.

2.8.1 Overview of creating redundant copies of data

Figure 2–22: Overview of how EADS creates redundant copies of data



EADS copies data to multiple EADS servers in a clockwise direction according to the specified multiplicity (the number of data copies plus the original).

As shown in this figure, if the multiplicity is 3 and the data storage location is EADS server 1, EADS first stores data on EADS server 1. EADS then copies data from EADS server 1 to EADS server 2 and EADS server 3 in a clockwise direction.

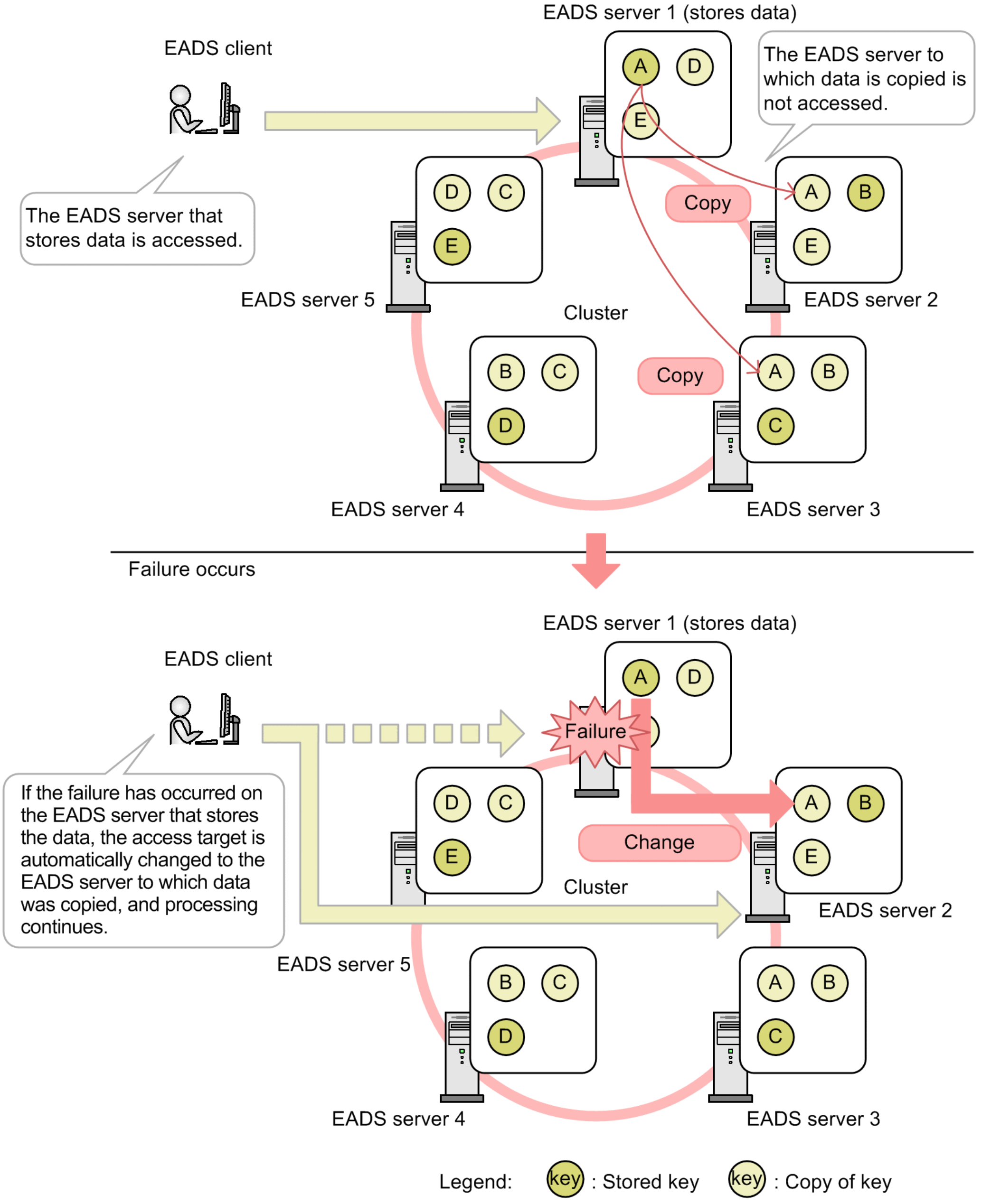
2.8.2 Continuing processing in the event of a failure when redundant copies of data have been created

When redundant copies of data have been created, the EADS client accesses the EADS server that stores the target data. The EADS client does not access the EADS servers where copies of the data are stored.

If a failure prevents the EADS client from accessing the EADS server that stores the data, it automatically changes the access target to an EADS server that stores a copy of the data and continues processing.

The following figure shows how data is accessed when EADS server 1 stores the data and how the access target is changed when a failure occurs.

Figure 2–23: Data access when redundant copies of data have been created and changing the access target in the event of a failure


EADS server 1 (stores data)
EADS client
The EADS server to which data is copied is not accessed.
The EADS server that stores data is accessed.
A
D
E
D
C
E
A
B
E
B
C
D
A
B
C
Copy
Copy
EADS server 5
EADS server 2
Cluster
EADS server 4
EADS server 3
Failure occurs
EADS server 1 (stores data)
EADS client
Failure
Change
If the failure has occurred on the EADS server that stores the data, the access target is automatically changed to the EADS server to which data was copied, and processing continues.
Cluster
EADS server 5
EADS server 2
EADS server 4
EADS server 3
Legend:
key : Stored key
key : Copy of key

Important note

The number of EADS servers that make up a cluster must be at least the data multiplicity × 2 - 1.

Data consistency is maintained as long as the number of EADS servers that have shut down due to failures is less than the number of data copies plus the original.

For example, if the data multiplicity is 3 and there are five EADS servers and two of the EADS servers are shut down, no data loss occurs.

2.9 Monitoring a cluster

The EADS servers in a cluster send heartbeats (packets distributed within the cluster) to notify each other that they are running normally. If there is an EADS server that does not send heartbeats, the other EADS servers in the cluster check whether that EADS server is alive. This is called cluster monitoring.

With cluster monitoring, the user can tune the speed at which errors are detected, by specifying settings such as the heartbeat interval.

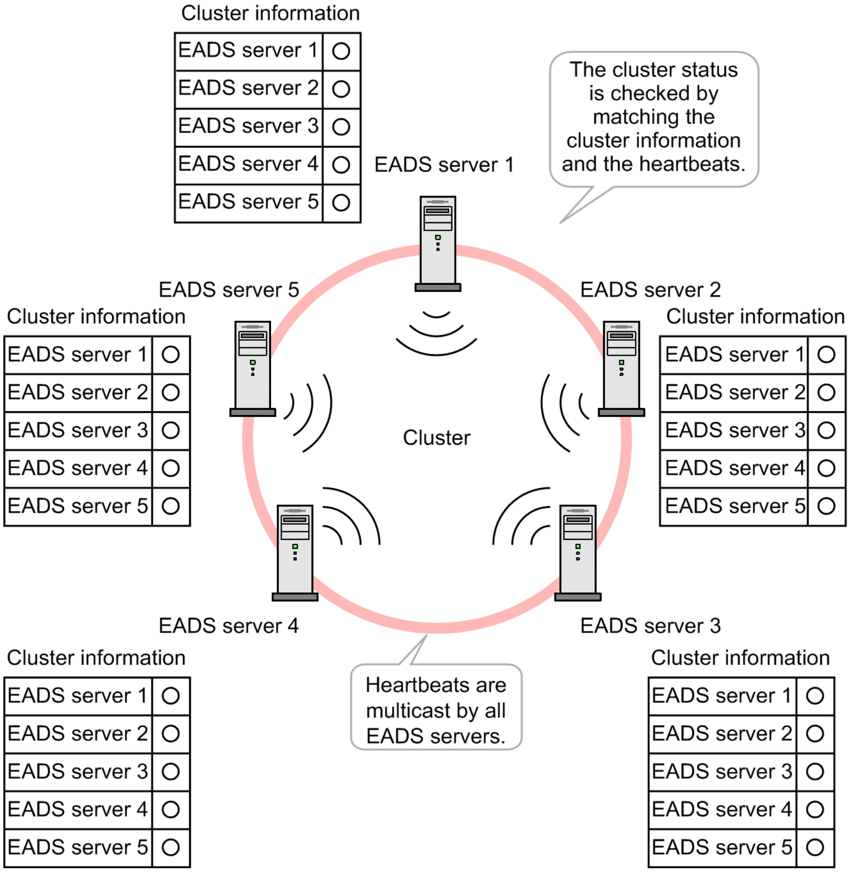
2.9.1 Overview of monitoring a cluster by sending heartbeats

The EADS servers in a cluster mutually send a heartbeat at a predefined interval (the default is every 0.4 seconds).

In a cluster, the EADS servers check the cluster status by comparing the cluster information managed by individual EADS servers with the multicast heartbeats.

The cluster information provides information about that EADS servers' state of life or death within the cluster.

Figure 2–24: Overview of monitoring a cluster



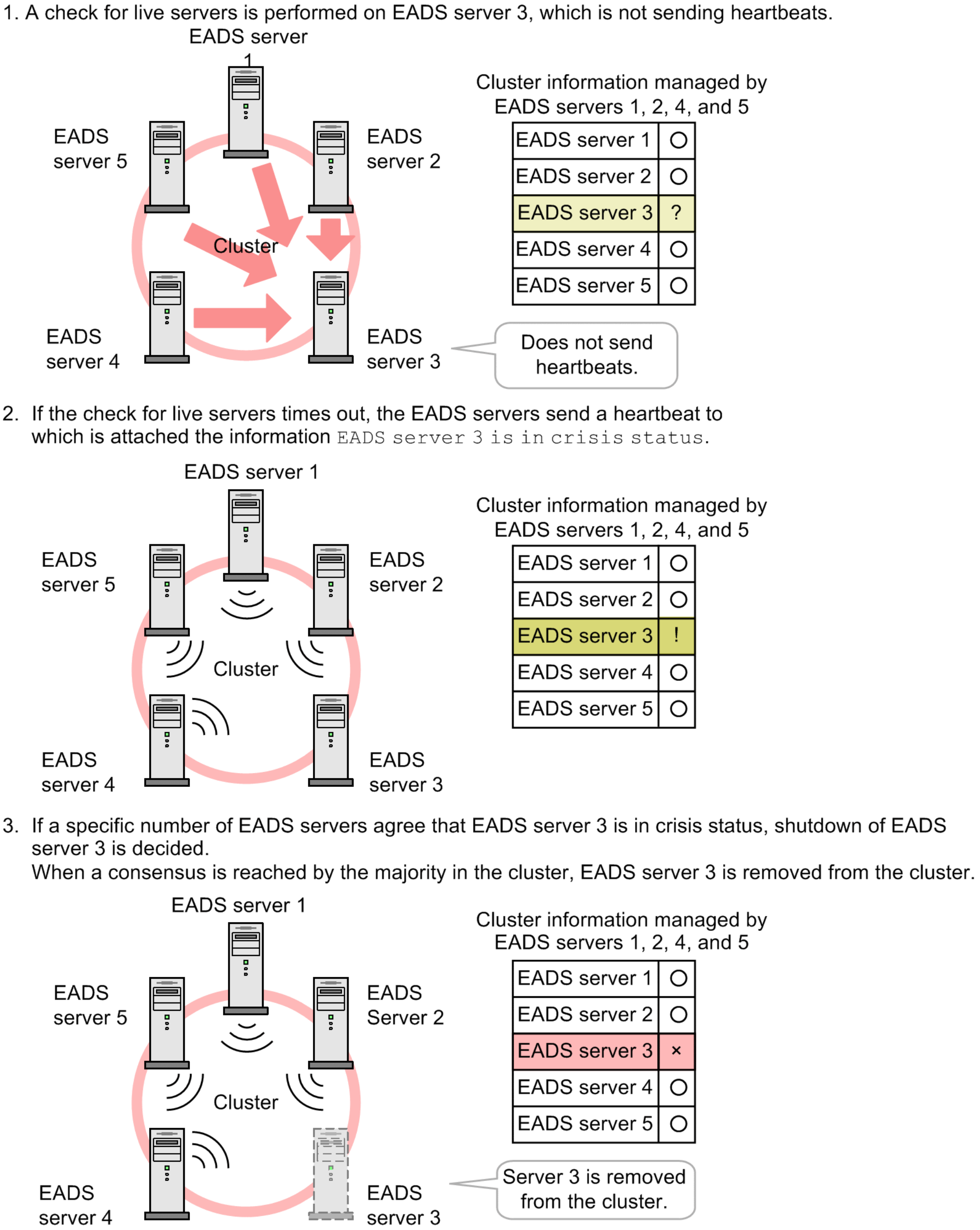
2.9.2 EADS server shutdown decided by the agreement of a specific number of EADS servers

If there is an EADS server that does not send a heartbeat within the specified timeout (the default is two seconds), the other EADS servers in the cluster check whether that EADS server is alive (using check for live servers). If the check for live servers times out, shutdown of the EADS server is decided by the agreement of a specific number of EADS servers (the default is one EADS server). After the server shutdown is decided, the corresponding EADS server is removed from the cluster.

Because the cluster is always monitored by heartbeat transmission from EADS servers, an EADS server shutdown can be detected at an early stage.

The following figure shows the flow of detecting an EADS server shutdown.

Figure 2–25: Flow of detecting an EADS server shutdown



1. A check for live servers is performed on EADS server 3, which is not sending heartbeats.

Cluster information managed by EADS servers 1, 2, 4, and 5

EADS server 1	○
EADS server 2	○
EADS server 3	?
EADS server 4	○
EADS server 5	○

Does not send heartbeats.

2. If the check for live servers times out, the EADS servers send a heartbeat to which is attached the information `EADS server 3 is in crisis status`.

Cluster information managed by EADS servers 1, 2, 4, and 5

EADS server 1	○
EADS server 2	○
EADS server 3	!
EADS server 4	○
EADS server 5	○

3. If a specific number of EADS servers agree that EADS server 3 is in crisis status, shutdown of EADS server 3 is decided.
When a consensus is reached by the majority in the cluster, EADS server 3 is removed from the cluster.

Cluster information managed by EADS servers 1, 2, 4, and 5

EADS server 1	○
EADS server 2	○
EADS server 3	×
EADS server 4	○
EADS server 5	○

Server 3 is removed from the cluster.

1. If no heartbeat is sent from EADS server 3, the EADS servers in the cluster perform a check for live servers on EADS server 3.
2. If the check for live servers times out before a heartbeat is sent from EADS server 3, the EADS servers in the cluster mutually send heartbeats with the attached information `EADS server 3 is in crisis status`.
3. If a specific number of EADS servers (the default is one EADS server) agree that EADS server 3 is in crisis status, shutdown of EADS server 3 is decided.

   The EADS servers in the cluster then decide whether EADS server 3 is to be removed from the cluster. If consensus is reached by the majority in the cluster, EADS server 3 is removed from the cluster.

   The removed EADS server is forcibly isolated. An isolated EADS server no longer accepts requests from the EADS client.

Important note

The cluster can continue operating as long as the number of EADS servers that have shut down due to failures is less than the number of data copies plus the original.

For example, if the data multiplicity is 3 and there are five EADS servers and two of the EADS servers are shut down, monitoring of the cluster can continue.

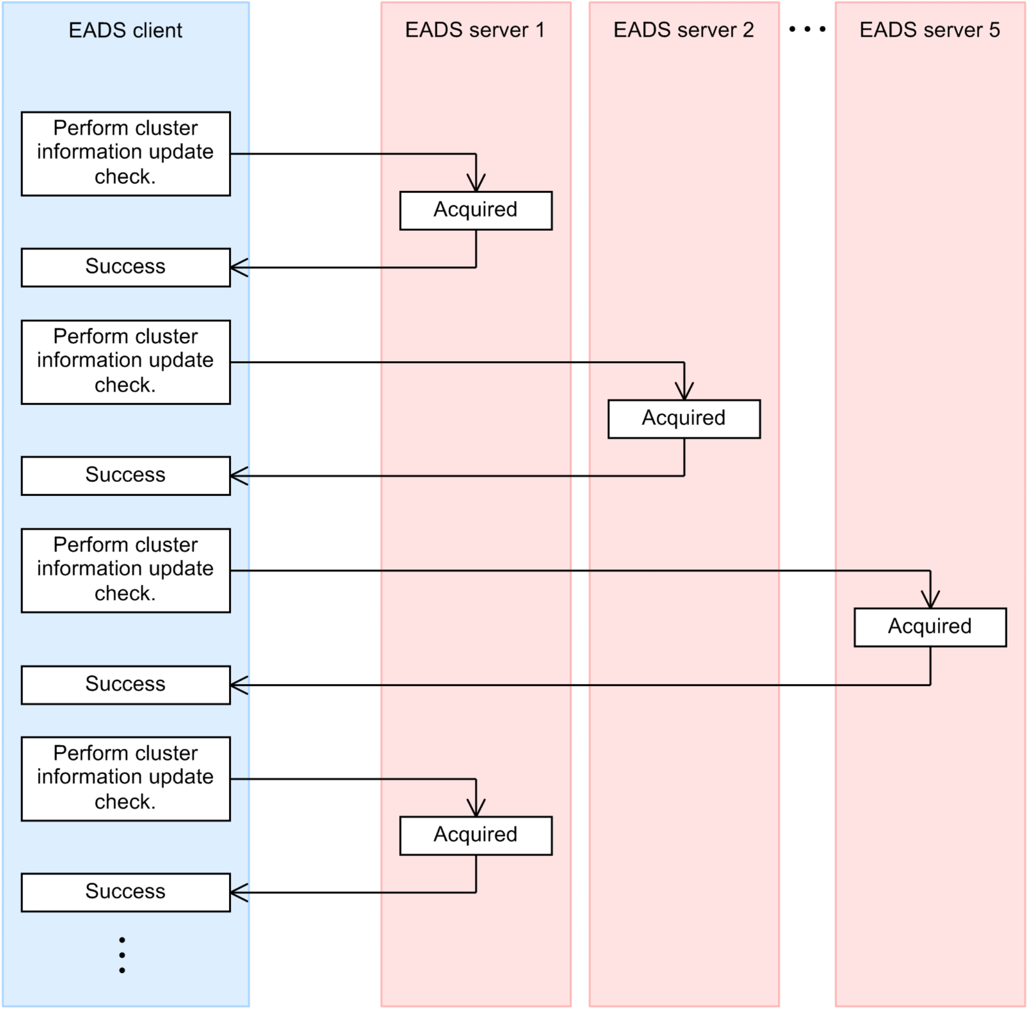
2.10 Cluster information update check by the EADS client

To access data in a cache, the EADS client identifies the EADS server that stores the data based on the cluster information that the EADS client maintains. This means that you do not have to know which EADS server stores the data when you have the EADS client access the data.

From the time the EADS client's initialization is completed to the time the EADS client terminates, it obtains cluster information from each and every EADS server in the cluster at a specified interval (the default is every second). This prevents the EADS client from managing outdated cluster information.

The EADS client obtains cluster information from the EADS servers by accessing the EADS servers in ascending order of their logical locations (hash values) based on the cluster information that the EADS client maintains.

Figure 2–26: Overview of cluster information update check


EADS client
EADS server 1
EADS server 2
• • •
EADS server 5
Perform cluster information update check.
Acquired
Success
Perform cluster information update check.
Acquired
Success
Perform cluster information update check.
Acquired
Success
Perform cluster information update check.
Acquired
Success

An error results when there is no response within the timeout value. Even though an error occurs, the EADS client continues to perform cluster information update checks on all EADS servers.

2.11 Cluster and EADS server status transitions

The available operations (API functions and commands) depend on the status of the cluster and EADS servers.

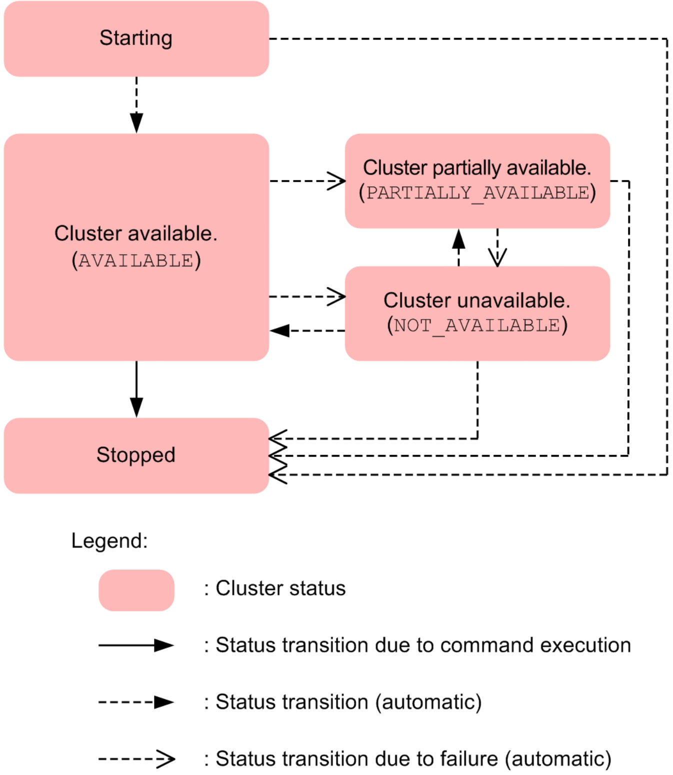
Note the following:

- An application program developer must be knowledgeable about cluster and EADS server statuses when designing application programs.
- The system operation administrator must check the status of the cluster and EADS servers before executing commands. You use the `eztool status` command to check the status of the cluster and EADS servers.

2.11.1 Cluster status transitions

The following figure shows the cluster status transitions.

Figure 2–27: Cluster status transitions


Starting
Cluster available.
(AVAILABLE)
Cluster partially available.
(PARTIALLY_AVAILABLE)
Cluster unavailable.
(NOT_AVAILABLE)
Stopped
Legend:
: Cluster status
: Status transition due to command execution
: Status transition (automatic)
: Status transition due to failure (automatic)

The following table explains each status shown in the figure.

Table 2–4: Cluster status

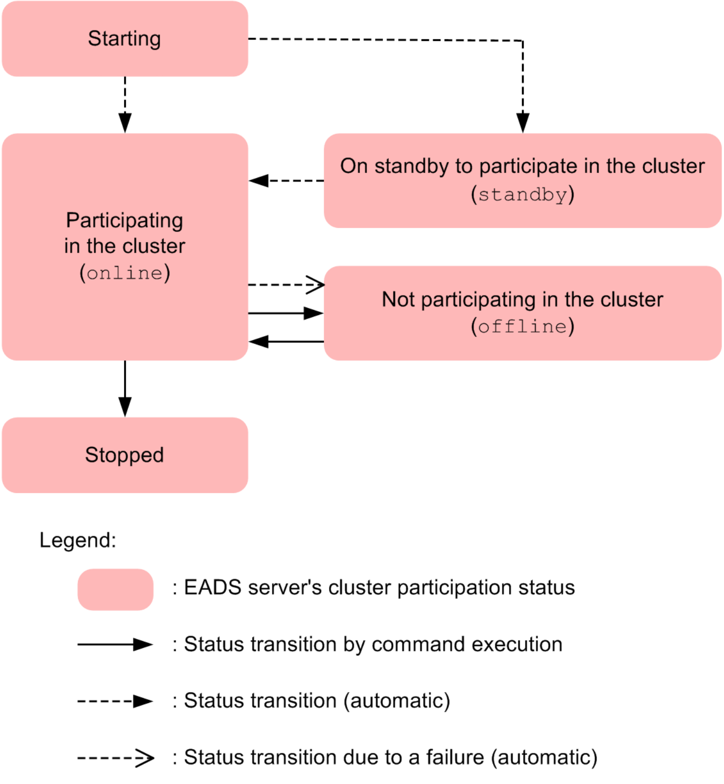
Cluster status		Description
Status	Status displayed by the eztool status command	
Cluster is available	`AVAILABLE`	The cluster is running normally.
Cluster is partially available	`PARTIALLY_AVAILABLE`	The cluster is partially running. The cluster might not be accessible depending on the key.

Cluster status		Description
Status	Status displayed by the eztool status command	
		If the number of EADS servers that make up the cluster is at least the data multiplicity × 2 - 1, the cluster might be placed in this status if more EADS servers are shut down than there are data copies plus the original. Once the cluster is placed in this status, it cannot be placed in `AVAILABLE` (cluster available) status unless all EADS servers are stopped.
Cluster is unavailable	`NOT_AVAILABLE`	The cluster is not running. The cluster is placed in this status in either of the following cases: • Split-brain occurred. • At least half of the EADS servers in the cluster are shut down. Once the cluster is recovered from split-brain, it can be placed in `AVAILABLE` status (cluster available) or `PARTIALLY_AVAILABLE` (cluster partially available).

2.11.2 EADS servers' cluster participation status

The following figure shows the EADS server cluster participation status transitions.

Figure 2–28: Cluster participation status transitions


Starting
On standby to participate in the cluster (standby)
Participating in the cluster (online)
Not participating in the cluster (offline)
Stopped
Legend:
: EADS server's cluster participation status
: Status transition by command execution
: Status transition (automatic)
: Status transition due to a failure (automatic)

The following table explains the statuses shown in the figure.

Table 2–5: EADS server cluster participation statuses

Cluster participation status		Description
Participation status	Status displayed by the eztool status command	
Participating in the cluster	`online`	The EADS server is participating in the active cluster.
Not participating in the cluster	`offline`	The EADS server is not participating in the active cluster. An EADS server that has been deleted from the cluster for a reason such as a communication error is placed in this status. An EADS server in this status might not be available for data access processing or command execution. An EADS server in this status can resume participation in the cluster after its status has been checked and it is recovered.
On standby to participate in the cluster	`standby`	The EADS server has never participated in the active cluster. An EADS server defined in the cluster properties is in this status when it has not started. Once an EADS server in this status is started, it is participating in the cluster.

2.11.3 EADS server status transitions

The following figure shows the EADS server status transitions.

Figure 2–29: EADS server status transitions (normal start)

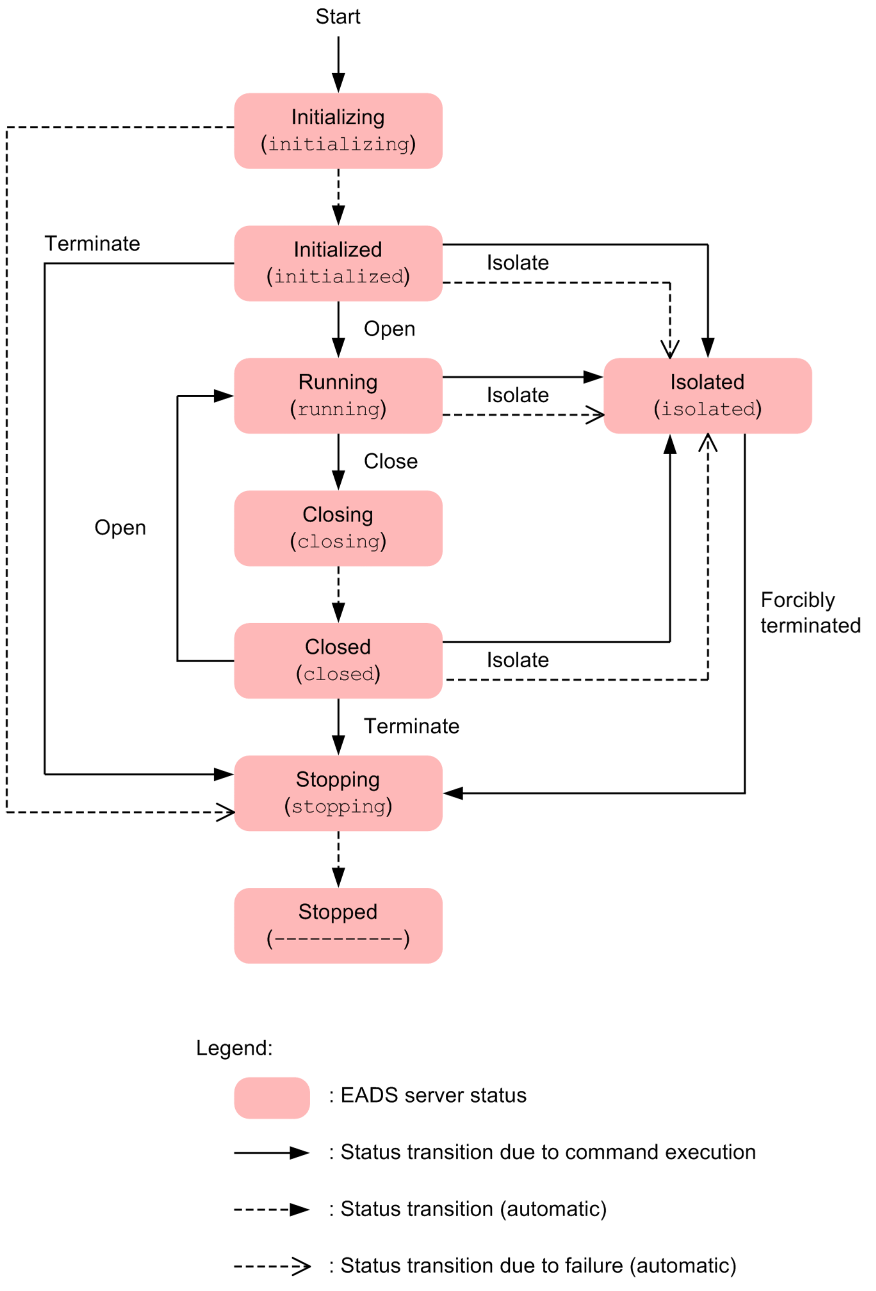
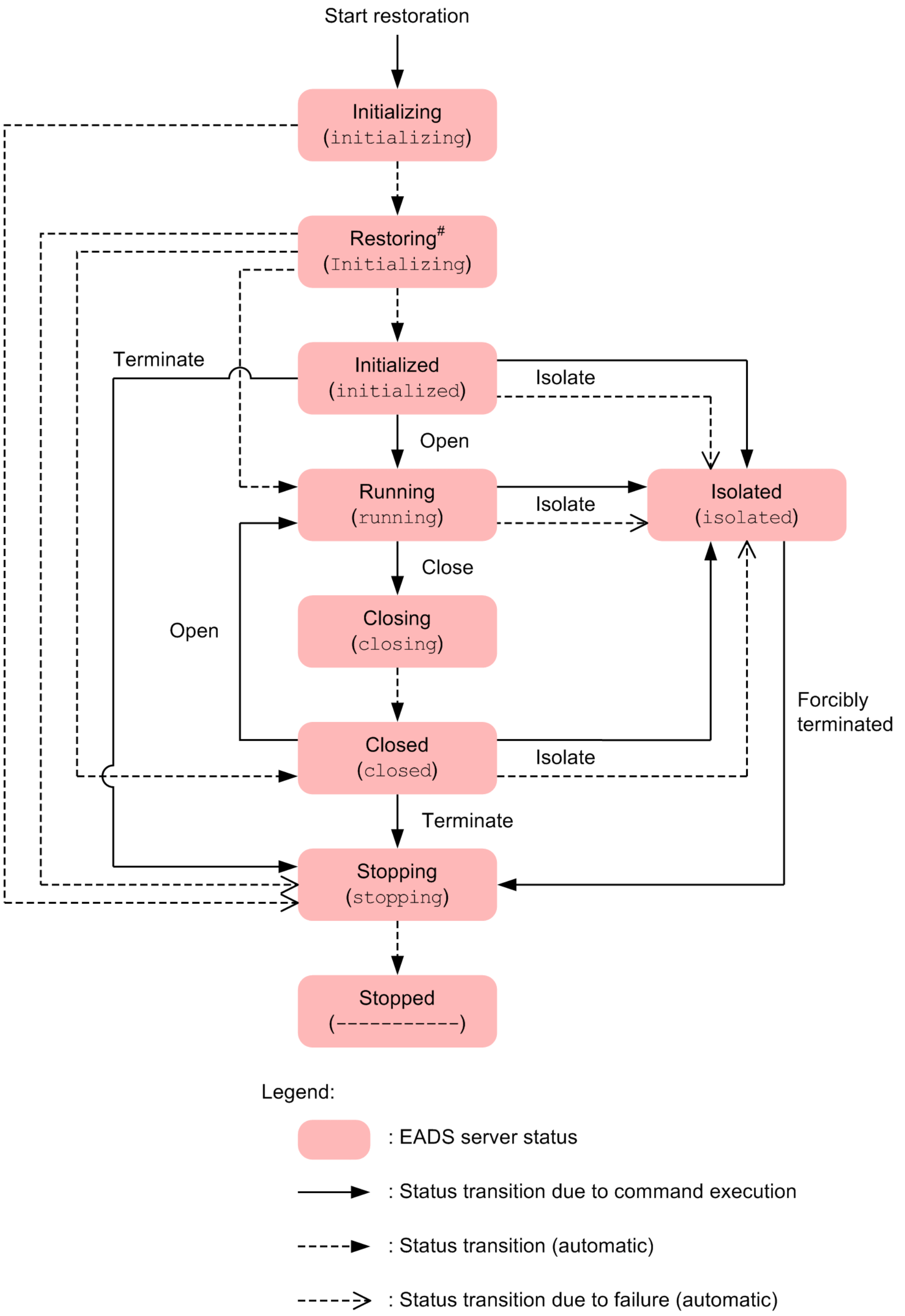


Figure 2–30: EADS server status transitions (restoration processing)

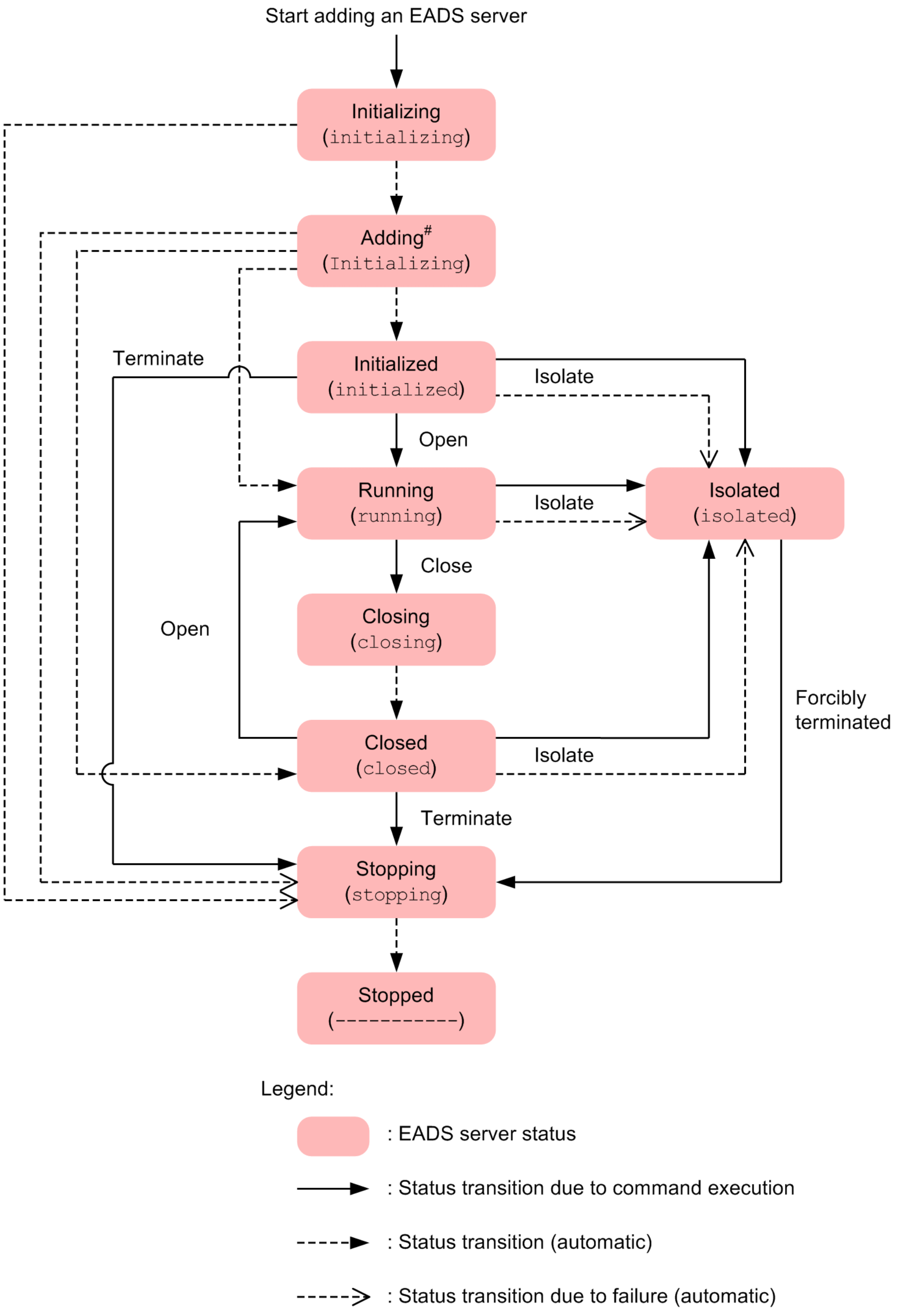


#

If restoration processing is successful, the restored EADS server is placed in the same status as that of the other EADS servers that were already running in the cluster when the restoration processing started.

For example, if the status of the other EADS servers that were running when the restoration processing started was `running`, the restored EADS server is also placed in `running` status.

Figure 2–31: EADS server status transitions (scale-out processing (adding EADS servers))



#

If EADS server addition processing is successful, the added EADS server is placed in the same status as that of the other EADS servers that were already running in the cluster when the EADS server addition processing started.

For example, if the status of the other EADS servers that were running when the EADS server addition processing started was `running`, the added EADS server is also placed in `running` status.

The following table explains each status shown in the figure.

Table 2–6: EADS server status

EADS server status		Description
Status name	Status displayed by the eztool status command	
Initializing	`initializing`	The EADS server is being initialized. This status is also displayed when an EADS server is being restored by using the `ezstart -r` command or being added by using the `ezstart -ai` command.
Restoring		
Adding		
Initialized	`initialized`	The EADS server has just been initialized. As with `closed` status, the EADS server cannot accept requests from EADS clients.
Running	`running`	The EADS server can accept requests from EADS clients.
Closing	`closing`	Although the EADS server cannot accept requests from EADS clients, there are threads that are still running.
Closed	`closed`	The EADS server cannot accept requests from EADS clients.
Isolated	`isolated`	As with `closed` status, the EADS server cannot accept requests from EADS clients. An EADS server that has been removed from the cluster for reasons such as a communication error is forcibly placed in `isolated` status. Use the `eztool isolate --stop` command to individually terminate the EADS servers in `isolated` status.
Stopping	`stopping`	The EADS server is in the process of stopping.
Stopped	`-----------`	The EADS server has stopped.

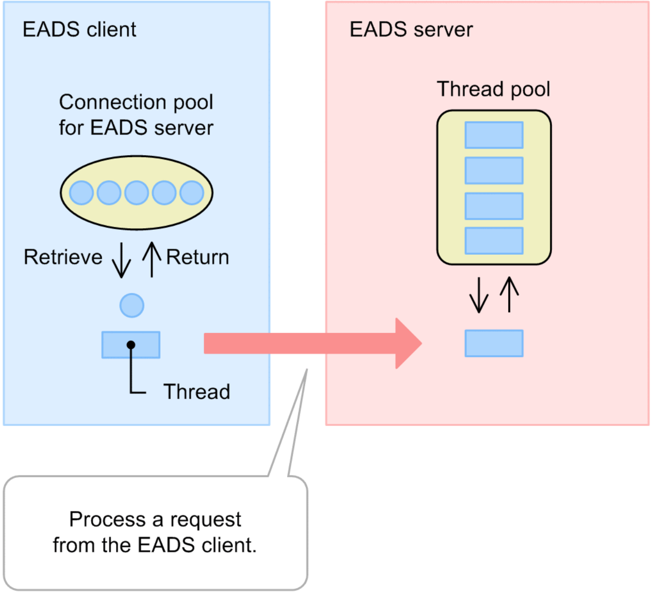
2.12 Improving throughput by using thread and connection pools

Thread pools and connection pools are used for communication between an EADS client and an EADS server.

If threads and connections are pooled before requests become concentrated, decreases in overall EADS server response speeds can be prevented.

Tune thread and connection pools according to the requirements of the system that uses EADS.

Figure 2–32: Overview of thread and connection pools borrow


EADS client
Connection pool for EADS server
Retrieve
Return
Thread
EADS server
Thread pool
Process a request from the EADS client.

2.12.1 Thread pools

On an EADS server, the threads for processing requests from an EADS client are created and pooled in advance. This is called a *thread pool*.

When the EADS server accepts a request, it allocates a thread from the thread pool and processes the request.

When an EADS server starts, the number of threads pooled in the thread pool equals the maximum number of simultaneous connections.

If there are no more available threads in the thread pool, an error results and communication is cut off. If the number of requests accepted exceeds the maximum number of simultaneous threads, processing is placed in wait status.

After the processing of one request is completed, the thread waits for the next request.

You can reduce the overhead of thread creation and deletion by using thread pools, because with them, repeated creation and deletion of threads is no longer necessary.

2.12.2 Connection pools

The EADS client reuses the connection that has been established by pooling it for each connection target. This is called a *connection pool*.

When communication begins, the EADS client checks for an existing connection with the connection target. If a connection has been pooled, EADS uses that connection to start communication.

When communication ends, EADS returns the connection to the connection pool.

If all connections in the connection pool are in use due to concentrated requests, as many new connections as can be pooled can be established.

If the maximum number of connections that can be pooled has already been reached, processing is placed in wait status until a connection is returned to the pool. Alternatively, you can set that an error is to be detected and processing is not to be placed in wait status.

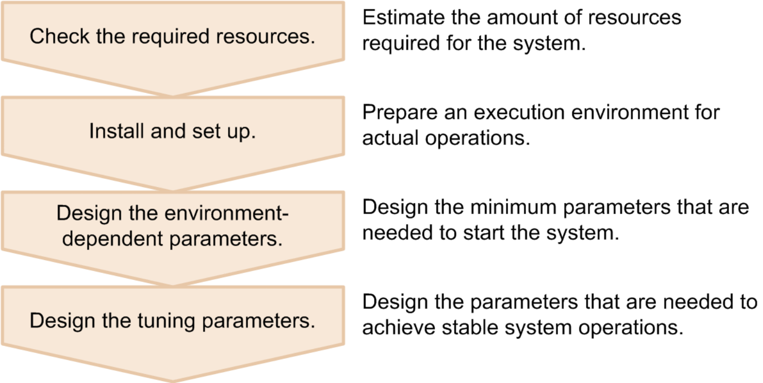
You can reduce communication overhead by using communication pools, because with them, it is not necessary to establish connections repeatedly.

3 General Procedure for Designing and Configuring a System

This chapter explains the general procedure for designing and configuring a system on which EADS has been deployed.

3.1 General procedure for designing and configuring a system

The following figure shows the general procedure for designing and configuring a system.


Check the required resources.
Estimate the amount of resources required for the system.
Install and set up.
Prepare an execution environment for actual operations.
Design the environment-dependent parameters.
Design the minimum parameters that are needed to start the system.
Design the tuning parameters.
Design the parameters that are needed to achieve stable system operations.

3.1.1 Checking the required resources

Before you configure your system, estimate the amount of resources the system requires, and then determine the number of machines and EADS servers needed to process them.

The number of EADS servers that make up a cluster must be at least the data multiplicity × 2 - 1.

See *4. Checking the Required Resources*.

3.1.2 Installing and setting up

Install the prerequisite program products, and then set up the EADS servers and EADS clients. Also, test the system by applying loads close to actual operation in the prepared execution environment.

See *5. Installing and Setting Up (EADS Servers)* or *6. Installing and Setting Up (EADS Clients)*.

3.1.3 Designing the environment-dependent parameters

Design the environment-dependent parameters for the system.

The environment-dependent parameters are the minimum parameters that need to be designed to start the system.

See *7. Designing the Environment-Dependent Parameters (EADS Servers)* or *8. Designing the Environment-Dependent Parameters (EADS Clients)*.

3.1.4 Designing the tuning parameters

Design the tuning parameters for the system.

The tuning parameters are parameters whose values need to be adjusted for your environment to achieve stable system operations.

See *9. Designing the Tuning Parameters*.

4 Checking the Required Resources

This chapter explains how to estimate the required amount of memory and disk capacity for the system.

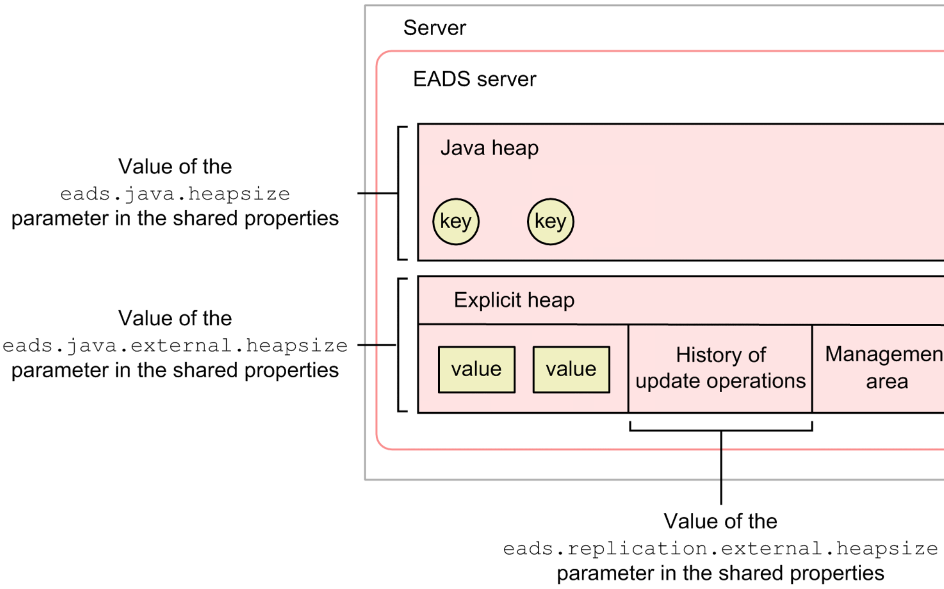
4.1 Estimating the required memory

This section explains how to estimate the memory capacity that is required to use EADS.

4.1.1 Memory configuration

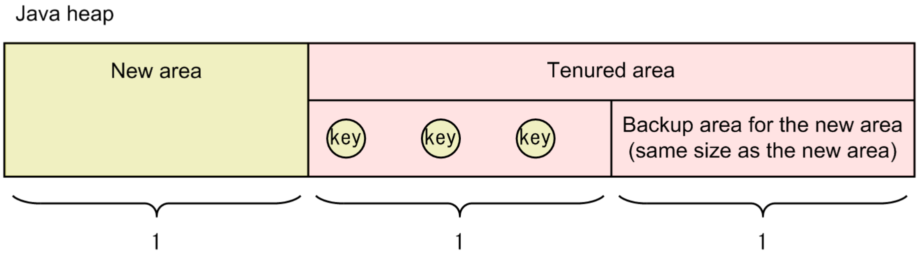
The following figure shows the configuration of memory used by an EADS server.

Figure 4–1: Memory configuration


Server
EADS server
Java heap
key
key
Explicit heap
value
value
History of update operations
Management area
Value of the eads.java.heapsize parameter in the shared properties
Value of the eads.java.external.heapsize parameter in the shared properties
Value of the eads.replication.external.heapsize parameter in the shared properties

(1) Java heap

A Java heap mainly consists of the new area and the tenured area. The key part of data is stored in the tenured area.



If the size of stored keys exceeds half of the tenured area size, `FullGC` (full garbage collection) occurs. Therefore, you must estimate the Java heap size based on the tenured area size. For details about how to estimate the Java heap size, see *4.1.2 Estimating the Java heap size*.

(2) Explicit heap

The explicit heap consists of an area for storing the value part of key-value pairs and an area for storing the history of update operations.

The history of update operations includes the API functions and information about keys and values.

Note that three percent of the explicit heap is used as a management area.

For details about how to estimate the explicit heap size, see *4.1.3 Estimating the explicit heap size*.

4.1.2 Estimating the Java heap size

The following subsections show the formulas for estimating the Java heap size.

(1) Estimating the Java heap size per EADS server

The tenured area size per one EADS server times three equals the Java heap size.

Specify the obtained value in the `eads.java.heapsize` parameter in the shared properties. The default is 3 gigabytes.

> **Reference note**
>
> When you create a cache on disk, add to the Java heap size the memory size used by that cache per EADS server. For details about estimating the size, see *4.1.2(2) Estimating the Java heap size used by a cache on disk*.

Java heap size (megabytes) =
{ (*maximum key size that can be stored in the cluster* (bytes) + 850 bytes)
× *number of data items stored per EADS server*
+ (*maximum key size that can be stored in the cluster* (bytes) + 250 bytes)
× *number of group names*[#]
+ *maximum number of simultaneous connections to EADS server*
× (14 + *data transmit and receive buffer size* (bytes) × 3)
+ *number of EADS servers* × 409,600 bytes
+ *Java heap area used for managing the history of update operations* (bytes)
+ *Java heap area used for isolation, restoration, and scale-out processing* (bytes) }
× 3 ÷ $1{,}024^2$

#

If the groups are arranged hierarchically, the number of group names of the first group is used.

Maximum key size that can be stored in the cluster (bytes):

`eads.cache.key.maxsize` parameter value in the shared properties

Number of data items stored per EADS server

The following shows the formula for estimating the number of data items stored per EADS server:

Number of data items stored per EADS server =
number of data items per range × *data multiplicity*

Number of data items per range:

The formulas for estimating the number of data items per range are shown below.

If you will be using the total data restriction function, specify the estimated value in the `eads.cache.keyCount` shared property parameter.

Number of data items per range =
(*number of data items stored per range in a memory cache*

+ *number of data items stored per range in a two-way cache*
+ *number of data items stored per range in a disk cache*)

Number of data items stored per range in a memory cache:

The following shows the formula for estimating the number of data items stored per range in a memory cache:

Number of data items stored per range in a memory cache =
number of data items stored in cluster's memory cache
÷ *number of EADS servers in the cluster*
+ *number of data items to be added to the memory cache for the total data restriction function*

Number of data items to be added to the memory cache for the total data restriction function:

The formula for estimating the number of data items to be added to the memory cache for the total data restriction function is shown below.

If you will not be using the total data restriction function, specify `0`.

If you will be using only a memory cache, add this value. If you will be using a two-way cache or a disk cache, specify `0`.

Number of data items to be added to the memory cache for the total data restriction function =
maximum number of simultaneous connections to EADS server × *maximum number of data items that can be updated simultaneously*

Maximum number of simultaneous connections to EADS server:

`eads.server.maxConnections` parameter value in the server properties

Maximum number of data items that can be updated simultaneously:

If you will be performing batch data operations, specify `10`. Otherwise, specify `1`.

Number of data items stored per range in a two-way cache:

The following shows the formula for estimating the number of data items stored per range in a two-way cache:

Number of data items stored per range in a two-way cache =
number of data items to be stored in the cluster's two-way cache
÷ *number of EADS servers in the cluster*
+ *number of two-way cache data items to be added for the total data restriction function*

Number of two-way cache data items to be added for the total data restriction function:

The formula for estimating the number of two-way cache data items to be added for the total data restriction function is shown below.

If you will not be using the total data restriction function, specify `0`.

Add this value if you will be using only two-way caches or if you will be using memory caches, disk caches, and two-way caches together. If you will be using only memory caches or only disk caches, specify `0`.

Number of two-way cache data items to be added for the total data restriction function =
maximum number of simultaneous connections to EADS server × *maximum number of data items that can be updated simultaneously*

Maximum number of simultaneous connections to EADS server:

`eads.server.maxConnections` parameter value in the server properties

Maximum number of data items that can be updated simultaneously:

If you will be performing batch data operations using a memory cache, specify `10`. Otherwise, specify `1`.

Number of data items stored per range in a disk cache:

The following shows the formula for estimating the number of data items stored per range in a disk cache:

Number of data items stored per range in a disk cache =
number of data items stored in the cluster's disk cache

÷ *number of EADS servers in the cluster*

+ *number of disk cache data items to be added for the total data restriction function*

Number of disk cache data items to be added for the total data restriction function:

The formula for estimating the number of disk cache data items to be added for the total data restriction function is shown below.

If you will not be using the total data restriction function, specify `0`.

Add this value if you will be using only a disk cache. If you will be using a memory cache or a two-way cache, specify `0`.

Number of disk cache data items to be added for the total data restriction function =

maximum number of simultaneous connections to EADS server × *maximum number of data items that can be updated simultaneously*

Maximum number of simultaneous connections to EADS server:

`eads.server.maxConnections` parameter value in the server properties

Maximum number of data items that can be updated simultaneously:

Specify `1`.

Maximum number of simultaneous connections to EADS server:

`eads.server.maxConnections` parameter value in the server properties

Data multiplicity:

`eads.replication.factor` parameter value in the shared properties

Data transmit and receive buffer size (bytes):

`eads.server.connection.buffersize` parameter value in the server properties

Java heap area used for managing the history of update operations (bytes):

The following shows the formula for estimating the Java heap area used for managing the history of update operations:

Java heap area used for managing the history of update operations (bytes) =

3,584 × (*maximum number of items that can be retained in the history of update operations*

+ *maximum number of consensus processes that can be executed simultaneously*)

× (*data multiplicity* × 2 - 1) × *numbers of caches*

+ 16 × *length of the queue for sending consensus messages* × (*number of EADS servers* - 1)

Maximum number of items that can be retained in the history of update operations:

The following shows the formula for estimating the maximum number of items that can be retained in the history of update operations.

Maximum number of items that can be retained in the history of update operations =

(*heartbeat timeout value* ÷ 1,000)

× *throughput of data update operations on one EADS server* (in operations/second)#

Heartbeat timeout value:

`eads.failureDetector.heartbeat.timeout` parameter value in the server properties

#

If the throughput varies greatly from one cache to another, determine the throughput for each cache and then specify the largest such value.

Maximum number of consensus processes that can be executed simultaneously:

`eads.replication.preparations` parameter value in the shared properties

Data multiplicity:

`eads.replication.factor` parameter value in the shared properties

Length of the queue for sending consensus messages:

`eads.replication.sendQueue.length` parameter value in the server properties

Java heap area used for isolation, restoration, and scale-out processing (bytes):

Estimate the maximum sizes of the Java heap areas used for isolation, restoration, and scale-out processing and specify the largest such value.

> Java heap area used for isolation processing (bytes) =
> (*maximum number of consensus processes that can be executed simultaneously* × 2
> × *maximum size of the history of update operations* (bytes)
> × (*maximum number of simultaneous threads for processing performed for caches*
> + *numbers of caches*) + (*data multiplicity* - 1) × *numbers of caches*
> × *size of data sent for complementary processing of the history of update operations* (bytes))
> × MIN(4 × (*data multiplicity* - 1), *number of EADS servers* - 1)
> Java heap area used for restoration and scale-out processing (bytes) =
> *size of area for storing the history of update operations* (megabytes) × 1,024^2
> ÷ (*data multiplicity* × 2 - 1)
> + *size of data transmitted during restoration processing and scale-out processing* (bytes)

Maximum number of consensus processes that can be executed simultaneously:

`eads.replication.preparations` parameter value in the shared properties

Maximum size of the history of update operations (bytes):

The following shows the formula for estimating the maximum size of the history of update operations.

> Maximum size of the history of update operations (bytes) =
> *maximum key size that can be stored in the cluster* + *maximum value size* × MAX(2, *maximum number of data items that can be updated simultaneously*)

Maximum key size that can be stored in the cluster:

`eads.cache.key.maxsize` parameter value in the shared properties

Maximum value size:

Maximum size that can be specified when `put`, `create`, `update`, or `replace` processing is performed.

MAX:

Choose the largest value within the parentheses that follow MAX.

Example: For MAX(2, 10), the calculation result is 10.

Maximum number of data items that can be updated simultaneously:

If you will be performing batch data operations using a memory cache, specify `10`. Otherwise, specify `1`.

Maximum number of simultaneous threads for processing performed for caches:

Number of redundant copies of data plus the original - 1 (use 1 if the number of redundant copies of data plus the original is 1)

Data multiplicity:

`eads.replication.factor` parameter value in the shared properties

Size of data sent for complementary processing of the history of update operations (bytes):

`eads.replication.fillgap.copy.datasize` parameter value in the server properties

MIN:

Selects the smaller value of the calculation results.

Example: For MIN(3 × 6, 4 + 7), the calculation result is 11.

Size of area for storing the history of update operations:

`eads.replication.external.heapsize` parameter value in the shared properties

For details about how to estimate this value, see *4.1.3(2) Size of the area for storing the history of update operations*.

Size of data transmitted during restoration processing and scale-out processing:

Estimate the size of data transmitted during restoration processing and scale-out processing for each cache and specify the largest such value.

- Memory caches

  `eads.transfer.datasize` parameter value in the server properties

- Disk caches and two-way caches

  `eads.cache.disk.transfer.datasize` parameter value in the cache properties

(2) Estimating the Java heap size used by a cache on disk

If you will be using a disk caches or a two-way cache, determine the Java heap size used by the cache.

The formulas for estimating the Java heap size for a disk cache and for a two-way cache are shown below. After estimating the sizes, obtain the sum of the obtained values, and then add that sum to the Java heap size estimated in *4.1.2(1) Estimating the Java heap size per EADS server*.

- Two-way cache

Value to be added to the estimated Java heap size (megabytes) =
(1,600 + 8 × *number of cache data files per range*
+ 0.8 × *number of data items stored per range in the two-way cache*) × *data multiplicity* ÷ 1,024

- Disk cache#

Value to be added to the estimated Java heap size (megabytes) =
(1,600 + 8 × *number of cache data files per range*
- 0.4 × *number of data items stored per range in the disk cache*) × *data multiplicity* ÷ 1,024

#

If there are many data items to be stored, a negative value might result. If that occurs, specify that negative value as is.

Number of cache data files per range:

`eads.cache.disk.filenum` parameter value in the cache properties

For details about how to estimate this value, see *4.4.1 Estimating the size and number of cache data files*.

Data multiplicity:

`eads.replication.factor` parameter value in the shared properties

4.1.3 Estimating the explicit heap size

The following shows the formula for estimating the explicit heap size.

The explicit heap size is the size of the area for storing the value part of key-value pairs plus the size of the area for storing the history of update operations. In addition, three percent of the explicit heap is used as a management area.

Specify the obtained value in the `eads.java.external.heapsize` parameter in the shared properties.

Explicit heap size (megabytes) =
(a ÷ $1{,}024^2$ + b ÷ $1{,}024^{2\#}$) ÷ 0.97

Explanation of the variables

a: Size of the area for storing the value part

b: Size of the area for storing the history of update operations

#

If you will be using the total data restriction function, round up the value of *a* ÷ $1{,}024^2$ to the next multiple of the number of redundant copies of data plus the original (megabytes).

The formula for estimating the size of each area is shown below.

(1) Size of the area for storing the value part

The following shows the formula for estimating the size of the area for storing the value part of key-value pairs. If values are not stored in the explicit heap (a disk cache only is used), specify `0`.

Size of the area for storing the value part (bytes) =
(*size of the value per item that is stored in a memory cache or a two-way cache*#1 (bytes)
+ 2 (bytes))#2
× (*sum of the number of data items to be stored per range in memory caches or the number of data items to be stored per range in two-way caches* × *data multiplicity*
+ *sum of the number of memory caches and the number of two-way caches*
× *data multiplicity* + 500)

#1

For details about the value size, see *15.2.2(3) Data types that can be specified as values*.

#2

Round up the value in the parentheses to a multiple of 16 bytes.

Data multiplicity:

`eads.replication.factor` parameter value in the shared properties

(2) Size of the area for storing the history of update operations

The formula for estimating the size of the area for storing the history of update operations is shown below.

Round up the obtained value up to the closest megabyte, and then specify the value for the `eads.replication.external.heapsize` parameter in the shared properties.

Size of the area for storing the history of update operations (bytes) =
{ (*data multiplicity* × 2 - 1) × *numbers of caches* × *maximum value for the history of update operations* + 500 }
× (*maximum size of the history of update operations* (bytes) + 1,024) + 1,048,576

Note:

Round up the decimal places to the closest whole number.

Data multiplicity:

`eads.replication.factor` parameter value in the shared properties

Maximum value for the history of update operations:

The following shows the formula for estimating the maximum value for the history of update operations.

> Maximum value for the history of update operations =
> *maximum number of items that can be retained in the history of update operations* + *maximum number of consensus processes that can be executed simultaneously*

Maximum number of items that can be retained in the history of update operations:

The following shows the formula for estimating the maximum number of items that can be retained in the history of update operations:

> Maximum number of items that can be retained in the history of update operations =
> (*heartbeat timeout value* ÷ 1,000)
> × *throughput of data update operations on one EADS server* (in operations/second)#

Heartbeat timeout value:

`eads.failureDetector.heartbeat.timeout` parameter value in the server properties

#

If the throughput varies greatly from one cache to another, determine the throughput for each cache and then specify the largest such value.

Maximum number of consensus processes that can be executed simultaneously:

`eads.replication.preparations` parameter value in the shared properties

Maximum size of the history of update operations (bytes):

The following shows the formula for estimating the maximum size of the history of update operations.

> Maximum size of the history of update operations (bytes) =
> *maximum key size that can be stored in the cluster* + *maximum value size* × MAX(2, *maximum number of data items that can be updated simultaneously*)

Maximum key size that can be stored in the cluster:

`eads.cache.key.maxsize` parameter value in the shared properties

Maximum value size:

Maximum size that can be specified when `put`, `create`, `update`, or `replace` processing is performed.

MAX:

Choose the largest value in the parentheses that follow MAX.

Example: For MAX(2, 10), the calculation result is 10.

Maximum number of data items that can be updated simultaneously:

If you will be performing batch data operations using a memory cache, specify `10`. Otherwise, specify `1`.

Important note

- If the size estimated for the explicit heap is not accurate enough and there is not enough space in the explicit heap, values and the history of update operations might be fragmented, resulting in poor performance. If the total data restriction function is not used and the explicit heap runs out of space for storing values, EADS servers will be isolated.
- If the explicit heap runs out of space for storing the history of update operations, the EADS servers attempt to obtain more explicit heap by deleting existing update operations history information. As a result, complementary processing of the history of update operations between EADS servers might fail. For details

about complementary processing of the history of update operations, see *9.3.2(7) Complementary processing of the history of update operations*.

4.1.4 Estimating the size of memory used by an EADS server

The following shows the formula for estimating the size of memory used by an EADS server.

Size of memory used by an EADS server (megabytes) =
Java heap size + *explicit heap size* + 150
+ (*maximum number of simultaneous connections to the EADS server* × *data multiplicity*) × 52 ÷ 1,024

Maximum number of simultaneous connections to the EADS server:

`eads.server.maxConnections` parameter value in the server properties

Data multiplicity:

`eads.replication.factor` parameter value in the shared properties

4.2 Estimating the required disk capacity

This section explains how to estimate the disk capacity that is required to use EADS.

4.2.1 Estimating the disk capacity required for EADS servers

This subsection explains how to estimate the disk capacity required for EADS servers.

If you will be using disk caches and two-way caches, add the required disk capacity estimated in *4.4 Estimating the sizes of cache files* to the following values that are estimated here.

- Total size of log files per EADS server
- Size of store data files

(1) Estimating the total size of log files per EADS server

The following shows the formula for estimating the total size of log files per EADS server.

Total size of log files per EADS server (megabytes) =
$a + b + c + d + e$

Explanation of the variables

a: Size of message log files output by an EADS server
b: Size of message log files output during command execution
c: Size of Java log files
d: Size of statistics files
e: Size of user message log files

The following subsections explain the formulas used to estimate the sizes of individual log files.

For details about the log files managed by the EADS servers, see *7.4.1 Types of log files*.

(a) Size of message log files output by an EADS server

The following shows the formula for estimating the size of message log files output by an EADS server.

Size of message log files output by an EADS server (megabytes) =
size of a message log file (bytes) $\div$ $1{,}024^2$
$\times$ *number of message log files* + 100

Size of a message log file (bytes):

`eads.logger.message.filesize` parameter value in the server properties

Number of message log files:

`eads.logger.message.filenum` parameter value in the server properties

Round up the value to the next full megabyte because the value of the `eads.logger.message.filesize` parameter in the server properties is specified in bytes.

(b) Size of message log files output during command execution

The following shows the formula for estimating the size of message log files output during command execution.

Size of message log files output during command execution (megabytes) =
size of a message log file (bytes) ÷ $1{,}024^2$
× *number of message log files*

Size of a message log file (bytes):

`eads.command.logger.message.filesize` parameter value in the command properties

Number of message log files:

`eads.command.logger.message.filenum` parameter value in the command properties

Round up the value to the next full megabyte because the value of the `eads.command.logger.message.filesize` parameter in the command properties is specified in bytes.

(c) Size of Java log files

The following shows the formula for estimating the size of Java log files.

Size of Java log files (megabytes) =
(*size of a Java log file* (megabytes)
× *number of Java log files*) × 2 + 32

Size of a Java log file (megabytes):

`eads.java.log.filesize` parameter value in the server properties

Number of Java log files:

`eads.java.log.filenum` parameter value in the server properties

(d) Size of statistics files

The following shows the formula for estimating the size of statistics files.

Size of statistics files (megabytes) =
(*size of statistics files* (`eads_stats.csv`)
+ *size of statistics files for caches* (`eads_cache_stats.csv`)
+ *size of statistics files per range* (`eads_store_stats.csv`)
+ *size of statistics files for user functions* (`eads_function_stats.csv`))
÷ $1{,}024^2$

The following shows the formulas for estimating the sizes of individual statistics files.

■ Size of statistics files (eads_stats.csv)

Size of statistics files (`eads_stats.csv`) (bytes) =
{1,024 + 1,024 × (86,400 ÷ *statistics output interval*)}
× (*number of statistics files to be acquired* + 1)

■ Size of statistics files for caches (eads_cache_stats.csv)

Size of statistics files for caches (`eads_cache_stats.csv`) (bytes) =
{1,024 + (2,048

× (*number of memory caches* + (*number of disk caches* + *number of two-way caches*) × *data multiplicity*))
× (86,400 ÷ *statistics output interval*)}
× (*number of statistics files to be acquired* + 1)

■ Size of statistics files per range (eads_store_stats.csv)

■ When using the total data restriction function
Size of statistics files per range (eads_store_stats.csv) (bytes) =
{1,024 + (2,048
× (*data multiplicity* + *number of disk caches* + *number of two-way caches*))
× (86,400 ÷ *statistics output interval*)}
× (*number of statistics files to be acquired* + 1)
■ When not using the total data restriction function
Size of statistics files per range (eads_store_stats.csv) (bytes) =
0

■ Size of statistics files for user functions (eads_function_stats.csv)

Size of statistics files for user functions (`eads_function_stats.csv`) (bytes) =
{1,024 + (2,048 × *number of user functions*)
× (86,400 ÷ *statistics output interval*)}
× (*number of statistics files to be acquired* + 1)

Statistics output interval:
`eads.statistics.interval` parameter value in the server properties

Number of statistics files to be acquired:
`eads.statistics.filenum` parameter value in the server properties

(e) Size of user message log files

The following shows the formula for estimating the size of user message log files.

Size of user message log files (megabytes) =
size of a user message log file (bytes) ÷ $1{,}024^2$
× *number of user message log files*

Size of a user message log file (bytes):
`eads.user.logger.filesize` parameter value in the server properties

Number of user message log files:
`eads.user.logger.filenum` parameter value in the server properties

Round up the value to the next full megabyte because the value of the `eads.user.logger.filesize` parameter in the server properties is specified in bytes.

(2) Estimating the size of store data files

The following shows the formula for estimating the size of store data files.

Size of store data files (megabytes) =
(*key size per data item*[#1] + *value size per data item*[#1] + 100)
× *number of data items stored per EADS server*[#2]

× (*maximum number of store-data-file generations that are output when the eztool export command is executed*
+ *maximum number of store-data-file generations that are output when the eztool stop command is executed* + *1)*
÷ $1{,}024^2$

#1

This is the size obtained after serialization.

In C, this is the size obtained by adding 10 bytes to the value size specified in the EADS client.

#2

This includes the number of data items copied for redundancy.

Maximum number of store-data-file generations that are output when the `eztool export` command is executed:

`eads.admin.backup.exportCommand.generation.maxNum` parameter value in the shared properties

Maximum number of store-data-file generations that are output when the `eztool stop` command is executed:

`eads.admin.backup.stopCommand.generation.maxNum` parameter value in the shared properties

Important note

The management information is still stored even if data is exported to a store data file while there is no data in memory, in which case a maximum of four kilobytes are used. Therefore, the size of the store data file in use is never zero bytes.

For details about store data files, see *7.4.1 Types of log files*.

(3) Estimating the total size of log files and store data files per EADS server

Add the values obtained in *(1) Estimating the total size of log files per EADS server* and *(2) Estimating the size of store data files*.

If you will be using disk caches and two-way caches, add the required disk capacity estimated in *4.4 Estimating the sizes of cache files*.

4.2.2 Estimating the disk capacity required for EADS clients

This subsection explains how to estimate the disk capacity required for EADS clients.

(1) Estimating the total size of log files per EADS client

The following shows the formula for estimating the total size of log files per EADS client.

Total size of log files per EADS client (megabytes) =
a + 70 megabytes

Explanation of the variables

a: Size of message log files output by an EADS client

For details about log files managed by EADS clients, see *8.4.1 Types of log files*.

(a) Size of message log files output by an EADS client

The following shows the formula for estimating the size of a message log file output by an EADS client.

Size of message log files output by an EADS client (megabytes) =
size of a message log file (bytes) ÷ $1{,}024^2$
× *number of message log files*

Size of a message log file (bytes):

`eads.client.logger.message.filesize` parameter value in the client properties

Number of message log files:

`eads.client.logger.message.filenum` parameter value in the client properties

Round up the value to the next full megabyte because the value of the `eads.client.logger.message.filesize` parameter in the client properties is specified in bytes.

4.3 Estimating the numbers of threads and file descriptors

This section explains how to estimate the numbers of threads and file descriptors required to use EADS.

4.3.1 Estimating the number of threads

(1) Number of threads per EADS server process

The following shows the formula for estimating the number of threads per EADS server process.

> Number of threads per EADS server process =
> *numbers of caches* × (*data multiplicity* × 2 - 1) × 3 + *number of EADS servers* × 6
> + *maximum number of simultaneous threads for processing performed for caches*
> + *numbers of caches* × 2
> + *maximum number of simultaneous connections to EADS server* + 41

Data multiplicity:

`eads.replication.factor` parameter value in the shared properties

Maximum number of simultaneous threads for processing performed for caches:

Number of redundant copies of data plus the original - 1 (use 1 if the number of redundant copies of data plus the original is 1)

Maximum number of simultaneous connections to EADS server:

`eads.server.maxConnections` parameter value in the server properties

(2) Number of threads per command process

The following shows the formula for estimating the number of threads per command process.

> Number of threads per command process =
> *number of EADS server processes* × 2 + 50

4.3.2 Estimating the number of file descriptors

(1) Number of file descriptors per EADS server process

The following shows the formula for estimating the number of file descriptors per EADS server process.

> Number of file descriptors per EADS server process =
> *number of EADS servers* × 15 + *maximum number of simultaneous connections to EADS server*
> + *number of libraries created by the user*# + *number of Java libraries*
> + *number of statistics files to be acquired* + 35
> + *number of file descriptors used by caches on disk*

#

This is the sum of the number of `jar` files for user-created user functions and the number of libraries used by all user functions.

Maximum number of simultaneous connections to EADS server:

`eads.server.maxConnections` parameter value in the server properties

Number of statistics files to be acquired:

`eads.statistics.filenum` parameter value in the server properties

Number of file descriptors used by caches on disk:

If you will be using disk caches and two-way caches, add the value obtained from the following formula:

Number of file descriptors used by caches on disk =
5 × *data multiplicity* × *number of caches that use the disk*

Data multiplicity:

`eads.replication.factor` parameter value in the shared properties

Number of caches that use the disk:

Sum of the number of disk caches and the number of two-way caches

The following shows the formula for estimating the number of file descriptors required during `eztool import` command execution.

Number of file descriptors required during `eztool import` command execution =
maximum number of simultaneous connections to EADS server × (*data multiplicity* - 1) × 2
+ *number of EADS server processes* × 100 + *number of libraries created by the user*#
+ *number of file descriptors required for user programs*

#

This is the sum of the number of `jar` files for user-created user functions and the number of libraries used by all user functions.

Maximum number of simultaneous connections to EADS server:

`eads.server.maxConnections` parameter value in the server properties

Data multiplicity:

`eads.replication.factor` parameter value in the shared properties

(2) Number of file descriptors per command process

The following shows the formula for estimating the number of file descriptors per command process.

Number of file descriptors per command process =
number of EADS server processes × 10 + 100

4.4 Estimating the sizes of cache files

This section is applicable when you will be using disk caches and two-way caches.

If you will be using disk caches and two-way caches, estimate the sizes of the cache files listed below for disk caches and two-way caches, and then add the obtained values to the required disk capacity estimated in *4.2.1 Estimating the disk capacity required for EADS servers*.

- Cache data files
- Cache index files

For details about cache data files and cache index files, see *7.7.2 Specifying the types of cache files and their storage locations*.

4.4.1 Estimating the size and number of cache data files

This subsection explains how to estimate the following values:

- Size of a cache data file
- Number of cache data files

To estimate the size and number of cache data files:

1. Determine the write block size for the medium.
2. Estimate the size of one record of data that is stored in a cache data file.
3. Estimate the size of one cache data file and the actual size of the data that can be stored.
4. Based on step 2, estimate the maximum size of valid data that can be managed in the system.
5. Determine a compaction interval.
6. Based on steps 2 and 5, estimate the capacity (for update data) required for storing data that is updated during compaction.
7. From steps 4 and 6, estimate the capacity required for the cache data file.
8. From step 7, estimate the size of the cache data file per range that is specified in the `eads.cache.disk.filesize` cache property parameter and the number of cache data files per range that is specified in the `eads.cache.disk.filenum` cache property parameter.
9. Before starting system operation, make sure that there will be no problem including the time required for compaction.

The following subsections explain each step.

(1) Determining the write block size

The size of the data that is written to a cache data file at one time is called the write block size.

Specify the write block size according to the medium used to store cache data files. Specify this value in the `eads.cache.disk.blocksize` cache property parameter.

- HDD storage medium
  1 kilobyte (default)

- SDD storage medium
  Page size of the SSD in use (in kilobytes)

(2) Estimating the size of one record

Data is stored in EADS servers in records that include the key, the value, and control information.

The following shows the formula for estimating the size of one record:

Size of one record (kilobytes) =
(*size of key stored in EADS servers* (bytes)
+ *size of value stored in EADS servers* (bytes) + 36)# ÷ 1,024

#

Round up the value in the parentheses to a multiple of the write block size (in kilobytes) determined in *4.4.1(1) Determining the write block size* (`eads.cache.disk.blocksize` parameter value in the cache properties).

Size of key stored in EADS servers:

The following shows the formula for estimating the size of a key stored in EADS servers:

Size of key stored in EADS servers (bytes) =
size of key in characters + 4

Size of value stored in EADS servers:

The formula for estimating the size of a value stored in EADS servers depends on the language used for creating application programs. The formulas are shown in the following.

- When using C or when byte arrays are used in Java

Size of value stored in EADS servers (bytes) =
size of value in bytes specified in the API function + 2

- When using non-byte arrays in Java

Size of value stored in EADS servers (bytes) =
size of value in bytes that has been serialized by the java.io.ObjectOutputStream class + 2

Reference note

The formulas shown above must be used to estimate the sizes of keys and values stored in EADS servers, because EADS servers store keys and values in a unique format for purposes of processing efficiency.

(3) Estimating the size of one cache data file and the amount of data that can be stored in it

The size of one cache data file is specified in the `eads.cache.disk.filesize` cache property parameter. Because a cache data file contains twice as much management information as the block size, the actual size of the data storage area is estimated as follows:

Data storage area per cache data file (kilobytes) =
(*size of one cache data file* (megabytes) × 1,024)
- (*write block size* (kilobytes) × 2)

Size of one cache data file:

eads.cache.disk.filesize parameter value in the cache properties

Write block size:

Value determined in *4.4.1(1) Determining the write block size* (eads.cache.disk.blocksize parameter value in the cache properties)

You can determine the amount of data that can be stored in a cache data file from the size of the data storage area estimated by the above formula and the record size estimated in *4.4.1(2) Estimating the size of one record*.

(4) Estimating the maximum size of valid data

Estimate the maximum size of valid data that is managed in the system.

Valid data means the data that can be acquired by get and excludes any data that has become invalid due to update or deletion processing.

The following shows the formula for estimating the maximum size of valid data:

Maximum size of valid data (kilobytes) =
number of data items stored per range in a disk cache × *size of one record* (kilobytes)

Size of one record:

Value estimated in *4.4.1(2) Estimating the size of one record*

Reference note

For details about data that becomes invalid during update and deletion processing, see *10.9 Reducing the data usage of cache data files (performing compaction on cache data files)*.

(5) Determining a compaction interval

If you will be using disk caches and two-way caches, you will need to perform compaction on cache data files.

Determine the compaction interval (hours) according to the amount of data that is stored in EADS servers.

For details about compaction processing, see *10.9 Reducing the data usage of cache data files (performing compaction on cache data files)*.

(6) Estimating the capacity required for update data

Because data is always appended to cache data files, the amount of data stored in cache data files increases each time data is updated. Therefore, a cache data file must be large enough to accommodate the amount of data that is stored during one round of compaction processing (in hours) at the compaction interval (in hours).

Estimate the amount of data (update data) that is updated during one round of compaction processing. The following shows the formula for estimating the capacity for update data:

Capacity for update data (kilobytes) =
Number of update operations per hour in EADS servers × *size of one record* (kilobytes)
× (*compaction interval (in hours)* + *time required for one round of compaction processing (in hours)*)

Number of update operations per hour in EADS servers:

Number of times `put`, `create`, `update`, and `replace` are executed

Size of one record:

Value estimated in *4.4.1(2) Estimating the size of one record*

Compaction interval:

Value determined in *4.4.1(5) Determining a compaction interval*

Time required for one round of compaction processing:

This value depends on the system. Before you start system operations, check the time required for a round of compaction processing, and specify that value.

(7) Estimating the capacity required for cache data files

The following shows the formula for estimating the capacity required for cache data files:

Capacity required for cache data files (megabytes) =
(*maximum size of valid data* (kilobytes) × 2
+ *capacity for update data* (kilobytes)) ÷ 1,024

Maximum size of valid data:

Value determined in *4.4.1(4) Estimating the maximum size of valid data*

Capacity for update data:

Value determined in *4.4.1(6) Estimating the capacity required for update data*

(8) Designing the size and number of cache data files

Based on the capacity required for cache data files that was determined in *4.4.1(7) Estimating the capacity required for cache data files*, design the size of cache data files per range and the number of cache data files in such a manner that the following condition is satisfied:

Capacity required for cache data files (megabytes) ≤
size of cache data files per range (megabytes) × *number of cache data files*#

#

If the number of cache data files is expected to increase in the future, also consider the capacity of those cache data files.

Specify the capacity of cache data files per range in the `eads.cache.disk.filesize` cache property parameter.

The following shows the formula for estimating the number of cache data files per range:

Number of cache data files per range =
number of cache data files + 2#

#

If you will not be using the total data restriction function, specify `1`.

Specify the obtained value in the `eads.cache.disk.filenum` parameter in the cache properties.

(9) Checking the compaction schedule

Before you start system operations, make sure that there is no problem with the time required for compaction and the capacity that will be allocated.

If the condition shown below is not satisfied, an error might occur due to insufficient capacity because compaction processing cannot keep up with data update operations:

Capacity for update data < capacity allocated by one round of compaction processing

Important note

If the reason for not satisfying the above condition is poor hardware performance, you must tune the number of times compaction is to be performed and the compaction interval and also consider such measures as updating hardware or adding hosts.

4.4.2 Estimating the size of a cache index file

The following shows the formula for estimating the size of one cache index file:

Size of one cache index file (bytes) =
48 + 4 × {(*size of cache data files per range* (megabytes) × 1,024
÷ *write block size*(kilobytes)) - 2}

Size of cache data files per range:

`eads.cache.disk.filesize` parameter value in the cache properties

Write block size:

Value determined in *4.4.1(1) Determining the write block size* (`eads.cache.disk.blocksize` parameter value in the cache properties)

5 Installing and Setting Up (EADS Servers)

This chapter explains how to configure EADS servers.

5.1 Installing an EADS server

This section explains how to install an EADS server.

Perform the installation procedure as the `root` user (superuser).

5.1.1 Preparations before starting the installation

Before you install an EADS server, perform the preparations described below as the `root` user (superuser).

(1) Setting up the hosts file

In the `hosts` file, establish a correspondence between IP address and host name.

Important note

EADS might not function correctly if any settings are wrong.

5.1.2 Installation procedure

Important note

If another version of EADS server is already installed, that EADS server will be overwritten (note that this applies only to version 03-00 or later). Before you start the installation procedure, make sure that the EADS server process is not running.

To install an EADS server:

1. Log in to the machine on which you plan to install the EADS server with root permissions.

2. Place the installation CD-ROM on the CD-ROM drive.

3. Use the OS's `mount` command to mount the CD-ROM file system.
   An example of using the `mount` command is shown below.
   For the underlined part, specify the name of the mount directory for the CD-ROM file system.

   ```
   mount -r -o mode=0544 /dev/cdrom /mnt/cdrom
   ```

4. Use Hitachi Program Product Installer's `setup` command to start the setup program.
   An example of using the `setup` command is shown below.
   For the underlined part, specify the name of the mount directory for the CD-ROM file system.

   ```
   /mnt/cdrom/X64LIN/setup /mnt/cdrom
   ```

5. In Hitachi Program Product Installer's main menu, press the **I** key.
   The program selection window is displayed.

6. In the program selection window, move the cursor to the program that you want to install, and then press the **Space** key.

   Select the program that you want to install. An at mark (`@`) is displayed on the left of the selected program.

7. Verify that an at mark (`@`) is displayed on the left of the selected program, and then press the **I** key.

   The following message is displayed on the last line.

   ```
   Install PP? (y: install, n: cancel)==>
   ```

8. When a message asking whether you want to install the program is displayed, press the **Y** key.

   Installation begins. If you press the **N** key, installation is cancelled and the program selection window is displayed again.

9. When a message is displayed indicating that the installation is finished, press the **Q** key.

   The main menu is displayed again.

10. In the main menu, press the **Q** key.

The installation is now finished.

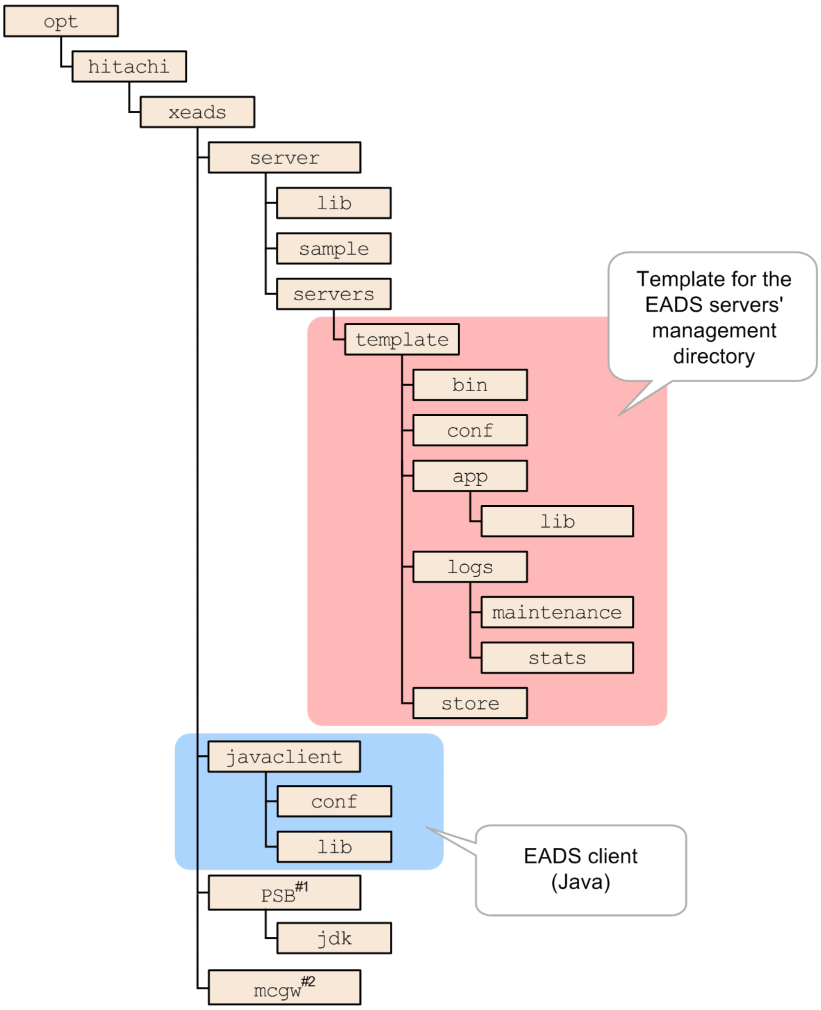
5.1.3 Post-installation procedure

After you have installed the EADS server, perform the tasks described below as the `root` user (superuser).

(1) Checking the directory configuration

After you have installed the EADS server, check the EADS server's directory configuration.

Figure 5–1: EADS server's directory configuration



#1: EADS uses only the `jdk` directory.
#2: Not used during normal operation.

The following table explains the EADS server's directory configuration shown in the figure:

Directory	Description
`/opt/hitachi/xeads/server/`	EADS server installation directory
`lib`	Stores the library files used for running the EADS server. The library files cannot be edited.
`sample`	Stores sample programs for the user functions that can be executed with the `eztool execfunc` command.
`servers`	Where the EADS server's management directory# is placed.
`template`	A template for the management directory#
`bin`	Stores the commands used to run the EADS server. The commands cannot be edited.
`conf`	Stores the following property files: • Server property file • Cluster property file • Shared property file

Directory	Description
	- Command property file - Cache property file
`app`	Stores function property files and `jar` files (user functions).
`lib`	Stores library files used in executing user functions.
`logs`	Output destination (default) of log files
`maintenance`	Output destination (default) of maintenance log files (used by the system)
`stats`	Output destination (default) of statistics files
`store`	Output destinations (defaults) for store data files, cache information files, and cache index files.
`/opt/hitachi/xeads/javaclient/`	EADS client (Java) installation directory
`conf`	Stores the client property file.
`lib`	Stores the library files.
`/opt/hitachi/xeads/PSB`	Directory for storing the JDK used by EADS.
`jdk`	JDK installation directory

#

The management directory is used to run the EADS server. Such items as the property files required for running the EADS server and the logs that are output during operation are stored in the management directory.

Important note

Do not change the directories and file owners, groups, and permissions that are created when EADS is installed. Store user-specific files only under the management directory. If these restrictions are not observed, EADS might not run correctly.

5.2 Setting up the EADS server

The tasks required after installation are performed by the OS user (system operation administrator).

5.2.1 Creating the management directory

The directory used for running an EADS server is called the management directory. Such items as the property files required for running the EADS server and the logs that are output during operation are stored in the management directory.

Log in as an OS user who runs the EADS server (system operation administrator), and then copy the `/opt/hitachi/xeads/server/servers/template` directory, a template for the management directory, to the `/opt/hitachi/xeads/server/servers` directory under an alias. The management directory name can consist of a maximum of 32 characters, including alphanumeric characters (from `0` to `9`, `A` to `Z`, and `a` to `z`) and underscores (_).

```
cp -rf /opt/hitachi/xeads/server/servers/template /opt/hitachi/xeads/server/
servers/any-management-directory-name
```

Note that in this manual, the term *EADS server name* means the management directory name.

You can assign a different OS user (system operation administrator) for each EADS server. Change the management directory access permissions as needed.

Important note

If you will be using the existing management directory, either delete the following files and directories or move them to another directory:

- All files and directories under the directory specified in the `eads.logger.dir` server property parameter
- All files and directories under the directory specified in the `eads.command.logger.dir` command property parameter

If the existing management directory is used, the commands under *management-directory*`/bin` are overwritten by the commands under the `/opt/hitachi/xeads/server/servers/template/bin` directory during installation.

Reference note

You can use a symbolic link created under the `/opt/hitachi/xeads/server/servers` directory as the management directory.

Use the symbolic link if you want to output to another disk the core dumps that are output directly under the management directory. Note that commands cannot be executed in the actual management directory.

5.2.2 Editing the property files

Specify the parameter values designed in *7. Designing the Environment-Dependent Parameters (EADS Servers)* in the following property files under *management-directory*/`conf`:

- Server property file (`eads_server.properties`)
- Cluster property file (`eads_cluster.properties`)
- Shared property file (`eads_shared.properties`)
- Command property file (`eads_management.properties`)
- Cache property file (`eads_cache.`*cache-name*`.properties`)#

#: If you will be using a memory cache, creation of a cache property file is optional.

For the tuning parameters, design the parameter values by referencing *9. Designing the Tuning Parameters*, and then specify the values in the property files.

If you are using user functions, create an optional function property file. Copy the created function property file together with the user functions to *management-directory*/`app` while the EADS server is stopped in the execution environment. For details, see *17.4 Creating a function property file (optional)* and *17.9 Distributing the directory to the execution environment*.

Important note

- The contents of the cluster property files and shared property files must be identical among all the EADS servers that make up a cluster. If the properties differ, the EADS servers cannot start because the cluster cannot be configured.
- The settings in the cache property files for the parameters shown below must be identical among all the EADS servers that make up a cluster. If any settings are different, caches cannot be created.
  - `eads.cache.type`
  - `eads.cache.disk.filesize`
  - `eads.cache.disk.filenum`
  - `eads.cache.disk.blocksize`

5.2.3 Distributing application programs

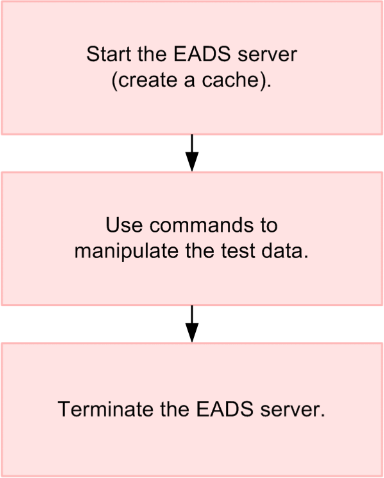
Distribute the application programs developed in the development environment to the execution environment.

For details about distributing user functions to the execution environment, see *17.9 Distributing the directory to the execution environment*.

5.3 Testing

After you have configured an execution environment, test it to see if it runs successfully. To test the execution environment, use commands. The following figure shows the general test procedure.

Figure 5–2: General test procedure



5.3.1 Starting the EADS server (creating a cache)

Start the EADS server and create a cache.

For details about the procedure, see *10.2 Starting the EADS servers (and creating caches)*.

5.3.2 Using commands to manipulate the test data

After you have created a cache, use commands to manipulate the test data.

(1) Storing test data

Execute the `eztool put` command to store test data.

```
eztool put cache-name key value
```

Example of command execution

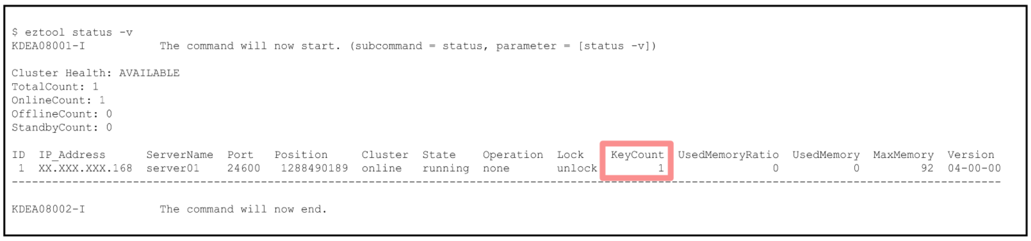
```
$ eztool put cache1 key1 value1
KDEA08001-I           The command will now start. (subcommand = put, parameter = [put, cache1, key1, value1])
KDEA08002-I           The command will now end.
```

(2) Checking whether the command ran successfully

Use the `eztool status -v` command to display the number of keys held by the EADS server.

```
eztool status -v
```

Example of command execution



```
$ eztool status -v
KDEA08001-I          The command will now start. (subcommand = status, parameter = [status -v])

Cluster Health: AVAILABLE
TotalCount: 1
OnlineCount: 1
OfflineCount: 0
StandbyCount: 0

ID  IP_Address       ServerName  Port    Position     Cluster  State    Operation  Lock    KeyCount  UsedMemoryRatio  UsedMemory  MaxMemory  Version
 1  XX.XXX.XXX.168  server01    24600   1288490189  online   running  none       unlock         1                0           0         92  04-00-00
-----------------------------------------------------------------------------------------------------------------------------------------------------

KDEA08002-I          The command will now end.
```

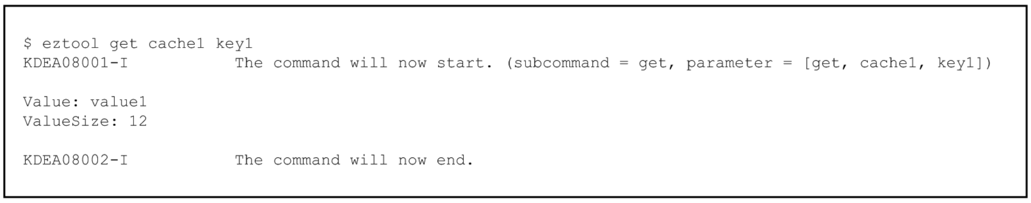
The `KeyCount` column displays the number of keys that were stored.

(3) Acquiring the stored data

Use the `eztool get` command to acquire the stored value.

```
eztool get cache-name key
```

Example of command execution



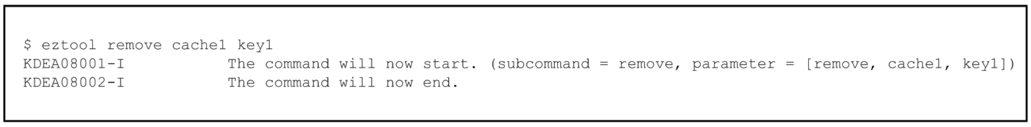
```
$ eztool get cache1 key1
KDEA08001-I          The command will now start. (subcommand = get, parameter = [get, cache1, key1])

Value: value1
ValueSize: 12

KDEA08002-I          The command will now end.
```

(4) Deleting the stored data

Use the `eztool remove` command to delete the stored value.

```
eztool remove cache-name key
```

Example of command execution

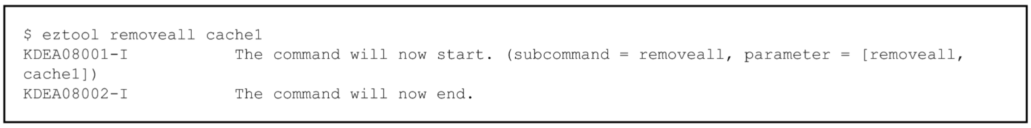


```
$ eztool remove cache1 key1
KDEA08001-I          The command will now start. (subcommand = remove, parameter = [remove, cache1, key1])
KDEA08002-I          The command will now end.
```

To delete keys stored in a specified cache and all the values associated with those keys, execute the `eztool removeall` command.

```
eztool removeall [-g group-name] cache-name
```

Example of command execution



```
$ eztool removeall cache1
KDEA08001-I          The command will now start. (subcommand = removeall, parameter = [removeall,
cache1])
KDEA08002-I          The command will now end.
```

When you specify the `-g` option in the `eztool removeall` command, you delete only the keys in the specified cache that belong to the specified group, and all the values associated with those keys.

5.3.3 Terminating the EADS server

Terminate the EADS server.

After the EADS server is terminated, all data is discarded from memory.

For details about the procedure, see *10.4 Terminating the EADS servers (and discarding data from memory)*.

5.4 Canceling EADS server setup

To cancel EADS server setup, execute the OS command shown below to delete the management directory that was set up.

```
rm -rf /opt/hitachi/xeads/server/servers/management-directory-name
```

The EADS server is now reset to the status that existed immediately after it was installed.

5.5 Uninstalling an EADS server

Important note

Check that the EADS server process is not running before you start uninstalling an EADS server.

This section explains how to uninstall an EADS server.

If an EADS server is uninstalled, all files except the management directory are deleted. Therefore, do not store user-specific files under the product directories.

To uninstall an EADS server:

1. Log in with root permissions to the machine from which you plan to uninstall the EADS server.
2. Use the OS command to start the setup program.

   ```
   /etc/hitachi_x64setup
   ```

3. In Hitachi Program Product Installer's main menu, press the **D** key.
   A list of programs currently installed is displayed.
4. Move the cursor to the program that you want to uninstall, and then press the **Space** key.
   An at mark (`@`) is displayed on the left of the selected program.
5. Press the **D** key.
   Uninstallation begins. When the uninstallation process is completed, the following message is displayed:

   ```
   Delete procedure completed.
   ```

6. When a message is displayed indicating that uninstallation is finished, press the **Q** key.
   The main menu is displayed again.
7. In the main menu, press the **Q** key.

Uninstallation is now finished.

6 Installing and Setting Up (EADS Clients)

This chapter explains how to configure EADS clients.

6.1 Installing an EADS client

This section explains how to install an EADS client.

Perform the installation procedure as the `root` user (superuser).

The installation procedure varies according to the application program language being used.

Java

First install an EADS server on a machine that will be used as an EADS server. An EADS client (Java) is stored in the EADS server's installation directory.

For details about how to install an EADS server, see *5.1.2 Installation procedure*.

After you have installed an EADS server, copy the following directory to another machine on which the EADS client is to be installed:

```
/opt/hitachi/xeads/javaclient
```

C

Install the following program product:

- Hitachi Elastic Application Data Store Client for C

For details about the installation procedure, see *5.1.2 Installation procedure*.

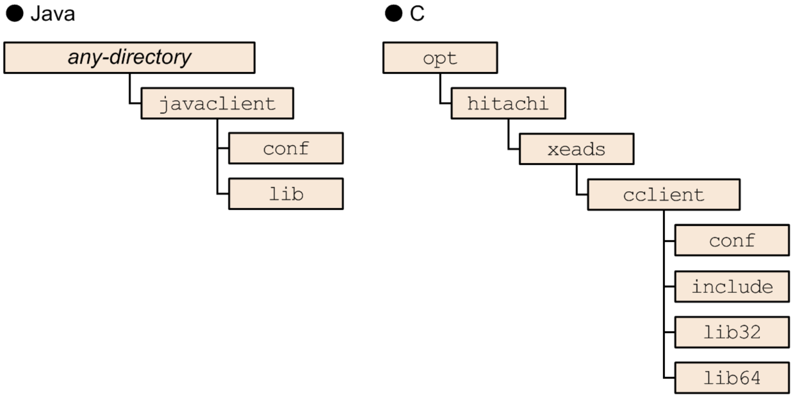
6.1.1 Post-installation procedures

Perform the tasks described in the following subsections after you have installed an EADS client.

(1) Checking the directory configuration

After you have installed the EADS client, check the EADS client's directory configuration.

Figure 6–1: EADS client's directory configuration



The following table explains the EADS client's directory configuration shown in the figure:

Directory	Description
any-directory/`javaclient/`	Directory to which the EADS client (Java) is copied
`conf`	Stores the client property file.

Directory	Description
`lib`	Stores the library files.
`/opt/hitachi/xeads/cclient/`	EADS client (C) installation directory.
`conf`	Stores the client property file.
`include`	Stores the header files.
`lib32`	Stores 32-bit edition of library files.
`lib64`	Stores 64-bit edition of library files.

Important note

Do not change the directories and file owners, groups, and permissions that are created when EADS is installed. Store user-specific files only under the management directory. If these restrictions are not observed, EADS might not run correctly.

(2) Specifying the libraries

Specify the libraries.

(a) Java

Specify the following libraries in the class path:

- `/opt/hitachi/xeads/javaclient/lib/eads-client.jar`
- `/opt/hitachi/xeads/javaclient/lib/eads-common.jar`
- `/opt/hitachi/xeads/javaclient/lib/hntrlib2-eads-j.jar`

If you run the EADS client on a Java EE server (uCosminexus Application Server), you must include the following libraries in the application program (WAR file (`WEB-INF/lib` directory)) On a Java EE server, you can use Servlet or JSP.

- `/opt/hitachi/xeads/javaclient/lib/eads-client.jar`
- `/opt/hitachi/xeads/javaclient/lib/eads-common.jar`
- `/opt/hitachi/xeads/javaclient/lib/hntrlib2-eads-j.jar`

(b) C

Specify the following libraries in the `LD_LIBRARY_PATH` environment variable:

- 32-bit edition
  `/opt/hitachi/xeads/cclient/lib32`
- 64-bit edition
  `/opt/hitachi/xeads/cclient/lib64`

6.2 Setting up an EADS client

This section explains where to place the application programs and how to edit the client property file.

6.2.1 Placement of the application programs

Place the application programs in any directory.

Important note

If you run the EADS client on a Java EE server (uCosminexus Application Server), deploy the application programs. Perform this task while Security Manager is released.

For details about how to create application programs, see *PART 4. Application Program Development*.

6.2.2 Editing the client property file

To edit the client property file:

1. Copy the client property file from the following directory to the directory specified in the application program (make sure that the file name also matches the one specified in the application program):
    - Java

      `/opt/hitachi/xeads/javaclient/conf/eads_sample_client.properties`
    - C

      `/opt/hitachi/xeads/cclient/conf/eads_sample_client.properties`
2. Edit the client property file.

   Specify in the client property file the values designed in *8. Designing the Environment-Dependent Parameters (EADS Clients)*.

For the tuning parameters, design the parameter values by referencing *9. Designing the Tuning Parameters*, and then specify the parameter values in the client property file.

6.3 Uninstalling an EADS client

This section explains how to uninstall an EADS client. The procedure varies depending on the programming language of the application program.

Java

Delete the directory to which the following EADS server directory was copied:

```
/opt/hitachi/xeads/javaclient
```

C

Uninstall the following program product:

- Hitachi Elastic Application Data Store Client for C

For the uninstallation procedure, see *5.5 Uninstalling an EADS server*. In this context, replace *EADS server* with *EADS client*.

7 Designing the Environment-Dependent Parameters (EADS Servers)

This chapter provides guidelines for designing the environment-dependent parameters for EADS servers.

7.1 Types of property files (used by EADS servers)

The table below lists and describes the types of property files used by the EADS servers.

This chapter explains only the environment-dependent parameters.

Table 7–1: Types of property files (used by EADS servers)

No.	Property file	File name	Description
1	Server property file	`eads_server.properties`	Defines an EADS server execution environment.
2	Cluster property file	`eads_cluster.properties`	Defines a cluster configuration.
3	Shared property file	`eads_shared.properties`	Defines settings that are common to all EADS servers.
4	Command property file	`eads_command.properties`	Defines the settings for command execution.
5	Cache property file	`eads_cache.`*cache-name*`.properties`	Defines cache properties. This file is created for each cache. For a memory cache, this file is optional. For a disk cache or a two-way cache, this file is required.
6	Function property file	This file can have any name. For details, see *17.4 Creating a function property file (optional)*.	Defines the settings for user function execution. This file is optional.

Important note

- The contents of the cluster property files and the shared property files must be identical among all the EADS servers that make up a cluster. If any properties are different, the EADS servers cannot start because the cluster cannot be configured.
- The settings in the cache property files for the parameters shown below must be identical among all the EADS servers that make up a cluster. If any settings are different, the caches cannot be created.
  - `eads.cache.type`
  - `eads.cache.disk.filesize`
  - `eads.cache.disk.filenum`
  - `eads.cache.disk.blocksize`
- Although the contents of all the property files need not be identical, we recommend that you use property files whose contents are the same (except for IP addresses and port numbers) on all EADS servers.

7.2 Format of property files

Specify parameters in the following format:

```
parameter=value
```

How to specify parameters

- A value extends up to an end-of-line character.
- A line beginning with a hash mark (#) is treated as a comment.
- A null line is ignored.
- A value cannot be followed by a character string such as a comment. If a character string is added, the value is interpreted as being invalid.
- If the same parameter is specified more than once, the last specification takes effect. For example, `false` would take effect if the following were specified:

  ```
  eads.logger.message.console.enable=true
  eads.logger.message.console.enable=false
  ```

Also note the following about shell scripts:

- The permitted characters are `0x20` through `0x7E` in ASCII codes, the newline character (`\n`), the carriage return character (`\r`), and the tab (`\t`).
- Specify one parameter per line. A line is through an end-of-line code (`\n` or `\r\n`) or the `EOF` (End of File).
- Only the equal sign (=) is supported as the separator between a parameter name and a specified value.

  Example: `a Δ b = c`

  Note: Δ indicates a space or a tab.

  EADS interprets that `a Δ b` is the parameter name and `c` is the value.
- A parameter name and a specified value can be preceded and followed by a single-byte space or a tab. EADS ignores the specified space or tab when it interprets the parameter.

  Example: `Δ a Δ = Δ b Δ`

  Note: Δ indicates a space or a tab.

  EADS interprets that `a` is the parameter name and `b` is the value.
- If two consecutive separators, single-byte spaces, or tabs are specified, the second and subsequent such characters are interpreted as part of the specified value.

  Example: `a==b`

  EADS interprets that `a` is the parameter name and `=b` is the value.
- If no value is specified (only a parameter name and a separator are specified), the value is interpreted as a null character string.
- If a specified parameter name contains a control character or only a parameter name is specified, that parameter is ignored.

7.3 Designing the communication-dependent parameters

Design the communication-dependent parameters.

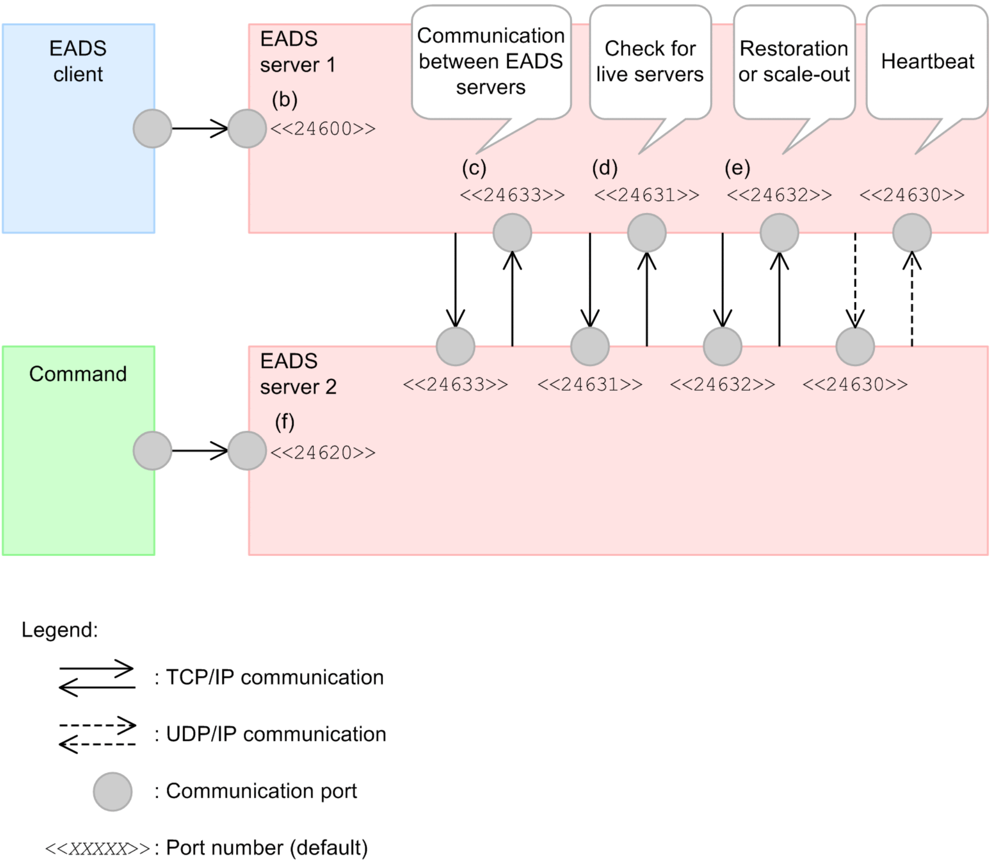
7.3.1 Specifying the IP address or host name and the port number

Specify the IP address (IPv4) or host name and the port number that the EADS server will use to communicate with the EADS client and other EADS servers.

For details about the mechanism of communication processing, see *2.2 Mechanisms of EADS communication processing*.

The following figure shows the communication ports used for communication between an EADS client and EADS servers.

Figure 7–1: Communication ports used for communication between an EADS client and EADS servers



The alphabetical letters assigned to the port numbers in the figure correspond to the letters in the explanation provided in *7.3.2 Communication-dependent parameters*.

(b): *7.3.2(1)(b) eads.server.port*

(c): *7.3.2(1)(c) eads.replication.port*

(d): *7.3.2(1)(d) eads.failureDetector.port*

(e): *7.3.2(1)(e) eads.transfer.port*

(f): *7.3.2(1)(f) eads.admin.operation.port*

If you specify connection-target EADS servers for the `eztool` command in the command properties and a connection cannot be established on the initial connection establishment attempt by the `eztool` command with the EADS server specified in the server properties, the command will attempt to establish a connection with another EADS server.

For details about the port numbers used for sending heartbeats between EADS servers, see *7.5.3(2)(b) eads.failureDetector.heartbeat.port* in *7.5.3 Cluster configuration-dependent parameters*.

Reference note

Because the transmission of heartbeats uses multicast communication, you must pay close attention when you set up the network. Specify a multicast address in the `eads.failureDetector.heartbeat.address` parameter in the shared properties.

When you specify port numbers, do not use a port number that is assigned by the OS to avoid duplication with other applications. The port numbers assigned by the OS depend on the type and version of the OS; for details, see the OS documentation.

7.3.2 Communication-dependent parameters

The table below lists the communication-dependent parameters.

Bold typeface indicates a required parameter.

Table 7–2: Communication-dependent parameters (EADS server)

No.	Property file	Parameter name	Value to be specified	Default value
1	Server property file	**`eads.server.address`**	IP address or host name	None
2		`eads.server.port`	Port number	`24600`
3		`eads.replication.port`	Port number	`24633`
4		`eads.failureDetector.port`	Port number	`24631`
5		`eads.transfer.port`	Port number	`24632`
6		`eads.admin.operation.port`	Port number	`24620`
7	Command property file	`eads.command.connect.sub.servers`	Name used to identify the connection-target EADS server (any name)	None
8		`eads.command.`*connection-target-EADS-server*`.address`	IP address or host name	None
9		`eads.command.`*connection-target-EADS-server*`.admin.operation.port`	Port number	`24620`

(1) Server property file

(a) eads.server.address

This parameter specifies the IP address or host name of the EADS server.

If you specify a host name, make sure that a unique IP address can be identified from the specified host name.

(b) eads.server.port

This parameter specifies the port number of the EADS server that is used for communication with the EADS client.

(c) eads.replication.port

This parameter specifies the port number used for communication between EADS servers.

(d) eads.failureDetector.port

This parameter specifies the port number used to check for live servers among the EADS servers.

(e) eads.transfer.port

This parameter specifies the port number used for EADS server restoration processing or scale-out processing.

(f) eads.admin.operation.port

This parameter specifies the port number used by commands.

(2) Command property file

(a) eads.command.connect.sub.servers

This parameter specifies the names (any names) used to identify the `eztool` command's connection-target EADS servers.

A name can consist of alphanumeric characters (`0` to `9`, `A` to `Z`, `a` to `z`).

When you specify multiple names, separate them with commas.

When the command is executed, it establishes an initial connection with the EADS server on which the command is executed (connection target specified in the `eads.server.address` and `eads.admin.operation.port` server property parameters).

When the `eztool` command is executed on the cluster and it fails to establish an initial connection with the EADS server on which it is executed, it attempts to establish a connection with the next connection-target EADS server specified in this parameter. When this parameter is omitted or a null character is specified in this parameter, establishment of an initial connection is attempted only for the EADS server on which the command is executed.

When any of the following commands is executed, this parameter is ignored:

- `eztool listgroup`
- `eztool listkey`
- `eztool removeall`
- `eztool execfunc`

Among the connection targets specified in this parameter and the eads.command.*connection-target-EADS-server*.admin.operation.port command property parameter, those that are specified in the `eads.server.address` and `eads.admin.operation.port` server property parameters are ignored.

(b) eads.command.connection-target-EADS-server.address

This parameter specifies the IP address or host name of the `eztool` command's connection-target EADS server.

If you specify a host name, make sure that a unique IP address can be identified from the specified host name.

Specify this parameter paired with the `eads.command.connect.sub.servers` parameter. For *connection-target-EADS-server*, specify the name used to identify the EADS server specified in the `eads.command.connect.sub.servers` parameter.

(c) eads.command.connection-target-EADS-server.admin.operation.port

This parameter specifies the port number of the `eztool` command's connection-target EADS server.

Specify this parameter paired with the `eads.command.connect.sub.servers` parameter. For *connection-target-EADS-server*, specify the name used to identify the EADS server specified in the `eads.command.connect.sub.servers` parameter.

(d) Example of specifications of the eads.command.connect.sub.servers parameter

The following shows an example of specifications of the `eads.command.connect.sub.servers` parameter.

```
eads.command.connect.sub.servers=sv1,sv2,sv3
eads.command.sv1.address=XXX.XXX.X.138
eads.command.sv1.admin.operation.port=24600
eads.command.sv2.address=XXX.XXX.X.139
eads.command.sv2.admin.operation.port=24601
eads.command.sv3.address=XXX.XXX.X.140
eads.command.sv3.admin.operation.port=24602
```

7.4 Designing the log file-dependent parameters

Design the log file-dependent parameters.

7.4.1 Types of log files

The following table lists and describes the types of log files that are managed by the EADS servers.

Table 7–3: Types of log files (managed by EADS servers)

No.	Log file	Description
1	Message log file	This file is used to output message logs for checking operations and monitoring for errors. The messages logs are output by the EADS server or commands.
2	Exception log file	This file is used to output track traces required for investigating the causes of errors.
3	User message log file	This file is used to output information about the execution of user functions.
4	User exception log file	This file is used to output stack traces for exceptions that occur in user functions.
5	Cache-file operation log file	This file is used to output information needed for checking and monitoring cache data file usage status and compaction status.
6	Maintenance log file	This file is used by the system. There is no parameter to be specified by the user.
7	Distribution maintenance log file	This file is used by the system. No parameter specification by the user is required.
8	Statistics file	This file is used to output statistics used for evaluating tuning, measuring performance, and estimating resources.
9	Java log file	This file is used to output information about JavaVM's garbage collection (GC) and memory.
10	Thread dump	This file is used to output information about the threads running in Java processes.
11	Startup log file	This file is used to output log information when the `ezstart` command starts EADS servers. No parameter specification by the user is required.

7.4.2 Specifying the file output destinations

You can change the output destinations of the log files and store data files. The following table lists the file output destinations.

Table 7–4: File output destinations (EADS servers)

No.	Log file type	Output destination	File name	
1	Message log file output by the EADS server (message log information)	Directory specified in the `eads.logger.dir` parameter in the server properties	`Wrap`	`eads_server_message[`*n*`].log`
			`Shift`	`eads_server_message.log`

No.	Log file type	Output destination	File name	
2	Exception log file output by the EADS server (Exception log information)	Directory specified in the eads.logger.dir parameter in the server properties	Wrap	eads_server_exception[*n*].log
			Shift	eads_server_exception.log
3	User message log file	Directory specified in the eads.logger.dir parameter in the server properties	Wrap	eads_user_message[*n*].log
			Shift	eads_user_message.log
4	User exception log file	Directory specified in the eads.logger.dir parameter in the server properties	Wrap	eads_user_exception[*n*].log
			Shift	eads_user_exception.log
5	Cache-file operation log file	Directory specified in the eads.logger.dir parameter in the server properties	Wrap	eads_server_cache[*n*].log
			Shift	eads_server_cache.log
6	Maintenance log file output by the EADS server (maintenance log information)	*directory-specified-in-the-eads.logger.dir-parameter-in-the-server-properties*/maintenance	eads_server_maintenance[*n*].log	
7	Distribution maintenance log file output by the EADS server (maintenance log information)	*directory-specified-in-the-eads.logger.dir-parameter-in-the-server-properties*/maintenance	eads_dist_maintenance[*n*].log	
8	Message log file output during command execution (message log information)	Directory specified in the eads.command.logger.dir parameter in the command properties	Wrap	eads_command_message[*n*].log
			Shift	eads_command_message.log
9	Exception log file output during command execution (Exception log information)	Directory specified in the eads.command.logger.dir parameter in the command properties	Wrap	eads_command_exception[*n*].log
			Shift	eads_command_exception.log
10	Maintenance log file output during command execution (maintenance log information)	*directory-specified-in-the-eads.command.logger.dir-parameter-in-the-command-properties*/maintenance	eads_command_maintenance[*n*].log	
11	Statistics file (EADS server)	*directory-specified-in-the-eads.logger.dir-parameter-in-the-server-properties*/stats	• eads_stats.csv • eads_stats_[*n*].csv[#1]	
12	Statistics file (cache)	*directory-specified-in-the-eads.logger.dir-parameter-in-the-server-properties*/stats	• eads_cache_stats.csv • eads_cache_stats_[*n*].csv[#1]	
13	Statistics file (user functions)	*directory-specified-in-the-eads.logger.dir-parameter-in-the-server-properties*/stats	• eads_function_stats.csv • eads_function_stats_[*n*].csv[#1]	
14	Statistics file (range)	*directory-specified-in-the-eads.logger.dir-parameter-*	• eads_store_stats.csv • eads_store_stats_[*n*].csv[#1]	

No.	Log file type	Output destination	File name
		in-the-server-properties/ stats	
15	Statistics file (maintenance information)	*directory-specified-in-the-eads.logger.dir-parameter-in-the-server-properties*/ maintenance/stats	• eads_maintenance_stats.csv • eads_maintenance_stats_[*n*].csv#1
16	Java log file	Directory specified in the eads.logger.dir parameter in the server properties	• javalog[*nn*].log • ehjavalog[*nn*].log#2
17	Thread dump	Directory specified in the eads.logger.dir parameter in the server properties	javacore[*PID*].[*YYMMDDhhmmss*].txt
18	Startup log file	Directory specified in the eads.logger.dir parameter in the server properties	eads_start.log

Legend:

[*n*], [*nn*], [*nnn*]: Sequence number of the file

[*PID*]: EADS server's process ID

[*YYMMDDhhmmss*]: *YY*: year, *MM*: month, *DD*: day, *hh*: hour (00 through 23), *mm*: minute, *ss*: second

Wrap, Shift: Log rotation method

You can select either of the following log rotation methods:

- Wrap: Wrap-around method

  The sequence number of the file is added to the file name.

- Shift: Shift method

  The file name is fixed.

  The sequence number of the file is added to the log file's backup file name.

#1

The statistics files are rotated each day and sequence numbers are assigned to the file names.

When the statistics files are rotated, as many statistics files as specified in the eads.statistics.filenum parameter in the server properties are retained as backup files.

For details about the rotation of statistics files, see *7.4.4 Specifying the rotation of statistics files*.

#2

This file is used to output logs related to the explicit heap.

Important note

A directory of another machine that is connected via a network cannot be specified as the file output destination.

7.4.3 Specifying the file sizes and the numbers of files

Specify the default values for the sizes of log files and the numbers of files. After you have configured the EADS servers, change the parameter values, if necessary.

7.4.4 Specifying the rotation of statistics files

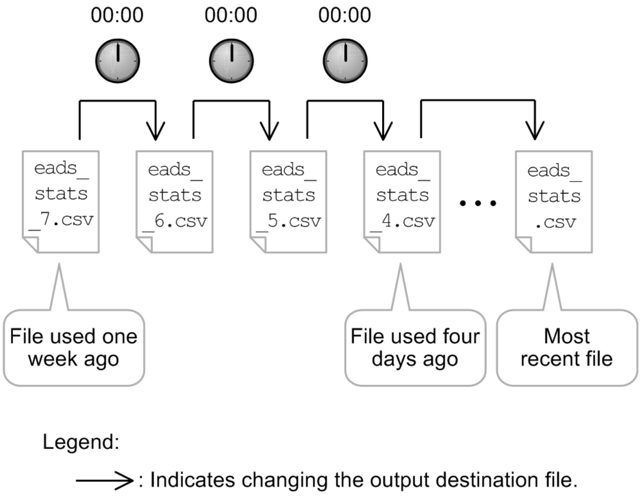
The statistics are output to the file when statistics are updated.

Specify the number of statistics files to be acquired in the `eads.statistics.filenum` parameter in the server properties.

One day's worth of statistics are output to one file. The statistics files are rotated each day.

The output destination is switched to a different file the first time the statistics are updated after a specific time (00:00) has passed.

Each time a statistics file is rotated, it is renamed `eads_stats_[`*n*`].csv`, where `[`*n*`]` indicates the sequence number of the file. The smaller the number, the newer the file, where `eads_stats.csv` (with no sequence number) is the most recent file.



7.4.5 Log file-dependent parameters

The following table lists the parameters that depend on log files.

Table 7–5: Log file-dependent parameters (EADS servers)

No.	Property file	Parameter name	Value to be specified	Default value
1	Server property file	`eads.logger.dir`	Path name	*management-directory*/`logs`
2		`eads.logger.message.rotationStyle`	• `Wrap` • `Shift`	`Wrap`

No.	Property file	Parameter name	Value to be specified	Default value
3		eads.logger.message.filesize	File size (4096 to 2147483647 (bytes))	1048576 (bytes)
4		eads.logger.message.filenum	Number of files (1 to 16)	2
5		eads.logger.message.console.enable	• true • false	false
6		eads.logger.exception.rotationStyle	• Wrap • Shift	Wrap
7		eads.logger.exception.filesize	File size (4096 to 2147483647 (bytes))	1048576 (bytes)
8		eads.logger.exception.filenum	Number of files (1 to 16)	2
9		eads.user.logger.rotationStyle	• Wrap • Shift	Wrap
10		eads.user.logger.filesize	File size (4096 to 2147483647 (bytes))	1048576 (bytes)
11		eads.user.logger.filenum	Number of files (1 to 16)	2
12		eads.user.logger.exception.rotationStyle	• Wrap • Shift	Wrap
13		eads.user.logger.exception.filesize	File size (4096 to 2147483647 (bytes))	1048576 (bytes)
14		eads.user.logger.exception.filenum	Number of files (1 to 16)	2
15		eads.cache.logger.diskCache.rotationStyle	• Wrap • Shift	Wrap
16		eads.cache.logger.diskCache.filesize	File size (4096 to 2147483647 (bytes))	1048576 (bytes)
17		eads.cache.logger.diskCache.filenum	Number of files (1 to 16)	2
18		eads.statistics.interval	1 to 3600	1
19		eads.statistics.filenum	1 to 366	7
20		eads.statistics.compaction.effect.division	1 to 10	5
21		eads.java.log.filesize	File size (1 to 2097152 (megabytes))	8 (megabytes)
22		eads.java.log.filenum	Number of files (1 to 99)	4

No.	Property file	Parameter name	Value to be specified	Default value
23	Command property file	`eads.command.logger.dir`	Path name	*management-directory*/`logs`
24		`eads.command.logger.message.rotationStyle`	• `Wrap` • `Shift`	`Wrap`
25		`eads.command.logger.message.filesize`	File size (`4096` to `2147483647` (bytes))	`1048576` (bytes)
26		`eads.command.logger.message.filenum`	Number of files (`1` to `64`)	`2`
27		`eads.command.logger.exception.rotationStyle`	• `Wrap` • `Shift`	`Wrap`
28		`eads.command.logger.exception.filesize`	File size (`4096` to `2147483647` (bytes))	`1048576` (bytes)
29		`eads.command.logger.exception.filenum`	Number of files (`1` to `16`)	`2`

(1) Server property file

(a) eads.logger.dir

This parameter specifies the path of the output destination directory for log files output by the EADS server.

If the specified path does not exist, it is created.

For details about the log files output under the directory specified in this parameter, see *7.4.2 Specifying the file output destinations*.

The path of an output destination directory can consist of a maximum of 96 characters, including alphanumeric characters (`0` to `9`, `A` to `Z`, `a` to `z`), underscores (`_`), colons (`:`), and separators (forward slashes (`/`)).

If you have changed the value of this parameter, either move all files and directories specified in this parameter to another directory or delete them.

(b) eads.logger.message.rotationStyle

This parameter specifies the rotation method for messages issued by the EADS server.

`Wrap`

Uses the wrap-around method.

`Shift`

Uses the shift method.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.logger.dir` server property parameter to another directory or delete them.

(c) eads.logger.message.filesize

This parameter specifies the size (in bytes) of one file to which message logs are output by the EADS server.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.logger.dir` server property parameter to another directory or delete them.

(d) eads.logger.message.filenum

This parameter specifies the number of files to which message logs are output by the EADS server.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.logger.dir` server property parameter to another directory or delete them.

(e) eads.logger.message.console.enable

This parameter specifies whether output of message logs from the EADS server to standard output is enabled.

`true`

Enables output to standard output.

`false`

Disables output to standard output.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.logger.dir` server property parameter to another directory or delete them.

(f) eads.logger.exception.rotationStyle

This parameter specifies the rotation method for the exception logs that are output by the EADS server.

`Wrap`

Uses the wrap-around method.

`Shift`

Uses the shift method.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.logger.dir` server property parameter to another directory or delete them.

(g) eads.logger.exception.filesize

This parameter specifies the size (in bytes) of an exception log file that is output by the EADS server.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.logger.dir` server property parameter to another directory or delete them.

(h) eads.logger.exception.filenum

This parameter specifies the number of exception log files to be output by the EADS server.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.logger.dir` server property parameter to another directory or delete them.

(i) eads.user.logger.rotationStyle

This parameter specifies the rotation method for user logs.

`Wrap`

Uses the wrap-around method.

`Shift`

Uses the shift method.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.logger.dir` server property parameter to another directory or delete them.

(j) eads.user.logger.filesize

This parameter specifies the size (in bytes) of one user message log file.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.logger.dir` server property parameter to another directory or delete them.

(k) eads.user.logger.filenum

This parameter specifies the number of user message log files.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.logger.dir` server property parameter to another directory or delete them.

(l) eads.user.logger.exception.rotationStyle

This parameter specifies the rotation method for the user exception logs.

`Wrap`

Uses the wrap-around method.

`Shift`

Uses the shift method.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.logger.dir` server property parameter to another directory or delete them.

(m) eads.user.logger.exception.filesize

This parameter specifies the size (in bytes) of a user exception log file.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.logger.dir` server property parameter to another directory or delete them.

(n) eads.user.logger.exception.filenum

This parameter specifies the number of user exception log files.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.logger.dir` server property parameter to another directory or delete them.

(o) eads.cache.logger.diskCache.rotationStyle

This parameter specifies the rotation method for the cache-file operation logs.

`Wrap`

Uses the wrap-around method.

`Shift`

Uses the shift method.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.logger.dir` server property parameter to another directory or delete them.

(p) eads.cache.logger.diskCache.filesize

This parameter specifies the size (in bytes) of one cache-file operation log file.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.logger.dir` server property parameter to another directory or delete them.

(q) eads.cache.logger.diskCache.filenum

This parameter specifies the number of cache-file operation log files.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.logger.dir` server property parameter to another directory or delete them.

(r) eads.statistics.interval

This parameter specifies an interval (in seconds) at which statistics are output.

(s) eads.statistics.filenum

This parameter specifies the number of statistics files that are acquired.

If you have changed the value of this parameter, either move the following files to another directory or delete them:

- All files with the extension `.mm` under *directory-specified-in-the-eads.logger.dir-parameter-in-the-server-properties*`/stats`
- All files with the extension `.mm` under *directory-specified-in-the-eads.logger.dir-parameter-in-the-server-properties*`/maintenance/stats`

(t) eads.statistics.compaction.effect.division

This parameter specifies the number of distribution ranges to be used when indicating through use of a cache statistic the distribution of the number of files in each compaction effects range.

For example, if `4` is specified in this parameter, the distribution is divided into four ranges, and the numbers of files in the compaction effects ranges 0% to 25%, 26% to 50%, 51% to 75%, and 76% to 100% are displayed.

If the value indicating effects is not an integer, all digits following the decimal point are discarded.

(u) eads.java.log.filesize

This parameter specifies the size (in megabytes) of a Java log file.

(v) eads.java.log.filenum

This parameter specifies the number of Java log files.

(2) Command property file

(a) eads.command.logger.dir

This parameter specifies the path of the output destination directory for log files output during command execution.

If the specified path does not exist, it is created.

For details about the log files output under the directory specified in this parameter, see *7.4.2 Specifying the file output destinations*.

The path of an output destination directory can consist of a maximum of 96 characters, including alphanumeric characters (`0` to `9`, `A` to `Z`, `a` to `z`), underscores (`_`), colons (`:`), and separators (forward slashes (`/`)).

If you have changed the value of this parameter, either move all files and directories specified in this parameter to another directory or delete them.

(b) eads.command.logger.message.rotationStyle

This parameter specifies the rotation method for messages that are issued during command execution.

`Wrap`

Uses the wrap-around method.

`Shift`

Uses the shift method.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.command.logger.dir` command property parameter to another directory or delete them.

(c) eads.command.logger.message.filesize

This parameter specifies the size (in bytes) of one file to which message logs are output during command execution.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.command.logger.dir` command property parameter to another directory or delete them.

(d) eads.command.logger.message.filenum

This parameter specifies the number of files to which message logs are output during command execution.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.command.logger.dir` command property parameter to another directory or delete them.

(e) eads.command.logger.exception.rotationStyle

This parameter specifies the rotation method for the exception logs that are output during command execution.

`Wrap`

Uses the wrap-around method.

`Shift`

Uses the shift method.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.command.logger.dir` command property parameter to another directory or delete them.

(f) eads.command.logger.exception.filesize

This parameter specifies the size of an exception log file (in bytes) that is output during command execution.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.command.logger.dir` command property parameter to another directory or delete them.

(g) eads.command.logger.exception.filenum

This parameter specifies the number of exception log files to be output during command execution.

If you have changed the value of this parameter, either move all files and directories under the directory specified in the `eads.command.logger.dir` command property parameter to another directory or delete them.

7.5 Designing the cluster configuration-dependent parameters

Design the parameters that depend on the cluster configuration.

7.5.1 Specifying the locations of EADS servers

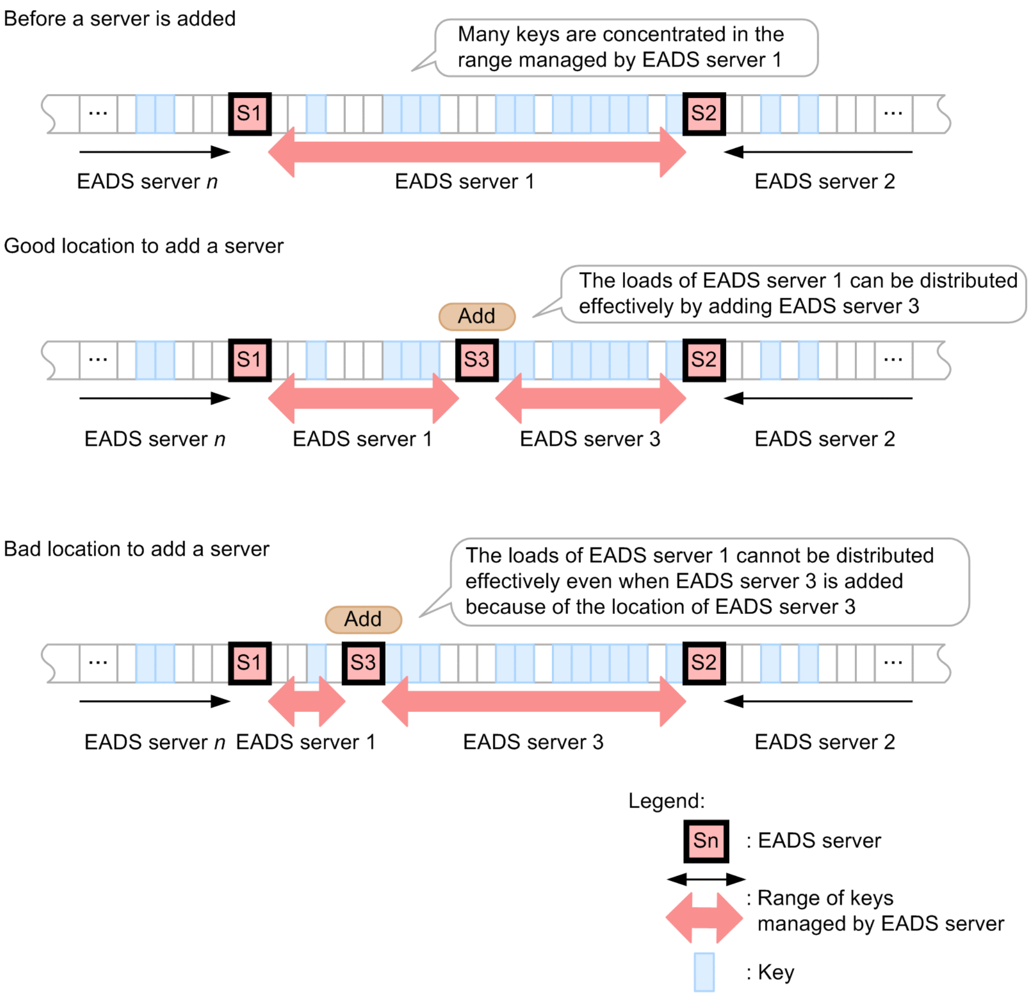
You use the `eads.node.`*EADS-server-ID*`.position` parameter to specify the location in the cluster where the specified EADS server is to be placed. Specifying the locations of EADS servers (hash values) enables you to distribute the workload.

If locations are not specified, the EADS servers are distributed evenly to logical locations in the cluster.

The placement of data (number of data items, data usage amounts, or data access counts) might become uneven depending on the nature of keys and applications. As a result, a large load might be placed on some of the EADS servers, resulting in poor response speeds and a shortage of resources.

Before you specify locations for the EADS servers, check the statistics (`KeyCount`, `UsedMemorySize`, and `RequestCount`) and analyze the intended locations.

Figure 7–2: Establishing a location for an EADS server


Before a server is added
Many keys are concentrated in the range managed by EADS server 1
S1
S2
EADS server n
EADS server 1
EADS server 2
Good location to add a server
The loads of EADS server 1 can be distributed effectively by adding EADS server 3
Add
S1
S3
S2
EADS server n
EADS server 1
EADS server 3
EADS server 2
Bad location to add a server
The loads of EADS server 1 cannot be distributed effectively even when EADS server 3 is added because of the location of EADS server 3
Add
S1
S3
S2
EADS server n
EADS server 1
EADS server 3
EADS server 2
Legend:
Sn : EADS server
: Range of keys managed by EADS server
: Key

For details about data distribution, see *2.5 Data distribution by consistent hashing*.

7.5.2 Specifying the data multiplicity

Use the `eads.replication.factor` parameter to specify the multiplicity of data.

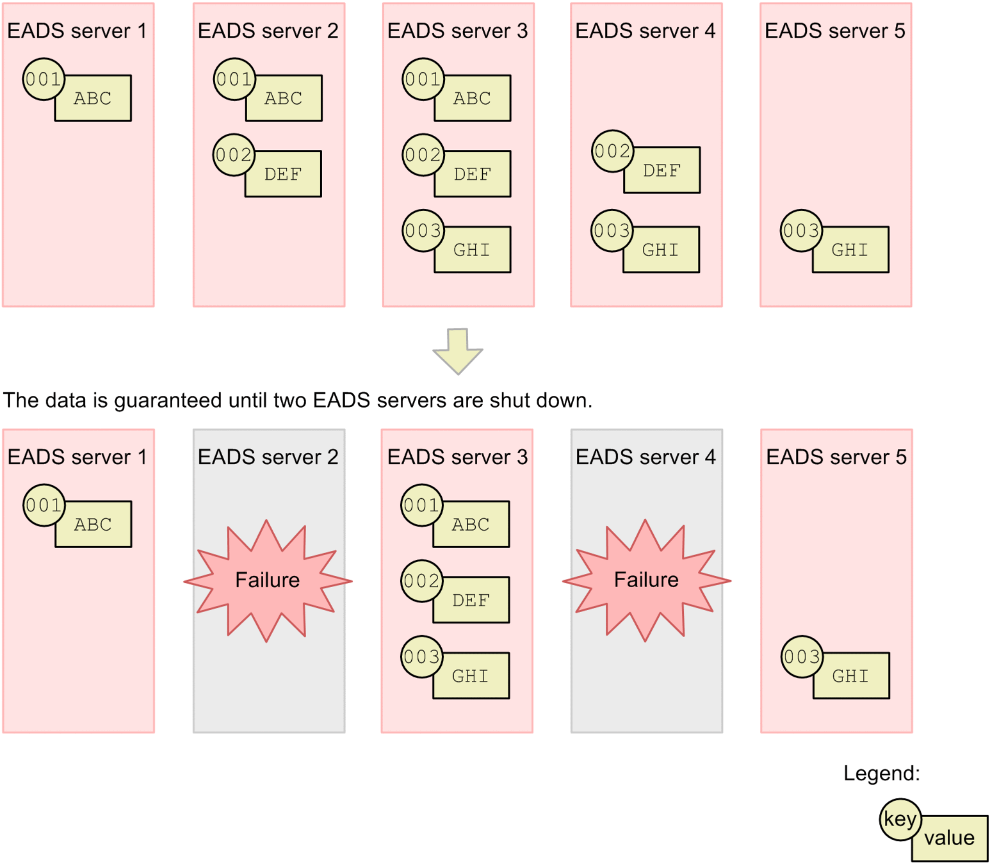
The number of EADS servers that make up a cluster must be at least the data multiplicity × 2 - 1. Data consistency is maintained as long as the number of EADS servers that have shut down due to failures is less than the number of data copies plus the original. Therefore, as the multiplicity becomes higher, reliability and fault tolerance improve.

On the other hand, more memory might be required and communication overhead between EADS servers might increase.

You must take into account advantages and disadvantages such as these when you set the multiplicity.

The following example sets 3 as the multiplicity.

Figure 7–3: Setting the data multiplicity



The data is guaranteed until two EADS servers are shut down.

If the multiplicity is set to 3, at least five EADS servers are required. If there are five EADS servers and two of them shut down due to failures, no data loss occurs.

For details about how redundant copies of data are created, see *2.8 Creating redundant copies of data*.

7.5.3 Cluster configuration-dependent parameters

The table below lists the parameters that depend on the cluster configuration.

Bold typeface indicates a required parameter.

Table 7–6: List of cluster configuration-dependent parameters

No.	Property file	Parameter name	Value to be specified	Default value
1	Cluster property file	**eads.node.*EADS-server-ID*.address**	IP address or host name	None
2		**eads.node.*EADS-server-ID*.port**	Port number	None
3		eads.node.*EADS-server-ID*.position	-2147483648 to 2147483647	None
4	Shared property file	eads.failureDetector.heartbeat.address	IP address (multicast address)	239.255.2.1
5		eads.failureDetector.heartbeat.port	Port number	24630
6		eads.replication.factor	1 to 5	2

Important note

The contents of the cluster property files and the shared property files must be identical among all the EADS servers that make up a cluster. If the properties differ, the EADS servers cannot start because the cluster cannot be configured.

If a value that is specified is outside the range of permissible values, the default value is set.

(1) Cluster property file

(a) eads.node.EADS-server-ID.address

This parameter specifies the IP addresses or host names (the eads.server.address parameter value in the server properties) of the EADS servers that make up the cluster.

If you specify a host name, make sure that a unique IP address can be identified from the specified host name.

EADS-server-ID is a user-assigned number (an integer from 1 through 96). The EADS server IDs are used for the store data file names to uniquely identify the individual EADS servers in the cluster. Although the EADS server IDs need not be sequential, they must be unique within the cluster.

(b) eads.node.EADS-server-ID.port

This parameter specifies the port numbers (the eads.server.port parameter value in the server properties) of the EADS servers that make up the cluster.

EADS-server-ID is a user-assigned number (an integer from 1 through 96). The EADS server IDs are used for the store data file names to uniquely identify the individual EADS servers in the cluster. Although the EADS server IDs need not be sequential, they must be unique within the cluster.

(c) eads.node.EADS-server-ID.position

This parameter specifies a location for an EADS server (hash value).

EADS-server-ID is a user-assigned number (an integer from 1 through 96). The EADS server IDs are used for the store data file names to uniquely identify the individual EADS servers in the cluster. Although the EADS server IDs need not be sequential, they must be unique within the cluster.

This parameter is optional. When this parameter is omitted, the EADS servers are distributed evenly to logical locations in the cluster.

When you use this parameter, specify locations for all EADS servers that make up the cluster. If the parameter is specified for some but not all of the EADS servers, an error occurs and EADS server startup fails.

(2) Shared property file

(a) eads.failureDetector.heartbeat.address

This parameter specifies the IP address (multicast address) used for transmitting heartbeats among the EADS servers.

You can also specify an alias of the IP address.

(b) eads.failureDetector.heartbeat.port

This parameter specifies the port number used for transmitting heartbeats among the EADS servers.

(c) eads.replication.factor

This parameter specifies the data multiplicity.

Important note

The number of EADS servers that make up the cluster must be at least the data multiplicity × 2 - 1.

An error will result if the number of EADS servers making up the cluster is less than the data multiplicity × 2 - 1, in which case startup of the EADS servers will fail.

7.6 Designing the backup file-dependent parameters

Design the backup file-dependent parameters.

A backup file to which data stored in a memory cache is output is called a store data file.

7.6.1 Specifying the file output destinations

You can change the output destinations of the store data files. The following table lists the file output destinations.

Table 7–7: Store data file output destinations

No.	Type of store data file	Output destination	File name
1	Store data file output during execution of the `eztool export` command	Directory specified in the `eads.admin.backup.dir` parameter in the server properties	• `eads_[`*xxx*`]_[`*EADS-server-ID*`].esd` • `eads_single_[`*xxx*`]_[`*EADS-server-ID*`].esd`
2	Store data file output during execution of the `eztool stop` command	Directory specified in the `eads.admin.backup.dir` parameter in the server properties	`eads_stop_[`*YYYYMMDDhhmmss*`]_[`*EADS-server-ID*`].esd`

Legend:

`[`*xxx*`]`: The store data file key specified during execution of the `eztool export` command

`[`*YYMMDDhhmmss*`]`: *YY*: year, *MM*: month, *DD*: day, *hh*: hour (00 through 23), *mm*: minute, *ss*: second

Notes:

A store data file name consists of a prefix, a store data file key (any value), and a suffix. The prefix depends on the command that output the store data file.

In the case of a store data file that is output when the `eztool export` command is executed and its store data file key is omitted, the command execution date and time become the store data file key, as shown in the table below.

In the case of a store data file that is output when the `eztool stop` command is executed, no store data file key can be specified. Its store data file key is always the command execution date and time.

-s or --single option	Store data file name	Generation management
Omitted	`eads_`*YYYYMMDDhhmmss*`_`*EADS-server-ID*`.esd`	Enabled
Specified	`eads_single_`*YYYYMMDDhhmmss*`_`*EADS-server-ID*`.esd`	Disabled

Legend:

YYYYMMDDhhmmss: Command execution date and time

YYYY: year, *MM*: month, *DD*: day, *hh*: hour (00 through 23), *mm*: minute, *ss*: second

Important note

A directory on another machine that is connected via a network cannot be specified as a file output destination.

7.6.2 Specifying the number of store data file generations

A store data file is output when the `eztool export` or `eztool stop` command is executed.

(1) Store data file output when the eztool export command is executed

When you use the `eztool export` command to export data, you normally omit the store data file key. In this case, the command execution date and time become the store data file key, as shown below:

`eads_`*YYYYMMDDhhmmss*`_`*EADS-server-ID*`.esd`

Legend:

YYYYMMDDhhmmss: Command execution date and time

YYYY: year, *MM*: month, *DD*: day, *hh*: hour (00 through 23), *mm*: minute, *ss*: second

The prefix `eads_` and the suffix `_`*EADS-server-ID*`.`*extension* (where the extension is `.esd`) are added automatically to the store data file key.

When the store data file key is in the format shown above and the `eztool import` command is executed, the store data file with the most recent command execution date and time is imported.

All data maintained by the EADS server is output to the store data file. This includes copies of data created for redundancy. The cache name and the key, update date, and value are output for each data item.

For details about the output destinations of store data files, see *7.4.2 Specifying the file output destinations*.

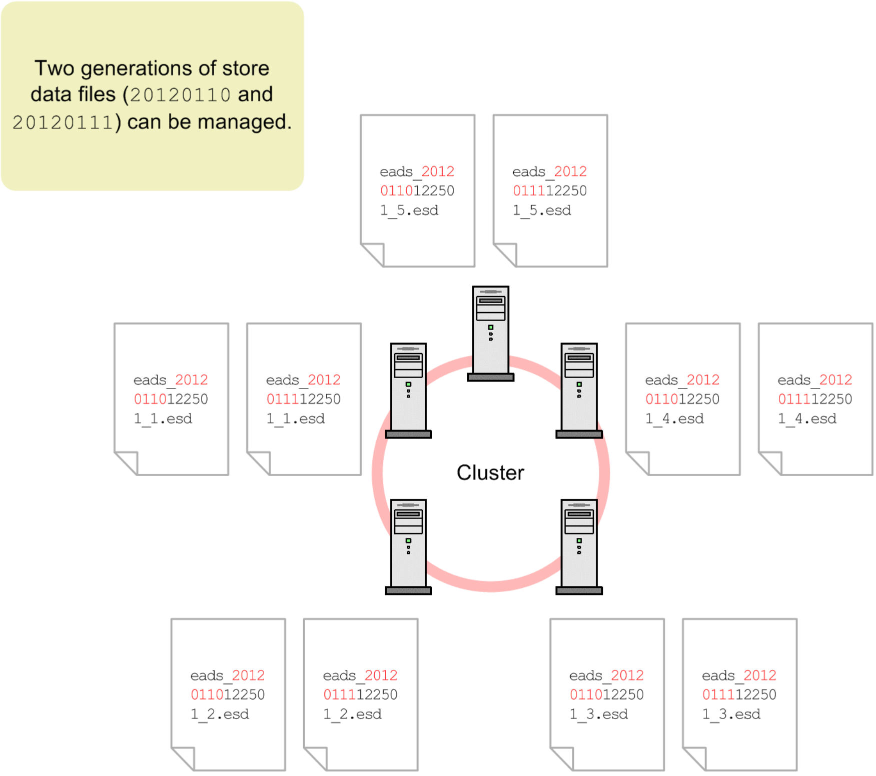
If the output destination already contains a store data file with the same name, the `eztool export` command results in an error; the existing file is not overwritten.

(2) Management of store data file generations

Store data files that have the same command execution date and time are treated as belonging to the same group, and their generations are managed in the entire cluster until the maximum number of generations specified in the `eads.admin.backup.exportCommand.generation.maxNum` parameter in the shared properties is reached. This prevents a disk capacity shortage, which can occur when store data files increase.

The following example specifies two (the default value) as the maximum number of store data file generations to be retained in the entire cluster.

Figure 7–4: Specifying the number of store data file generations


Two generations of store data files (20120110 and 20120111) can be managed.
eads_20120110122501_5.esd
eads_20120111122501_5.esd
eads_20120110122501_1.esd
eads_20120111122501_1.esd
eads_20120110122501_4.esd
eads_20120111122501_4.esd
Cluster
eads_20120110122501_2.esd
eads_20120111122501_2.esd
eads_20120110122501_3.esd
eads_20120111122501_3.esd

If more than this maximum number of generations of store data files are output, the `eztool export` command results in an error. Execute the `eztool deleteesd` command to delete any unneeded store data.

For details about how to check and delete store data files, see *11.7 Managing store data files*.

Important note

Generation management of store data files is not performed in the following cases:

- The value `0` is specified for the maximum number of generations (in the `eads.admin.backup.exportCommand.generation.maxNum` parameter in the shared properties)
- The `-s` or `--single` option is specified in the `eztool export` command

(3) Store data file output when the eztool stop command is executed

When the `eztool stop` command is executed to terminate the EADS server, data is exported to files and then the EADS server is terminated.

A store data file key cannot be specified in this case; instead, the command's execution date and time become the store data file key, as shown below.

The prefix `eads_stop_` and the suffix `_`*EADS-server-ID*`.`*extension* (where the extension is `.esd`) are added automatically to the store data file key:

`eads_stop_`*YYYYMMDDhhmmss*`_`*EADS-server-ID*`.esd`

Legend:

YYYYMMDDhhmmss: Command execution date and time

YYYY: year, *MM*: month, *DD*: day, *hh*: hour (00 through 23), *mm*: minute, *ss*: second

`eads.admin.backup.stopCommand.generation.maxNum` parameter in the shared properties, specify the maximum number of store data file generations that can be output when the `eztool stop` command is executed.

If more than this maximum number of generations of store data files are output, the most recent store data files are output and the oldest store data files are deleted.

> **Important note**
>
> If the `--no_export` option is specified in the `eztool stop` command, no data is output to a file when the EADS server is terminated.

7.6.3 Backup file-dependent parameters

The following table lists the parameters that depend on the backup files.

Table 7–8: Backup file-dependent parameters

No.	Property file	Parameter name	Value to be specified	Default value
1	Server property file	`eads.admin.backup.dir`	Path name	*management-directory*/`store`
2	Shared property file	`eads.admin.backup.exportCommand.generation.maxNum`	`0` to `32`	`2`
3		`eads.admin.backup.stopCommand.generation.maxNum`	`1` to `32`	`1`

(1) Server property file

(a) eads.admin.backup.dir

This parameter specifies the path of the output destination directory for store data files.

(2) Shared property file

(a) eads.admin.backup.exportCommand.generation.maxNum

This parameter specifies the maximum number of store data file generations that can be output when the `eztool export` command is executed.

If more store data file generations than this maximum value are output, the `eztool export` command results in an error.

If the output destination already contains a store data file with the same name, the `eztool export` command results in an error; the existing file is not overwritten.

When the value `0` is specified in this parameter, generation management of store data files is not performed.

(b) eads.admin.backup.stopCommand.generation.maxNum

This parameter specifies the maximum number of store data file generations that can be output when the `eztool stop` command is executed.

If more store data file generations than this maximum value are output, the most recent store data files are output and the oldest store data files are deleted.

7.7 Designing the cache operation-dependent parameters

This section is applicable when you will be using disk caches or two-way caches.

Design the cache operation-dependent parameters.

7.7.1 Specifying parameters for each type of cache type

The cache property parameter settings depend on the type of cache to be created.

For details about the cache types, see *2.3.1 Cache types*.

(1) Creating memory caches

Creation of cache property files is optional for memory caches.

To create a cache property file for memory caches, specify `Memory` in the `eads.cache.type` parameter.

(2) Creating disk caches or two-way caches

If you create disk caches or two-way caches, creation of cache property files is required.

In such a case, you must also specify the following cache property parameters:

- `eads.cache.type`
  To create a disk cache, specify `Disk`.
  To create a two-way cache, specify `2Way`.
- `eads.cache.disk.`*`n`*`.dir`
  See *7.7.2(2) Specifying storage locations for cache files*.
- `eads.cache.disk.filesize`
  Specify the value estimated in *4.4.1 Estimating the size and number of cache data files*.
- `eads.cache.disk.filenum`
  Specify the value estimated in *4.4.1 Estimating the size and number of cache data files*.

7.7.2 Specifying the types of cache files and their storage locations

If you will be using disk caches or two-way caches, specify the storage locations for the files that will store the cache information.

A file storing cache information is called a cache file.

When you will be using disk caches or two-way caches, you must specify the storage locations for the cache files.

(1) Types of cache files

The following subsections describe the three types of cache files.

(a) Cache data file

Cache data files store data (keys and values) that are stored in caches.

(b) Cache index file

Cache index files store indexes for the data in cache data files.

As many cache index files are created as there are cache data files.

(c) Cache information file

Cache information files store settings for caches.

One cache information file is created for each cache.

(2) Specifying storage locations for cache files

In this subsection, you specify storage locations for cache files.

You must specify storage locations for cache data files. The following table lists the storage locations for cache files.

Table 7–9: Storage locations for cache files and the file names

No.	Type of cache file	Storage location	File name
1	Cache data file	*directory-specified-in-the-eads.cache.disk.n.dir-parameter-in-the-cache-properties/cache-name*	`eads_data_[`*EADS-server-ID*`]_[`*cache-name*`]_[`*range-ID*`]_[`*nnnnn*`].ecf`
2	Cache index file	*directory-specified-in-the-eads.cache.disk.info.dir-parameter-in-the-cache-properties/cache-name*	`eads_index_[`*EADS-server-ID*`]_[`*cache-name*`]_[`*range-ID*`]_[`*nnnnn*`].ecf`
3	Cache information file	*directory-specified-in-the-eads.cache.disk.info.dir-parameter-in-the-cache-properties/cache-name*	`eads_info_[`*EADS-server-ID*`]_[`*cache-name*`].ecf`

Legend:

`[`*range-ID*`]`: Number (integer `01` to `96`) used to identify a range in a cache. The range ID matches the server ID of the EADS server on which the data is stored.

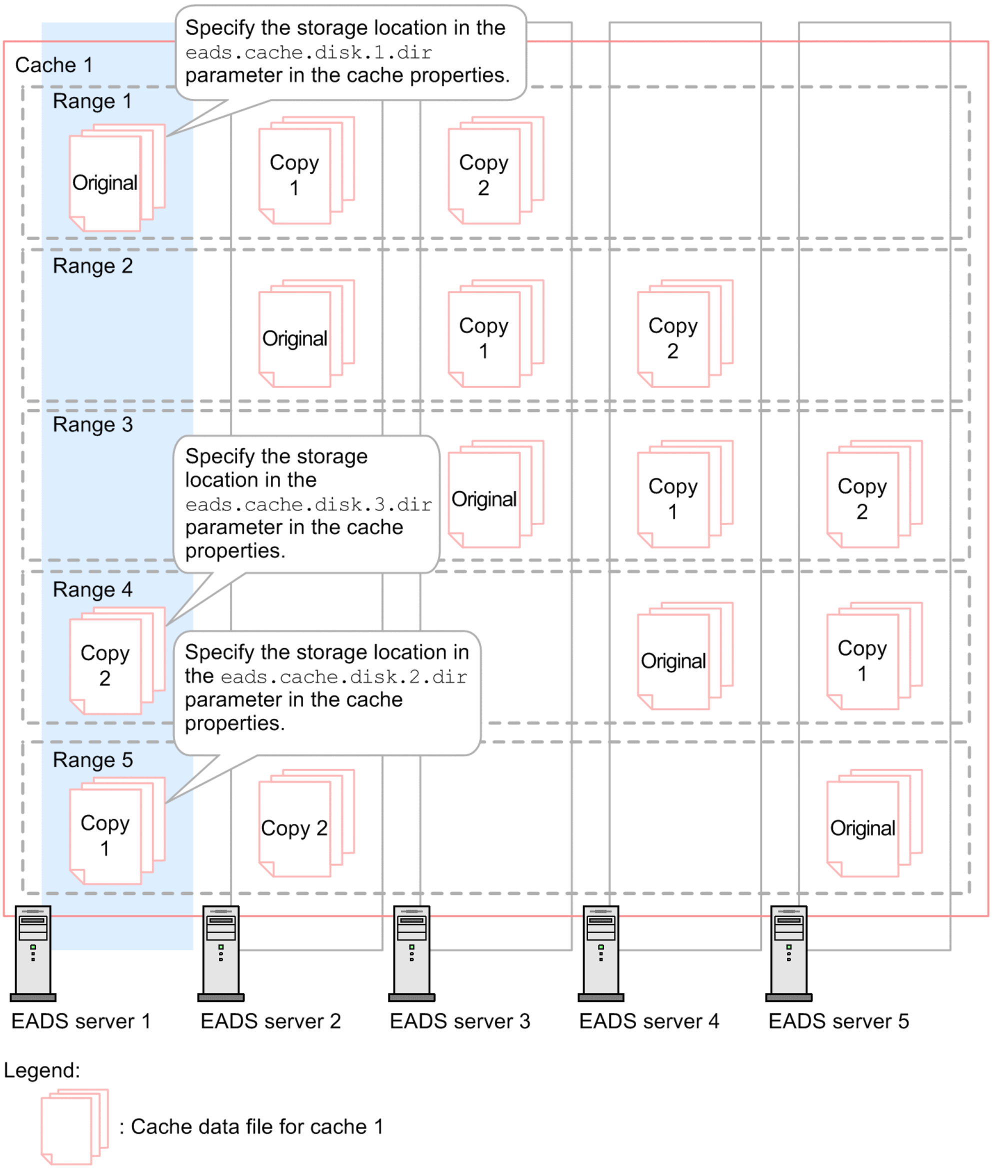
`[`*nnnnn*`]`: Sequential file number (five-digit integer)

(3) Relationship between storage locations for cache data files and EADS servers when redundant copies of data are created

Specify as many storage locations for cache data files as the multiplicity value specified in the `eads.cache.disk.`*n*`.dir` parameter. For example, if the number of redundant copies of data plus the original is set to `3`, specify a value from `1` to `3` for *n*.

The following explains the relationship between storage locations for cache data files and EADS servers when the number of redundant copies of data plus the original is set to `3`, using EADS server 1 as an example.

Figure 7–5: Relationship between storage locations for cache data files and EADS servers


Cache 1
Range 1
Original
Copy 1
Copy 2
Range 2
Original
Copy 1
Copy 2
Range 3
Original
Copy 1
Copy 2
Range 4
Copy 2
Original
Copy 1
Range 5
Copy 1
Copy 2
Original
Specify the storage location in the eads.cache.disk.1.dir parameter in the cache properties.
Specify the storage location in the eads.cache.disk.3.dir parameter in the cache properties.
Specify the storage location in the eads.cache.disk.2.dir parameter in the cache properties.
EADS server 1
EADS server 2
EADS server 3
EADS server 4
EADS server 5
Legend:
: Cache data file for cache 1

Because data is distributed in EADS, cache data files are stored in each EADS server.

The following cache data files are stored on EADS server 1:

- Cache data file that stores data in range 1 (original data)
- Cache data file that stores a copy of data in range 4
- Cache data file that stores a copy of data in range 5

Specify the storage locations for these cache data files in the `eads.cache.disk.`*`n`*`.dir` parameters in the cache properties.

Example specification of the `eads.cache.disk.`*`n`*`.dir` parameter (when using the same directory to manage the cache data files):

```
eads.cache.disk.1.dir=/hdd/cache_server01     # Storage location for the
original data
eads.cache.disk.2.dir=/hdd/cache_server01     # Storage location for the
data in range 5 (copy 1)
```

```
eads.cache.disk.3.dir=/hdd/cache_server01     # Storage location for the
data in range 4 (copy 2)
```

Example specification of the `eads.cache.disk.`*`n`*`.dir` parameter (when using different directories to manage the cache data files):

```
eads.cache.disk.1.dir=/hdd/cache_server01_range01     # Storage location
for the original data
eads.cache.disk.2.dir=/hdd/cache_server01_range05     # Storage location
for the data in range 5 (copy 1)
eads.cache.disk.3.dir=/hdd/cache_server01_range04     # Storage location
for the data in range 4 (copy 2)
```

7.7.3 Specifying the sizes of cache files

For details about specifying the sizes of cache files, see *4.4 Estimating the sizes of cache files*.

7.7.4 Cache operation-dependent parameters

The table below lists the parameters that depend on cache operations.

Bold typeface indicates a required parameter.

Table 7–10: Cache operation-dependent parameters

No.	Property file	Parameter name	Value to be specified	Default value
1	Server property file	`eads.cache.disk.getError.isolate.enable`	• `true` • `false`	`true`
2	Cache property file	**`eads.cache.type`**	• `Memory` • `Disk` • `2Way`	None
3		`eads.cache.disk.info.dir`	Path name	*management-directory*`/store`
4		**`eads.cache.disk.`*`n`*`.dir`**	Path name	None
5		**`eads.cache.disk.filesize`**	File size (`16` to `128` (megabytes))	None
6		**`eads.cache.disk.filenum`**	Number of files (`8` to `32768`)	None
7		`eads.cache.disk.blocksize`	Data size • `1` (kilobytes) • `2` (kilobytes) • `4` (kilobytes) • `8` (kilobytes) • `16` (kilobytes) • `32` (kilobytes)	`1` (kilobytes)

No.	Property file	Parameter name	Value to be specified	Default value
			- 64 (kilobytes) - 128 (kilobytes)	
8		eads.cache.disk.transfer.interval	0 to 60000 (milliseconds)	1000 (milliseconds)
9		eads.cache.disk.transfer.datasize	10240 to 2147483647 (bytes)	102400 (bytes)

Important note

The settings in the cache property files for the parameters shown below must be identical among all the EADS servers that make up the cluster. If any settings are different, caches cannot be created.

- eads.cache.type
- eads.cache.disk.filesize
- eads.cache.disk.filenum
- eads.cache.disk.blocksize

(1) Server property file

(a) eads.cache.disk.getError.isolate.enable

This parameter specifies whether the EADS server is to be isolated in the event of a disk I/O error when a disk cache or two-way cache is used.

`true`

Isolates the EADS server when a disk I/O error occurs.

`false`

Does not isolate the EADS server when a disk I/O error occurs in the following API functions, because these do not affect data integrity:

- API functions (`get`, `getAll`)
- The following methods of the `Group` interface:
  - `getLastUpdateTime()`
  - `getValueUsageSize()`
- The following methods of the `Store` interface:
  - `getLastUpdateTime()`
  - `getEHeapUsageSize()`
  - `getDiskUsageSize()`
- `eztool get` command

The API functions include those that are executed in user functions.

When `false` is specified in this parameter, the EADS server will not be isolated when a disk I/O error occurs in processing such as a `get` because of a transient disk failure. This can be expected to improve availability.

However, in the case of an application program in which `get` processing is performed for a long period of time, isolation processing will not occur even if a permanent disk failure occurs. As a result, the `get` processing might remain in error status for a long time.

This parameter is ignored when neither disk caches nor two-way caches are used.

(2) Cache property file

(a) eads.cache.type

This parameter specifies the type of cache to use.

`Memory`

Uses the memory cache.

`Disk`

Uses the disk cache.

`2Way`

Uses the two-way cache.

(b) eads.cache.disk.info.dir

This parameter specifies the storage location for the cache information files and cache index files.

The path of a storage directory can consist of a maximum of 200 characters, including alphanumeric characters (`0` to `9`, `A` to `Z`, `a` to `z`), underscores (_), colons (`:`), and separators (forward slashes (`/`)).

For details about the file names of the files that are stored under the directory specified in this parameter, see *7.7.2(2) Specifying storage locations for cache files*.

Important note

Check that the absolute path of this parameter value differs from the absolute paths of the `eads.cache.disk.`*`n`*`.dir` parameter values in the same EADS server.

(c) eads.cache.disk.n.dir

This parameter specifies the storage location for cache data files.

Specify for *n* an integer in the range from `1` to the number of redundant copies of data plus the original (`eads.replication.factor` parameter value).

If you create redundant copies of data, specify this parameter as many times as there are data copies plus the original.

The path of a storage directory can consist of a maximum of 1,024 characters, including alphanumeric characters (`0` to `9`, `A` to `Z`, `a` to `z`), underscores (_), colons (`:`), and separators (forward slashes (`/`)).

For details about the file names of the files that are stored under the directory specified in this parameter, see *7.7.2(2) Specifying storage locations for cache files*.

Important note

- Check that the absolute path of this parameter value differs from the absolute paths of the `eads.cache.disk.info.dir` parameter values in the same EADS server.
- If you specify a directory immediately under the directory specified in the `eads.cache.disk.info.dir` parameter, make sure that the specified directory name differs from the cache name.
- Disks that operate at extremely different speeds cannot be used for the EADS servers that make up a cluster. For example, SSDs and HDDs cannot be intermixed as storage locations for cache data files in the disk configuration.
- If you run multiple EADS servers on the same physical machine, specify a different storage location for each EADS server.

(d) eads.cache.disk.filesize

This parameter specifies the size (in megabytes) of the cache data files per range.

(e) eads.cache.disk.filenum

This parameter specifies the number of cache data files per range.

(f) eads.cache.disk.blocksize

This parameter specifies the amount of data (in kilobytes) that is to be written into cache data files at one time.

If the storage medium for cache data files is an HDD, specify one kilobyte (default) in this parameter. If an SSD is used, specify the page size of the SSD.

(g) eads.cache.disk.transfer.interval

This parameter specifies a data transmission interval (in milliseconds) during restoration processing on disk caches and two-way caches.

During restoration processing, this parameter value is applied to the EADS server subject to restoration processing.

Specify this parameter and the `eads.cache.disk.transfer.datasize` cache property parameter in such a manner that the following condition is satisfied:

Bandwidth available for data transmission during restoration processing (bps) ≥

MAX(*bandwidth available for data transmission during restoration processing on each cache* (bps))

MAX:

Choose the largest value in the parentheses that follow MAX.

Example: For MAX(3 × 6, 4 + 7), the calculation result is 18.

Bandwidth available for data transmission during restoration processing on each cache (bps):

Obtain the value for each cache by using the following formula.

- Memory caches

Bandwidth available for data transmission during restoration processing (bps) ≥

(*size of data transmitted during restoration processing* (bytes) × 8)

÷ {(*data transmission interval during restoration processing* (milliseconds)

+ *time required for data transmission* (milliseconds)) ÷ 1,000}

Size of data transmitted during restoration processing (bytes):

`eads.transfer.datasize` parameter value in the server properties

Data transmission interval during restoration processing (milliseconds):

`eads.transfer.interval` parameter value in the server properties

Time required for data transmission (milliseconds):

Time required to transmit the size of data specified in the `eads.transfer.datasize` server property parameter (milliseconds)

- Disk caches and two-way caches

Bandwidth available for data transmission during restoration processing (bps) ≥
(*size of data transmitted during restoration processing* (bytes) × 8)
÷ {(*data transmission interval during restoration processing* (milliseconds)
+ *time required for data transmission* (milliseconds)) ÷ 1,000}

Size of data transmitted during restoration processing (bytes):

`eads.cache.disk.transfer.datasize` parameter value in the cache properties

Data transmission interval during restoration processing (milliseconds):

`eads.cache.disk.transfer.interval` parameter value in the cache properties

Time required for data transmission (milliseconds):

Time required to transmit the amount of data specified in the `eads.cache.disk.transfer.datasize` cache property parameter (milliseconds)

This is the time required for *Restoration processing (data transmission)* in the figure in *9.3.2(5) Cluster recovery processing*. This time value depends on the environment.

As the time required for restoration processing becomes shorter, the communication workload for restoration processing increases. Conversely, as the communication workload for restoration processing decreases, the time required for restoration processing increases.

When memory caches are used, this parameter is ignored and the value of the `eads.transfer.interval` server property parameter is used.

For details about restoration processing, see *9.3.2(5) Cluster recovery processing*.

Important note

Determine the value of this parameter in such a manner that the disk's write performance is not exceeded.

(h) eads.cache.disk.transfer.datasize

This parameter specifies the size (in bytes) of the data that will be sent during restoration processing on disk caches and two-way caches.

This parameter's value is applied during restoration processing to the EADS server subject to restoration processing.

During restoration processing, the active EADS servers send data to the EADS server being restored in order to recover data consistency. Data is sent in units of 10 kilobytes at the interval specified in the `eads.cache.disk.transfer.interval` parameter until this parameter's value is reached.

When memory caches are used, this parameter is ignored and the value of the `eads.transfer.datasize` server property parameter is used.

Important note

- Because at least one data item is always sent during restoration processing, the size of the send data might exceed this parameter's value. The amount of data to be sent will not be limited if a value that is smaller than the size of the data stored in the EADS server is specified in this parameter.
- Determine a value to specify for this parameter so that the transfer speed of the restore data does not exceed the write speed of the disk.

8 Designing the Environment-Dependent Parameters (EADS Clients)

This chapter provides guidelines for designing the environment-dependent parameters for EADS clients.

8.1 Type of property file (used by EADS clients)

The table below lists and describes the type of property file used by EADS clients.

This chapter explains only the environment-dependent parameters.

Table 8–1: Type of property file (used by EADS clients)

Property file	File name	Description
Client property file	Any name#	Defines an EADS client execution environment.

#

Copy the client property file from the following directory to the directory specified in the application program and edit it as needed (make sure that the file name also matches the one specified in the application program):

- Java
  `/opt/hitachi/xeads/javaclient/conf/eads_sample_client.properties`
- C
  `/opt/hitachi/xeads/cclient/conf/eads_sample_client.properties`

8.2 Format of property files

Specify parameters in the following format:

```
parameter=value
```

How to specify parameters

- A value extends up to an end-of-line character.
- A line beginning with a hash mark (#) is treated as a comment.
- A null line is ignored.
- A value cannot be followed by a character string such as a comment. If a character string is added, the value is interpreted as being invalid.
- If the same parameter is specified more than once, the last specification takes effect. For example, `false` would take effect if the following were specified:

  ```
  eads.client.logger.message.console.enable=true
  eads.client.logger.message.console.enable=false
  ```

If the application programs are coded in C, also note the following:

- The permitted characters are `0x20` through `0x7E` in ASCII codes, the newline character (`\n`), the carriage return character (`\r`), and the tab (`\t`).
- Specify one parameter per line. One line extends through an end-of-line character (`\n` or `\r\n`) or an EOF (end of file).
- The equal sign (=), space, and tab can be used as delimiters (between a parameter name and a value).
- If two or more delimiters are specified consecutively, the second and any subsequent delimiters are interpreted as a part of the value.

  Example 1: `a=b=c`

  EADS interprets that `a` is the parameter name and `b=c` is the value.

  Example 2: `a b = c`

  EADS interprets that `a` is the parameter name and `b = c` is the value.
- If no value is specified (that is, if only a parameter name and a delimiter are specified), the value is interpreted as a null character string.
- A line containing control characters or only a parameter name is ignored.
- Although a space or a tab can be specified before and after a parameter name and a value, EADS ignores the specified space or tab when it interprets the parameter.

  Example: `Δ a Δ = Δ b Δ`

  Note: `Δ` indicates a space or a tab.

  EADS interprets that `a` is the parameter name and `b` is the value.

8.3 Designing the communication-dependent parameters

Design the communication-dependent parameters.

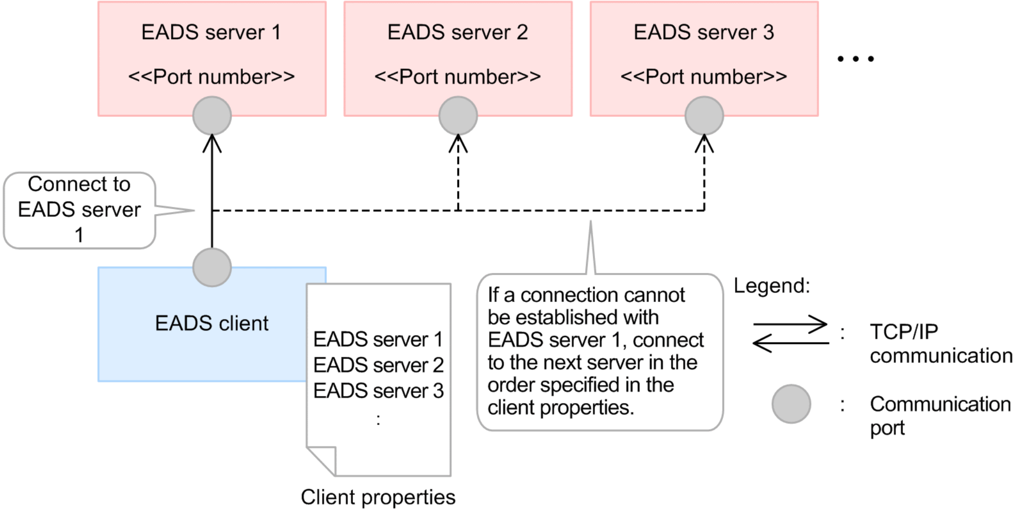
8.3.1 Specifying the connection-target EADS server, the IP address or host name, and the port number

The first time the EADS client connects to an EADS server, it selects randomly a connection target from among the EADS servers specified in the `eads.client.connect.servers` parameter in the client properties (for subsequent connections, the EADS client selects a connection target based on the cluster information acquired from the first EADS server).

If multiple EADS servers are specified in the client properties and no connection can be established with a certain EADS server, the EADS client can try to connect to another EADS server.

For details about the mechanism of communication processing, see *2.2 Mechanisms of EADS communication processing*.

Figure 8–1: Designing the communication-dependent parameters (for an EADS client)



8.3.2 Communication-dependent parameters

The table below lists the communication-dependent parameters.

Bold typeface indicates a required parameter.

Table 8–2: Communication-dependent parameters (EADS clients)

No.	Property file	Parameter name	Value to be specified	Default value
1	Client property file	**`eads.client.connect.servers`**	Name used to identify the connection-target EADS server (any name)	None

No.	Property file	Parameter name	Value to be specified	Default value
2		**`eads.client.`***connection-target-EADS-server***`.address`**	IP address or host name	None
3		`eads.client.`*connection-target-EADS-server*`.port`	Port number	`24600`

(1) Client property file

(a) eads.client.connect.servers

This parameter specifies the name (any name) used to identify the connection-target EADS server when the EADS client is initialized.

The name can consist of alphanumeric characters (`0` to `9`, `A` to `Z`, `a` to `z`).

If you specify multiple names, separate them with commas. When multiple names are specified, the first time the EADS client connects to an EADS server, it selects randomly a connection target from among the EADS servers specified in this parameter. For subsequent connections, the EADS client selects a connection target based on the cluster information acquired from the first EADS server.

If an attempt to connect to an EADS server fails, the EADS client attempts to connect to the other EADS servers one at a time in the order they were specified, starting from the one immediately following the last EADS server whose connection failed.

Important note

If multiple names are specified (delimited by the comma) and there is a name consisting of the null character string, the processing is as follows, depending on the language of the application program:

- EADS clients (Java) ignore a name consisting of a null character string and resume processing.
- EADS clients (C) treat such a name as invalid, resulting in an error.

(b) eads.client.connection-target-EADS-server.address

This parameter specifies the IP address or host name of the connection-target EADS server when the EADS client is initialized.

If you specify a host name, make sure that a unique IP address can be identified from the specified host name.

Specify this parameter paired with the `eads.client.connect.servers` parameter. For *connection-target-EADS-server*, specify the name used to identify the EADS server specified in the `eads.client.connect.servers` parameter.

(c) eads.client.connection-target-EADS-server.port

This parameter specifies the port number of the connection-target EADS server when the EADS client is initialized.

Specify this parameter paired with the `eads.client.connect.servers` parameter. For *connection-target-EADS-server*, specify the name used to identify the EADS server specified in the `eads.client.connect.servers` parameter.

(d) Example of specifications of the eads.client.connect.servers parameter

The following shows an example of specifications of the `eads.client.connect.servers` parameter.

```
eads.client.connect.servers=sv1,sv2,sv3
eads.client.sv1.address=XXX.XXX.X.138
eads.client.sv1.port=24600
eads.client.sv2.address=XXX.XXX.X.139
eads.client.sv2.port=24601
eads.client.sv3.address=XXX.XXX.X.140
eads.client.sv3.port=24602
```

8.4 Designing the log file-dependent parameters

Design the log file-dependent parameters.

8.4.1 Types of log files

The following table lists and describes the types of log files that are managed by EADS clients.

Table 8–3: Types of log files (managed by EADS clients)

No.	Log file	Description
1	Message log file	This file is used to output message logs for checking operations and monitoring for errors.
2	Maintenance log file	This file is used by the system. There is no parameter to be specified by the user.

8.4.2 Specifying the file output destinations

You can change the log file output destinations. The following table lists the file output destinations.

Table 8–4: File output destinations (EADS client)

Log file name	Output destination	File name
Message log file	*directory-specified-in-the-eads.client.logger.dir-parameter/EADS-client-name*#	`eads_client_message[`*n*`].log`
Maintenance log file	*directory-specified-in-the-eads.client.logger.dir-parameter/EADS-client-name*#`/maintenance`	`eads_client_maintenance[`*n*`].log`

Legend:

`[`*n*`]`: Sequence number of the file

#

This is the EADS client name specified in the client API function. If the EADS client name is the null character string, the subdirectory of the EADS client name is omitted.

Important note

A directory of another machine that is connected via a network cannot be specified as the file output destination. If such a directory is specified, the operation is not guaranteed.

8.4.3 Specifying the file sizes and the numbers of files

Specify the default values for the sizes of log files and the numbers of files. After you have configured the EADS client, change the parameter values, if necessary.

8.4.4 Log file-dependent parameters

The table below lists the parameters that depend on the log files.

Bold typeface indicates a required parameter.

Table 8–5: Log file-dependent parameters (EADS clients)

No.	Property file	Parameter name	Value to be specified	Default value
1	Client property file	**eads.client.logger.dir**	Path name	None
2		eads.client.logger.message.filesize	File size (4096 to 2147483647 (bytes))	1048576 (bytes)
3		eads.client.logger.message.filenum	Number of files (1 to 16)	2
4		eads.client.logger.message.console.enable	• true • false	false
5		eads.client.logger.initErrorOut	• true • false	false

(1) Client property file

(a) eads.client.logger.dir

This parameter specifies the path of the output destination directory for log files.

If the specified path does not exist, it is created.

For details about the log files output under the directory specified in this parameter, see *8.4.2 Specifying the file output destinations*.

Do not specify the same output destination at the same time in multiple processes.

(b) eads.client.logger.message.filesize

This parameter specifies the size (in bytes) of one file to which message logs are output.

(c) eads.client.logger.message.filenum

This parameter specifies the number of message log files.

(d) eads.client.logger.message.console.enable

This parameter specifies whether output of message logs to standard output is enabled.

`true`

Enables output to standard output.

`false`

Disables output to standard output.

(e) eads.client.logger.initErrorOut

This parameter specifies whether error message are to be output to the standard error output when initialization of a logger fails.

`true`

Outputs error messages to the standard output.

`false`

Does not output error messages to the standard output.

9 Designing the Tuning Parameters

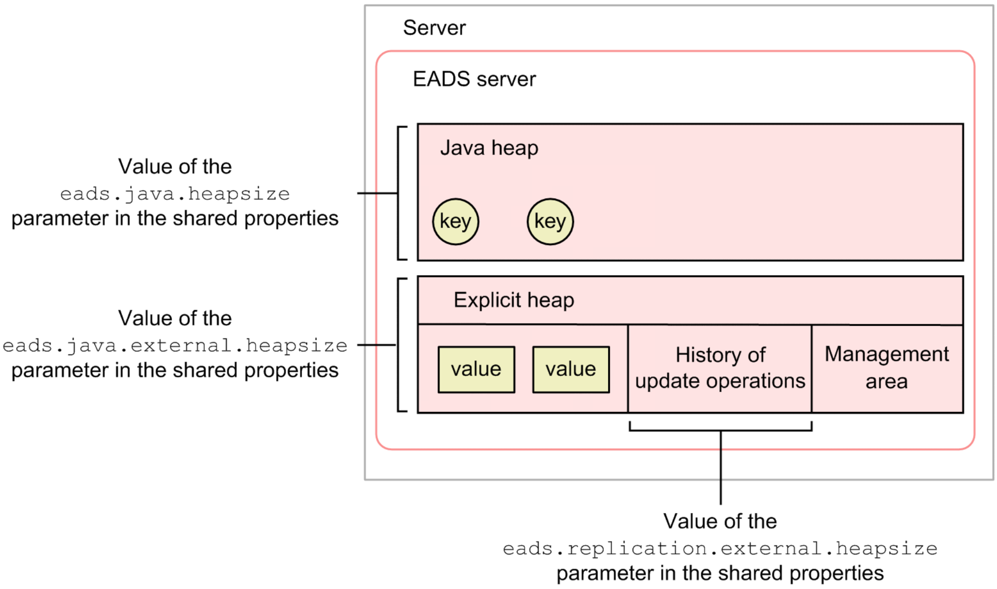
This chapter provides guidelines for designing the tuning parameters.

9.1 Designing the parameters related to memory and buffers

Design the parameters related to memory and buffers.

9.1.1 Specifying the memory sizes

The following figure shows the configuration of memory used by an EADS server.


Server
EADS server
Java heap
key
key
Explicit heap
value
value
History of update operations
Management area
Value of the eads.java.heapsize parameter in the shared properties
Value of the eads.java.external.heapsize parameter in the shared properties
Value of the eads.replication.external.heapsize parameter in the shared properties

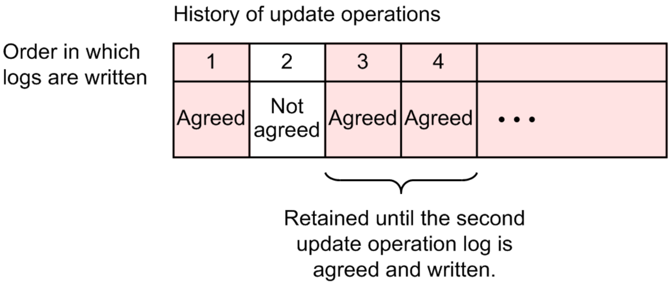
The explicit heap consists of the following areas:

- Area for storing the history of update operations
- In memory caches and two-way caches, the area for storing the value part of key-value pairs

The history of update operations includes the API functions and information about keys and values. The history of update operations is used to ensure data consistency when data is being replicated.

If a failure occurs during consensus processing, another active EADS server takes the place of the EADS server where the failure occurred, and this other active EADS server participates in the consensus processing on the basis of this history. This history log is retained until the consensus and write processing for the previous update operation is completed.

The history of update operations is deleted automatically when the EADS server determines it to be no longer needed.

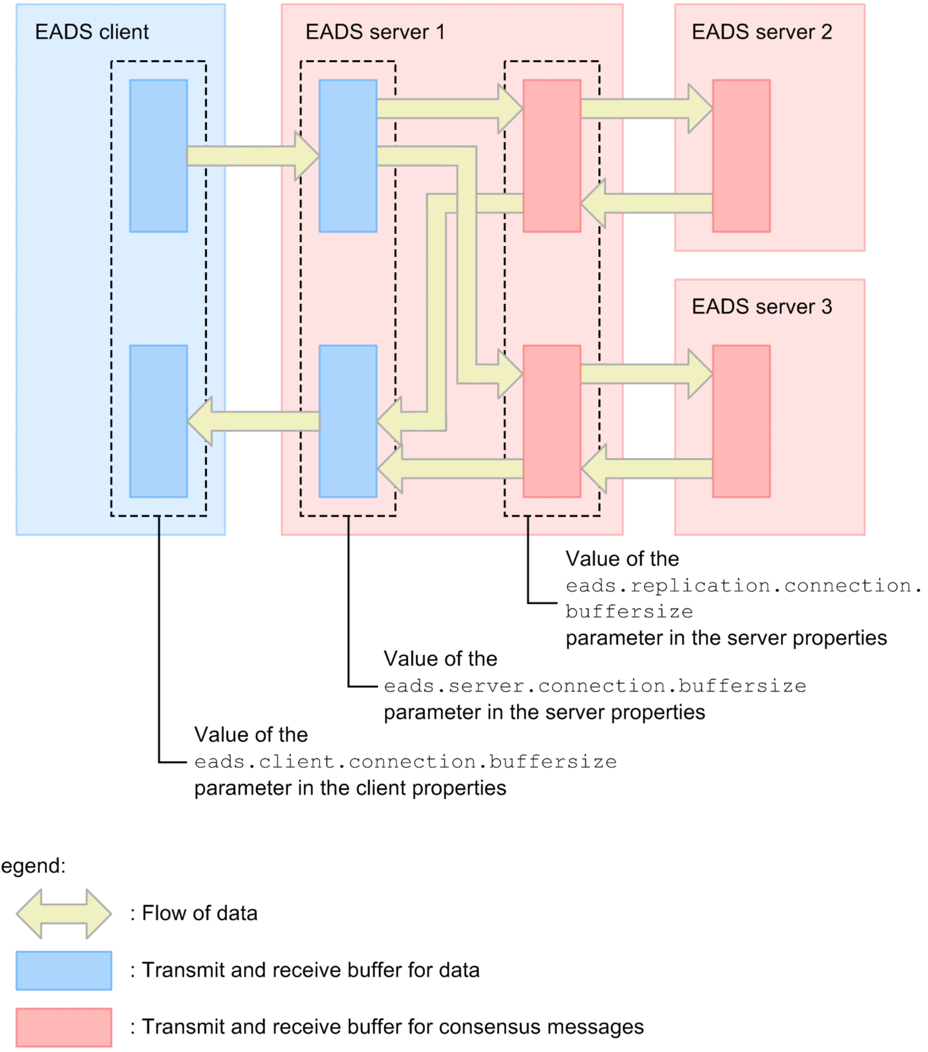

History of update operations
Order in which logs are written
1
2
3
4
Agreed
Not agreed
Agreed
Agreed
• • •
Retained until the second update operation log is agreed and written.

Approach

Determine the Java heap size and the explicit heap size per EADS server by referencing *4.1.2 Estimating the Java heap size* and *4.1.3 Estimating the explicit heap size*.

9.1.2 Specifying the buffer size

The following figure provides an overview of the buffer used for communication by the EADS servers.



Value of the `eads.replication.connection.buffersize` parameter in the server properties

Value of the `eads.server.connection.buffersize` parameter in the server properties

Value of the `eads.client.connection.buffersize` parameter in the client properties

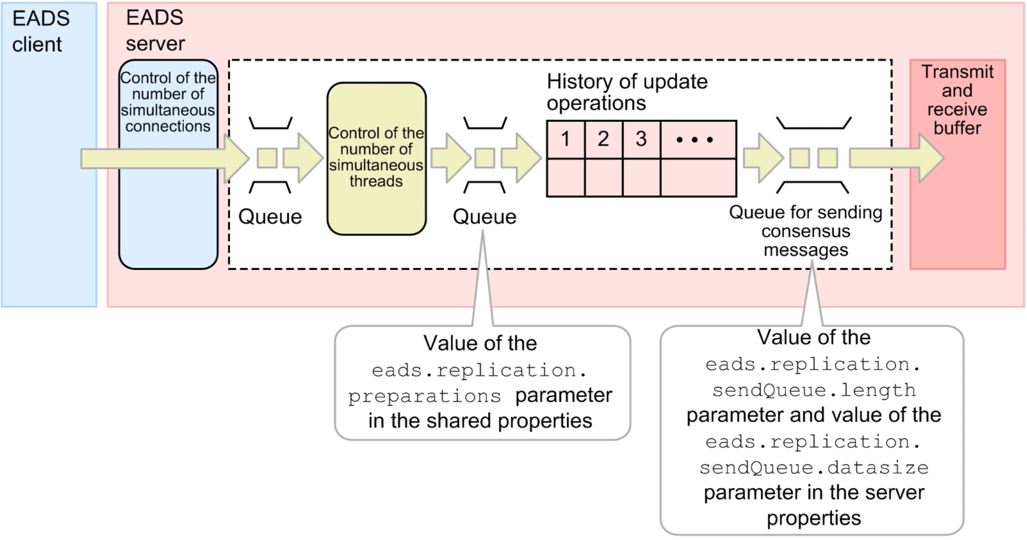
Legend:

⟺ : Flow of data

🟦 : Transmit and receive buffer for data

🟥 : Transmit and receive buffer for consensus messages

Requests sent from the EADS client are controlled in the EADS server as shown below.


EADS client
EADS server
Control of the number of simultaneous connections
Queue
Control of the number of simultaneous threads
Queue
History of update operations
1
2
3
• • •
Queue for sending consensus messages
Transmit and receive buffer
Value of the eads.replication.preparations parameter in the shared properties
Value of the eads.replication.sendQueue.length parameter and value of the eads.replication.sendQueue.datasize parameter in the server properties

Requests (data update operations) sent from the EADS client are retained in the history, and then queued for sending consensus messages. Queues for sending consensus messages are allocated for the number of EADS servers to which the consensus messages will be sent.

You can also use the `eads.replication.sendQueue.length` parameter in the server properties to specify the length of the queue for sending consensus messages. Use the `eads.replication.sendQueue.datasize` parameter in the server properties to specify the maximum amount of data that can be stored in the send queue.

You can use the `eads.replication.preparations` parameter in the shared properties to specify the maximum number of consensus processes that can be executed simultaneously.

For details about how to specify the maximum number of simultaneous connections and the maximum number of simultaneous threads, see *9.2 Designing the parameters related to thread pools and connection pools*.

Approach

You can send and receive data efficiently by adjusting the buffer size according to the amount of data handled.

9.1.3 Parameters related to memory and buffers

The following table lists the parameters related to memory and buffers.

Table 9–1: Parameters related to memory and buffers

No.	Property file	Parameter name	Value to be specified	Default value
1	Server property file	`eads.server.connection.buffersize`	Buffer size (`1024` to `16777216` (bytes))	`4096` (bytes)
2		`eads.replication.connection.buffersize`	Buffer size (`1024` to `16777216` (bytes))	`131071` (bytes)
3		`eads.replication.sendQueue.length`	Queue size (`10000` to `1000000`)	`100000`

No.	Property file	Parameter name	Value to be specified	Default value
4		eads.replication.sendQueue.datasize	1048576 to 2147483647 (bytes)	16777216 (bytes)
5		eads.transfer.connection.buffersize	Buffer size (1024 to 16777216 (bytes))	131071(bytes)
6		eads.transfer.datasize	10240 to 2147483647 (bytes)	1048576 (bytes)
7		eads.replication.fillgap.copy.datasize	1024 to 16777216 (bytes)	10240 (bytes)
8		eads.admin.operation.resume.send.datasize	0 to 2147483647 (bytes)	1048576 (bytes)
9	Shared property file	eads.cache.key.maxsize	1 to 1024 (bytes)	1024 (bytes)
10		eads.replication.preparations	1 to 100	20
11		eads.replication.external.heapsize	1 to 268435456 (megabytes)	450 (megabytes)
12		eads.java.heapsize	Heap size#1	3072 (megabytes)
13		eads.java.external.heapsize	Heap size (2 to 2147483647 (megabytes))	1024 (megabytes)
14		eads.java.permanent.maxsize	Memory size#2	83 (megabytes)
15		eads.cache.limiter.enable	- true - false	true
16		eads.cache.keyCount	1024 to 1073741824	1048576
17	Client property file	eads.client.connection.buffersize	Buffer size (1024 to 16777216 (bytes))	4096 (bytes)

#1

The specified values are applied to JavaVM's heap size options (-Xmx and -Xms).

#2

The specified values are applied to JavaVM's memory size options (-XX:PermSize and -XX:MaxPermSize).

(1) Server property file

(a) eads.server.connection.buffersize

This parameter specifies the size (in bytes) of the work area buffer that is allocated each time connection is established and used to send and receive data.

(b) eads.replication.connection.buffersize

This parameter specifies the size (in bytes) of the transmit and receive buffer for consensus messages.

We recommend that you specify the TCP window size that is specified in the OS.

Depending on the OS, a buffer of a size that is different from the value of this parameter might be used.

(c) eads.replication.sendQueue.length

This parameter specifies the size of the consensus message send queue.

As many send queues are allocated as there are EADS servers to which consensus messages are to be sent.

Use the following formula to estimate the queue length. If the estimate result is smaller than the minimum value of this parameter, set the minimum value.

$2 \times a \times b + 2 \times a \times \textit{numbers of caches} + (c - 1) \times a \times \textit{numbers of caches}$

- *a*: `eads.replication.preparations` parameter value in the shared properties
- *b*: `eads.replication.factor` parameter value in the shared properties minus 1 (the value is 1 if the number of redundant copies of data plus the original is 1)
- *c*: `eads.replication.factor` parameter value in the shared properties

Important note

If consensus messages exceeding the specified queue length enter the send queue, a communication error occurs.

(d) eads.replication.sendQueue.datasize

This parameter specifies the maximum amount (in bytes) of data that can be stored in a queue for sending consensus messages.

Use the following formula to obtain the value you can specify:

$2 \times a \times b \times (c + \textit{numbers of caches}) + d \times (e - 1) \times \textit{numbers of caches}$

- *a*: `eads.replication.preparations` parameter value in the shared properties
- *b*: Maximum size of the history of update operations
  The following shows the formula for estimating the maximum size (in bytes) of the history of update operations.

  eads.cache.key.maxsize parameter value in the shared properties + *maximum value size*# × MAX(2, *maximum number of data items that can be updated simultaneously*)

  #

  Maximum size that can be specified when `put`, `create`, `update`, or `replace` processing is performed.

  MAX:

  Choose the largest value in the parentheses that follow MAX.

  Example: For MAX(2, 10), the calculation result is 10.

  Maximum number of data items that can be updated simultaneously:

  If you will be performing batch data operations using a memory cache, specify `10`. Otherwise, specify `1`.
- *c*: `eads.replication.factor` parameter value in the shared properties minus 1 (the value is 1 if the number of redundant copies of data plus the original is 1)
- *d*: `eads.replication.fillgap.copy.datasize` parameter value in the server properties
- *e*: `eads.replication.factor` parameter value in the shared properties

Important note

If you specify an excessively small value compared to the estimate result, retries might occur frequently due to send queue overflows, and further processing might be disabled.

(e) eads.transfer.connection.buffersize

This parameter specifies the size (in bytes) of the data transmit and receive buffer that is used during restoration processing and scale-out processing.

We recommend that you specify the TCP window size that is specified in the OS.

Depending on the OS, a buffer of a size that is different from the value of this parameter might be used.

(f) eads.transfer.datasize

This parameter specifies the size (in bytes) of data that will be transmitted during restoration processing and scale-out processing.

During restoration processing and scale-out processing, this parameter value is applied to the EADS server subject to restoration processing and to the EADS servers added during scale-out processing.

During restoration processing and scale-out processing, the active EADS servers send data to the EADS server being restored and the EADS servers added by scale-out processing in order to recover data consistency. Data is sent in units of 10 kilobytes at the interval specified in the `eads.transfer.interval` parameter until this parameter value is reached.

For details about restoration processing, see *9.3.2(5) Cluster recovery processing*. For details about scale-out processing, see *9.3.2(6) Cluster scale-out processing (adding EADS servers)*.

When disk caches and two-way caches are restored, this parameter is ignored and the value of the `eads.cache.disk.transfer.datasize` cache property parameter is restored.

Important note

Because at least one data item is always sent during restoration processing and scale-out processing, the amount of data to be sent will not be limited if a value that is smaller than the size of the data stored in the EADS server is specified in this parameter.

(g) eads.replication.fillgap.copy.datasize

This parameter specifies the amount (in bytes) of data that is sent each time data is copied to the EADS server during complementary processing of history of update operations.

For details about the complementary processing of history of update operations, see *9.3.2(7) Complementary processing of the history of update operations*.

Use the following formula to obtain the value you can specify:

maximum size of the history of update operations × *maximum number of consensus processes that can be executed simultaneously*

Maximum size of the history of update operations:

The following shows the formula for estimating the maximum size (in bytes) of the history of update operations.

eads.cache.key.maxsize *parameter value in the shared properties* + *maximum value size*# × MAX(2, *maximum number of data items that can be updated simultaneously*)

#

Maximum size that can be specified when put, create, update, or replace processing is performed.

MAX:

Choose the largest value in the parentheses that follow MAX.

Example: For MAX(2, 10), the calculation result is 10.

Maximum number of data items that can be updated simultaneously:

If you will be performing batch data operations using a memory cache, specify 10. Otherwise, specify 1.

Maximum number of consensus processes that can be executed simultaneously:

eads.replication.preparations parameter value in the shared properties

Important note

- Do not specify an excessively large value, as that would adversely affect memory usage and CPU usage rate.
- A larger amount of data than the amount specified by this parameter might be sent.
- Even if you specify a small data amount, complementary processing is performed for at least one history item.

(h) eads.admin.operation.resume.send.datasize

This parameter specifies the size (in bytes) of differential data that is transferred at one time when the eztool resume command is executed.

If 0 is specified, one data item is transmitted at a time.

(2) Shared property file

(a) eads.cache.key.maxsize

This parameter specifies the maximum size (in bytes) of any key that can be stored in the cluster.

The key storage area is allocated based on the size specified in this parameter.

By restricting the maximum key size, you can design a smaller Java heap size for storing keys compared to when the default key size is used.

Important note

- A cluster whose maximum key size is set to a value smaller than that of another cluster might not be able to import store data files that are output from that other cluster.
- If disk caches or two-way caches are used, cache data files and cache index files that have been output in a cluster in which the maximum key size is set to a large value might not be usable for data relocation in a cluster in which the maximum key size is set to a smaller value.

(b) eads.replication.preparations

This parameter specifies the maximum number of consensus processes that can be performed simultaneously.

Specify the smaller of the values shown below. Note, however, that if the values are different between EADS servers, specify the largest value in the cluster.

- `eads.server.maxConnections` parameter value in the server properties
- `eads.server.cache.maxExecuteThreads` parameter value in the server properties + `eads.server.function.maxExecuteThreads` parameter value in the server properties

(c) eads.replication.external.heapsize

This parameter specifies the size (in megabytes) of the area for storing the history of update operations.

Be aware that if this parameter's value is greater than the *eads.java.external.heapsize parameter value in the shared properties* × 0.97 (in units of megabytes; digits to the right of the decimal point are discarded), EADS server startup processing will fail.

(d) eads.java.heapsize

This parameter specifies the size (in megabytes) of the Java heap in which keys are stored.

(e) eads.java.external.heapsize

This parameter specifies the size (in megabytes) of the explicit heap in which values and the history of update operations are stored.

Note that three percent of the specified explicit heap size is used as a management area (the value is rounded up in megabytes).

Be aware that if the `eads.replication.external.heapsize` parameter value in the shared properties is greater than *this parameter value* × 0.97 (in units of megabytes; digits to the right of the decimal point are discarded), EADS server startup processing will fail.

(f) eads.java.permanent.maxsize

This parameter specifies the size (in megabytes) of the `Permanent` area.

The `Permanent` area is used to store such information as the loaded EADS servers and the classes of user functions.

(g) eads.cache.limiter.enable

This parameter specifies whether the total data restriction function is to be enabled.

When the total data restriction function is enabled and a shortage of space at the data storage location is foreseen, the EADS server can be protected from being shut down by setting an error in the corresponding processing.

`true`

Enables the total data restriction function.

`false`

Disables the total data restriction function.

(h) eads.cache.keyCount

This parameter specifies the number of data items per range that will be monitored by the total data restriction function.

If `false` is specified in the `eads.cache.limiter.enable` parameter (total data restriction function is disabled), this parameter's value is ignored.

(3) Client property file

(a) eads.client.connection.buffersize

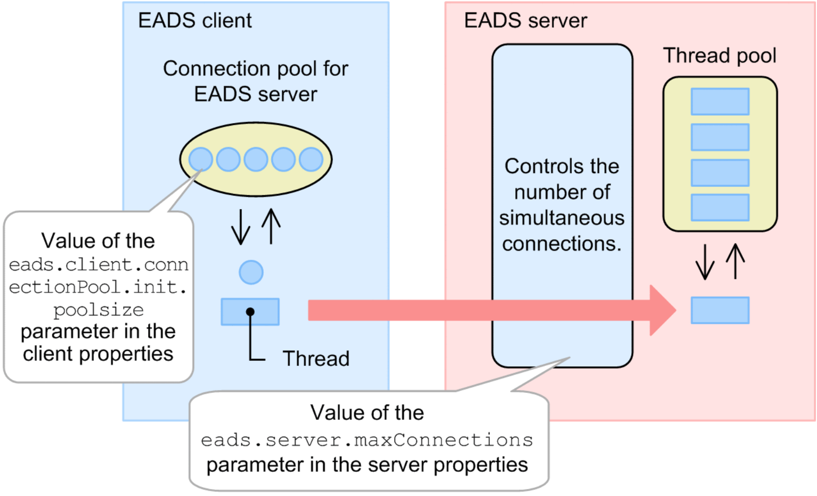
This parameter specifies the size (in bytes) of the buffer that will be used by the EADS client to send and receive data.

9.2 Designing the parameters related to thread pools and connection pools

Design the parameters related to thread pools and connection pools according to the number of request processes that are executed simultaneously.

9.2.1 Setting the maximum number of simultaneous connections

EADS reduces the overhead of generating threads and connections and improves throughput by using thread pools and connection pools. For details about thread pools and connection pools, see *2.12 Improving throughput by using thread and connection pools*.


EADS client
Connection pool for EADS server
Value of the eads.client.connectionPool.init.poolsize parameter in the client properties
Thread
EADS server
Controls the number of simultaneous connections.
Thread pool
Value of the eads.server.maxConnections parameter in the server properties

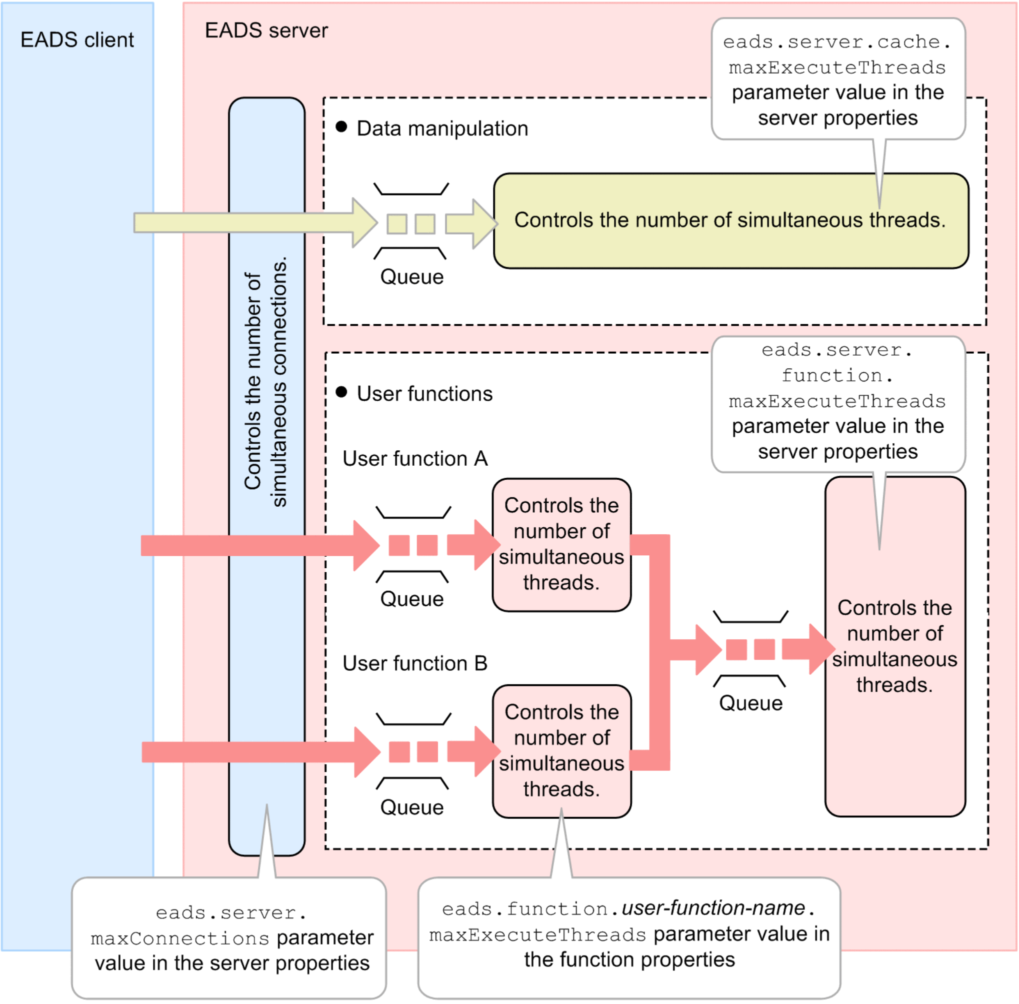
Approach

You can improve resource utilization efficiency by adjusting the maximum number of simultaneous connections (the `eads.server.maxConnections` parameter value in the server properties).

9.2.2 Setting the maximum number of simultaneous threads

The EADS server controls the number of simultaneous connections from the EADS client and the number of simultaneous threads.

For the number of simultaneous threads, the EADS server controls the requests for performing data operations separately from the requests for user functions.


EADS client
EADS server
eads.server.cache.maxExecuteThreads parameter value in the server properties
Data manipulation
Queue
Controls the number of simultaneous threads.
Controls the number of simultaneous connections.
eads.server.function.maxExecuteThreads parameter value in the server properties
User functions
User function A
Controls the number of simultaneous threads.
Queue
User function B
Controls the number of simultaneous threads.
Queue
Queue
Controls the number of simultaneous threads.
eads.server.maxConnections parameter value in the server properties
eads.function.*user-function-name*.maxExecuteThreads parameter value in the function properties

- Number of simultaneous threads for data operation

  This value is specified in the `eads.server.cache.maxExecuteThreads` parameter in the server properties.

  Any request exceeding the maximum number of simultaneous threads is queued until the connection between the EADS client and EADS server is released (the `eads.server.connection.keepAlive.timeout` parameter value in the server properties).

- Number of simultaneous threads for user functions

  The EADS server controls the number of simultaneous threads for each user function (`eads.function.`*user-function-name*`.maxExecuteThreads` parameter in the function properties) and the number of simultaneous threads for all user functions (`eads.server.function.maxExecuteThreads` parameter in the server properties).

Approach

You can improve resource utilization efficiency by adjusting the maximum number of simultaneous threads.

9.2.3 Parameters related to thread pools and connection pools

The following table lists the parameters related to thread pools and connection pools.

Table 9–2: Parameters related to thread pools and connection pools

No.	Property file	Parameter name	Value to be specified	Default value
1	Server property file	**eads.server.maxConnections**	1 to 1024	10
2		eads.server.cache.maxExecuteThreads	1 to the eads.server.maxConnections parameter value	eads.server.maxConnections parameter value
3		eads.server.function.maxExecuteThreads	1 to the eads.server.maxConnections parameter value	eads.server.maxConnections parameter value
4	Function property file	eads.function.*user-function-name*.maxExecuteThreads	0 to the eads.server.function.maxExecuteThreads parameter value	0
5	Client property file	eads.client.connectionPool.poolsize	1 to 1024	10
6		eads.client.connectionPool.exceedMaxSizeError.enable	• true • false	false

(1) Server property file

(a) eads.server.maxConnections

This parameter specifies the maximum number of simultaneous connections to the EADS server.

If an attempt is made to establish more connections than the specified maximum number of simultaneous connections, an error is returned and communication is closed.

Specify at least the sum of the eads.client.connectionPool.poolsize parameter values specified in the client properties of the EADS clients that will be connected.

Note that the eads.client.connectionPool.poolsize parameter's value is also set in backlog of the Listen queue. If the value exceeds an OS limitation, the OS's limit value is set.

Extend the OS's limit value taking into account the value of the eads.client.connectionPool.poolsize parameter in the client properties. For details about how to extend limit values, see the OS documentation.

(b) eads.server.cache.maxExecuteThreads

This parameter specifies the maximum number of simultaneous threads for performing data operations.

(c) eads.server.function.maxExecuteThreads

This parameter specifies the maximum number of simultaneous threads for user functions.

(2) Function property file

(a) eads.function.user-function-name.maxExecuteThreads

This parameter specifies the maximum number of simultaneous threads for each user function.

For a user function name, specify a fully qualified class name.

If zero is specified, there is no limit to the number of simultaneous threads.

(3) Client property file

(a) eads.client.connectionPool.poolsize

This parameter specifies the maximum number of connections to be pooled for the same connection target.

As many connections as the value specified in this parameter can be established.

Specify the number of threads that can execute client API functions simultaneously.

(b) eads.client.connectionPool.exceedMaxSizeError.enable

This parameter specifies whether a request from an EADS client is to result in an error when the maximum number of connections to be pooled for the same connection target is reached and all are in use.

`true`

The client API function is to result in an error.

`false`

The client API function is not to result in an error. Instead, the client API function is to be placed in wait status.

9.3 Designing the timeout-related parameters

This section explains how to design the timeout-related parameters.

9.3.1 Setting the timers for monitoring communication

For communications between the EADS clients and EADS servers that use TCP, EADS detects communication errors by monitoring the following durations:

- Duration from the start of connection to its completion, using a socket
- Duration from the start of a data write operation to its completion
- Duration from the start of a data read operation to its completion

Multiple data read operations might occur when a single message is received.

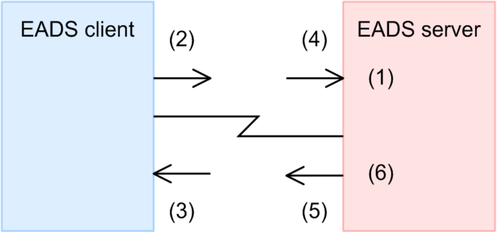
Approach

EADS detects communication errors more quickly by shortening the monitoring interval, and it prevents timeouts from occurring frequently by increasing the monitoring interval.

(1) Specifying timeout values for communication between EADS client and EADS server

You can specify timeout values for communication between EADS client and EADS server as shown in the following figure.

Figure 9–1: Specifying timeout values for communication between EADS client and EADS server



The table below lists the parameters used for specifying communication timeout values. The numbers 1 to 6 in the table correspond to the numbers in *Figure 9-1 Specifying timeout values for communication between EADS client and EADS server*.

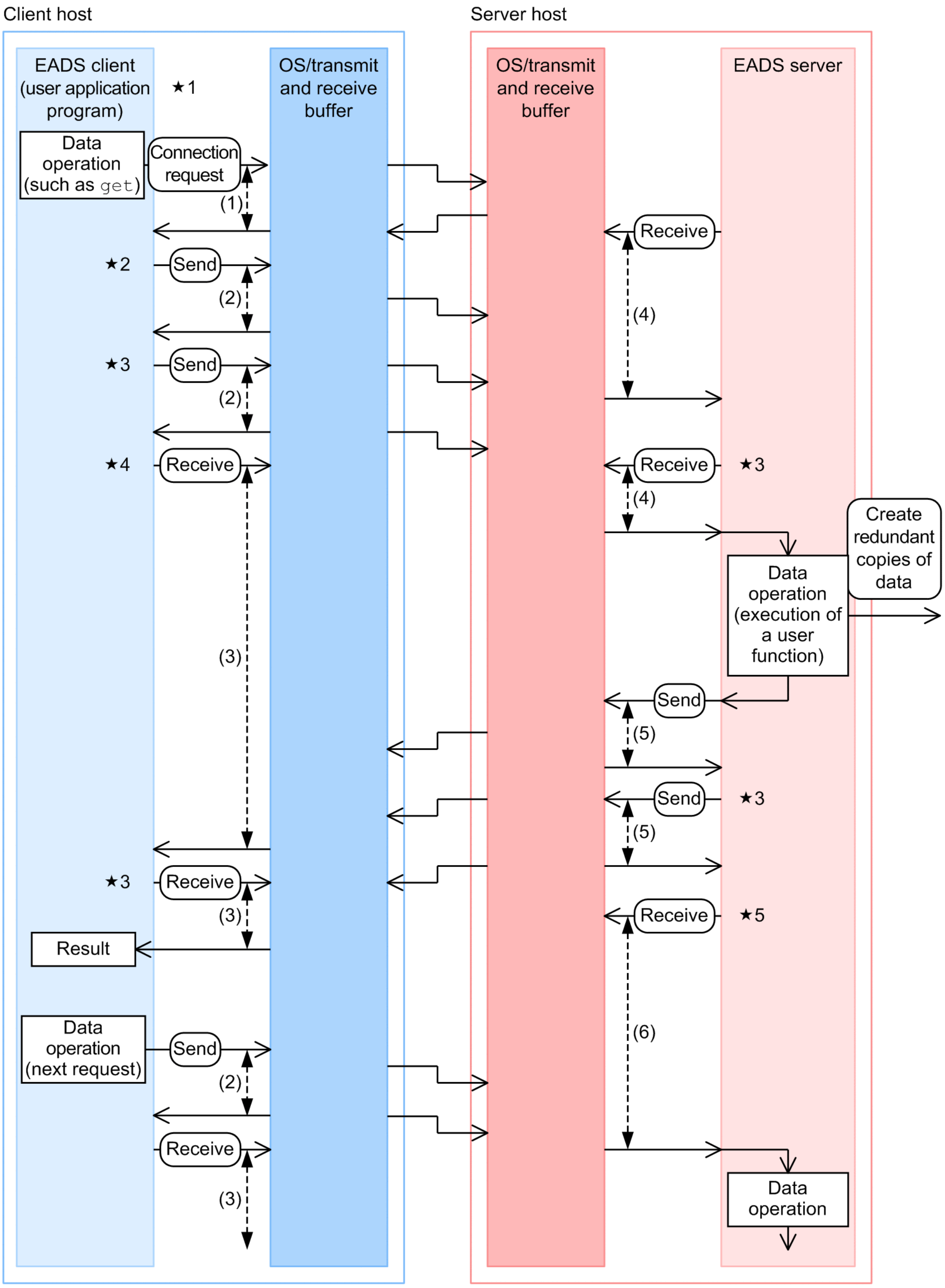
Table 9–3: Parameters used for specifying communication timeout values

No.	Timeout value to be specified	Property file	Parameter name
1	Connection to the EADS server	Client Property file	`eads.client.connection.send.timeout`
2	Data transmission to the EADS server		
3	Data reception from the EADS server		`eads.client.connection.receive.timeout`

No.	Timeout value to be specified	Property file	Parameter name
4	Data reception from the EADS client	Server Property file	`eads.server.connection.timeout`
5	Data transmission to the EADS client		
6	Closing an idle permanent connection		`eads.server.connection.keepAlive.timeout`

The following figure shows the locations at which timeout values can be specified for communication between EADS client and EADS server.

Figure 9–2: Locations at which timeout values can be specified for communication between EADS client and EADS server


Client host
Server host
EADS client (user application program)
★1
OS/transmit and receive buffer
OS/transmit and receive buffer
EADS server
Data operation (such as get)
Connection request
(1)
Receive
★2
Send
(2)
(4)
★3
Send
(2)
★4
Receive
Receive
★3
(4)
Create redundant copies of data
Data operation (execution of a user function)
(3)
Send
(5)
Send
★3
(5)
★3
Receive
(3)
Receive
★5
Result
(6)
Data operation (next request)
Send
(2)
Receive
(3)
Data operation

Details of specifying timeout values ((1) to (6) in the figure) are explained in the following subsections:

- *9.3.1(2) Approach to specifying communication timeout values*
- *9.3.1(3) Specifying a timeout value for closing a permanent connection*

In addition, tips (★ 1 to ★ 5) are explained in *9.3.1(4) Tips for considering the timeout values to be specified*.

(2) Approach to specifying communication timeout values

The communication timeout values indicated by 1 to 5 in *Figure 9-2 Locations at which timeout values can be specified for communication between EADS client and EADS server* are used to detect the following events, not as markers for when the corresponding processing is to be completed:

- Physical closure of a channel
- Closure of a channel to the communication target for a reason such as a failure at the communication target

We do not recommend that you use the communication timeout function for closing the connection, even when no response is returned within the expected time.

The EADS server processing continues regardless of the status of the communication channel to the EADS client. Data is not rolled back when channel closure is detected when a response is sent to the EADS client. Therefore, if the communication timeout function is used to close communication, the EADS client cannot obtain the data operation results.

We recommend that you tune the communication timeout values so that failures at the communication target and channel closure will be detected correctly, rather than using them to detect delays in the processing and the network.

(3) Specifying a timeout value for closing a permanent connection

A timeout value for closing a permanent connection (6 in *Figure 9-2 Locations at which timeout values can be specified for communication between EADS client and EADS server*) is used to prevent the connection and thread from becoming unavailable until the TCP keep-alive idle time set in the OS is reached when the EADS server cannot detect a shutdown of the OS or host on which the application program (user program) is running.

We recommend that you consider the following two points and specify a sufficient value within a range that will not affect other functions and that will avoid unnecessary communication closure and errors.

- Relationship with the number of connections on the EADS client

  The EADS client maintains permanent connections even when there is no operation request from the application program to the EADS server for a specified period of time. Permanent connections are used for periodic checking of cluster information. The maximum number of permanent connections per EADS server is specified in the `eads.client.connectionPool.poolsize` parameter in the client properties. The permanent connections are used at the interval specified for each EADS server in the `eads.client.clusterInfo.update.interval` parameter in the client properties.

  Only one thread is used to check the cluster information. If an attempt is made to check the cluster information at the address of a host whose OS is not running normally, the processing will be placed in wait status for a maximum of the duration specified in the `eads.client.connection.send.timeout` parameter in the client properties. During this time, communication with other EADS servers for checking the cluster information will be placed on hold.

  In such a situation, to prevent permanent connections from being closed when there is no request from the EADS client, specify a sufficient value in the `eads.server.connection.keepAlive.timeout` parameter in the server properties that can satisfy the following condition:

eads.server.connection.keepAlive.timeout parameter value in the server properties
> (eads.client.connection.send.timeout parameter value in the client properties × *permissible number of EADS server failures*
+ eads.client.clusterInfo.update.interval parameter value in the client properties)
× *eads.client.connectionPool.poolsize parameter value in the client properties*

  If the EADS server process is shut down but there is no problem on the OS, wait status does not last as long as the duration specified in the above `eads.client.connection.send.timeout` parameter. Therefore, specify the permissible number of EADS server failures taking into account permissible failures such as host failures, not process failures.

- Relationship with the maximum number of connections on the EADS server

  If the `eads.server.maxConnections` parameter is specified in the server properties, the number of connections from the EADS client is limited and any connection attempted beyond the specified value will result in an error.

  If the minimum value is specified in this parameter and a network disconnection that cannot be detected from the EADS server's network occurs due to a host or OS error, the EADS server's connection might be maintained until the time specified in the `eads.server.connection.keepAlive.timeout` parameter in the server properties is reached. As a result, the EADS client might not be able to reestablish connection after error recovery processing. To resolve this problem, specify a parameter value that satisfies the following condition:

eads.server.connection.keepAlive.timeout parameter value in the server properties
< time required for the EADS client to reestablish connection after a failure

Note that the handling time depends greatly on the nature of the failure. If there is sufficient memory, we recommend that you add the number of connections that will be used for reconnection to the `eads.server.maxConnections` parameter value in advance and specify a sufficient value, such as 3600 seconds, in the `eads.server.connection.keepAlive.timeout` parameter.

(4) Tips for considering the timeout values to be specified

This subsection provides tips for considering the timeout values to be specified. These tips correspond to ★ 1 to ★ 5 in Figure 9-2.

Tip 1 (★ 1)

Normally, a permanent connection is used for communication from the EADS client. Therefore, connection processing occurs only when connection is established for the first time and when the number of threads that use client libraries increases. For the connection timeout value, the same value is used as for the transmission timeout value (`eads.client.connection.send.timeout` parameter value in the client properties).

Tip 2 (★ 2)

Data send processing is treated as being complete when data has been stored in the sender's send buffer. Therefore, data send processing succeeds regardless of the statuses of the receive process and channel as long as the data fits in the send buffer (if there is a problem in the channel, the next receive processing results in an error).

If an attempt is made to send data that is larger than the size of the send buffer, a send error might occur due to a problem (such as `FullGC`) at the receiving process. If you will be sending data that is larger than the send buffer size, take into account the time required for processing events such as the receiving process's `FullGC`.

Tip 3 (★ 3)

Send and receive processing takes place internally in multiple segments. The timeout values are applied to each segment. Therefore, the timeout values are not for guaranteeing the length of time before replies are returned.

Tip 4 (★ 4)

The EADS client performs the next read processing when its current send processing is completed. The EADS client does not consider whether the EADS server has received the data that the EADS client sent.

The EADS client's receive timeout includes the time for performing data operations and executing user functions on the EADS server. If there is a user function that requires a long time to process and the client properties are specified to accommodate such a user function's processing time, the value might not be suitable for normal communication processing. In such a case, we recommend that you use API functions with timeout settings instead of specifying the timeout values in the client properties.

Tip 5 (★ 5)

After sending a response to the EADS client, the EADS server waits for the next request. For this receive processing, the timeout value for permanent connections is used (`eads.server.connection.keepAlive.timeout` parameter value specified in the server properties), not the normal communication timeout value (`eads.server.connection.timeout` parameter value specified in the server properties). For details about the timeout value for permanent connections, see *9.3.1(3) Specifying a timeout value for closing a permanent connection*.

9.3.2 Setting the timers for monitoring the cluster

The EADS servers in the cluster mutually send heartbeats to notify one another that they are operating normally within the cluster.

EADS servers also detect communication errors by monitoring the communication with the EADS servers during command execution and the time spent from start to completion of command execution.

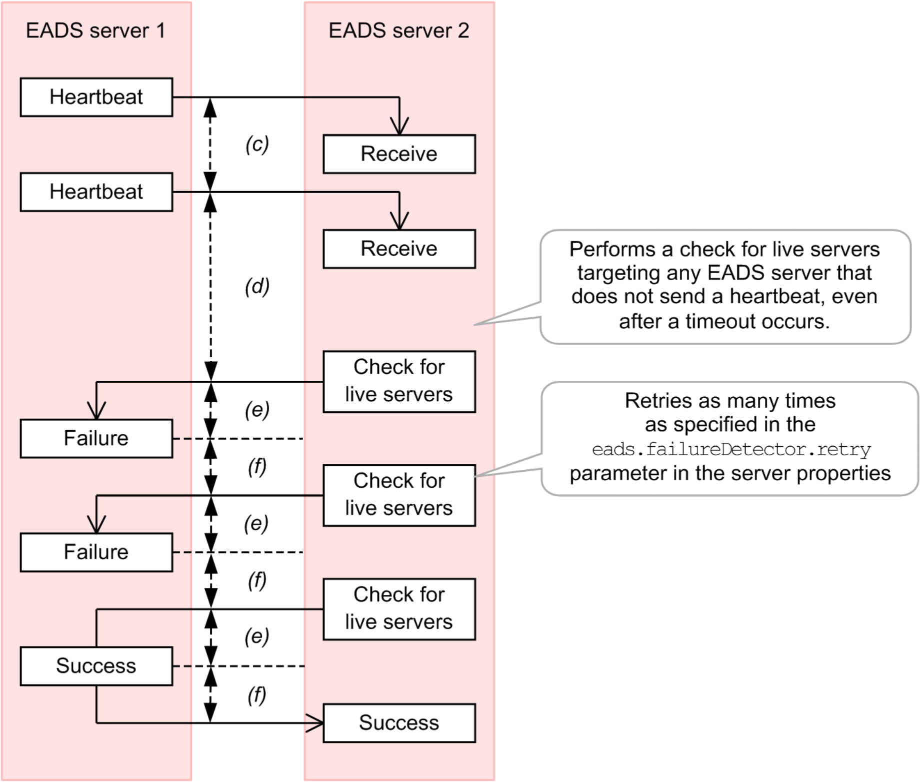
Approach

EADS detects communication errors more quickly by shortening the monitoring interval, and prevents timeouts from occurring frequently by increasing the monitoring interval.

(1) Sending heartbeats and checking for live servers

For details about monitoring the cluster by sending heartbeats, see *2.9 Monitoring a cluster*.

The following figure shows the timers for heartbeat transmission and the check for live servers.


EADS server 1
EADS server 2
Heartbeat
Receive
(c)
Heartbeat
Receive
(d)
Performs a check for live servers targeting any EADS server that does not send a heartbeat, even after a timeout occurs.
Check for live servers
(e)
Failure
(f)
Retries as many times as specified in the eads.failureDetector.retry parameter in the server properties
Check for live servers
(e)
Failure
(f)
Check for live servers
(e)
Success
(f)
Success

The alphabetical letters in the figure correspond to explanations provided in the following subsections of *9.3.3 Timeout-related parameters*:

(c): *9.3.3(1)(c) eads.failureDetector.heartbeat.interval*

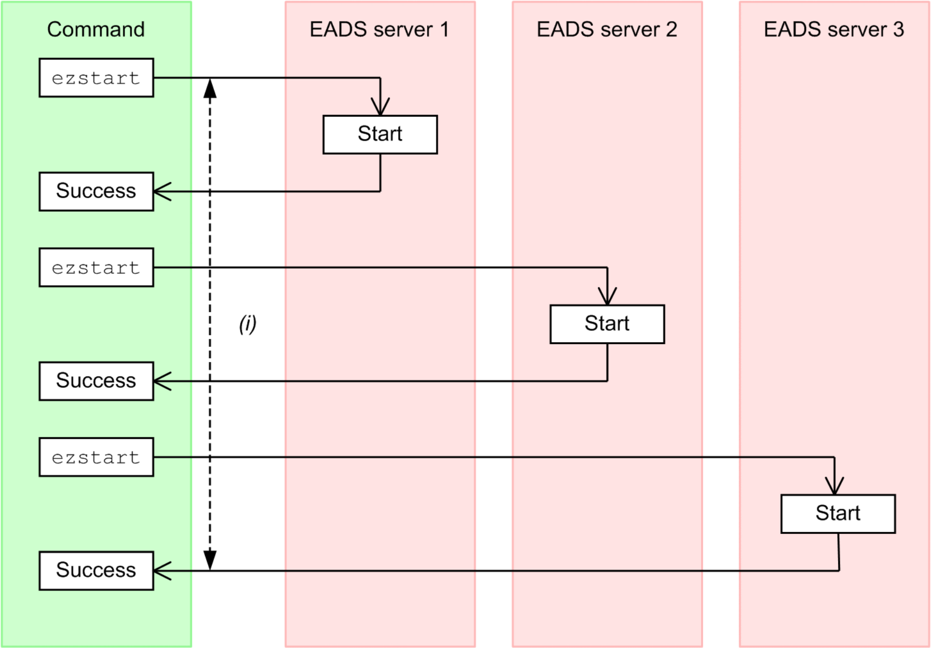
(d): *9.3.3(1)(d) eads.failureDetector.heartbeat.timeout*

(e): *9.3.3(1)(e) eads.failureDetector.connection.timeout*

(f): *9.3.3(1)(f) eads.failureDetector.read.timeout*

(2) Starting the cluster

The following figure shows the timers used when the EADS servers are started, based on an example of executing the `ezstart` command.



The alphabetical letters in the figure correspond to explanations provided in the following subsections of *9.3.3 Timeout-related parameters*:

(i): *9.3.3(1)(i) eads.admin.boot.timeout*

The EADS server in the cluster that has the smallest EADS server ID (as specified in the cluster properties) is started first. The EADS servers started sequentially thereafter receive heartbeats from the first EADS server and participate in the cluster.

The first EADS server that was started updates its cluster information based on the heartbeats received from the other EADS servers. This updated cluster information is then shared in the cluster.

(a) If the cluster properties differ from those of other EADS servers

An active EADS server sends a heartbeat with hashed cluster properties added.

The EADS server that receives the heartbeat checks the hash value. If the hash values do not match, the startup fails.

(b) If an EADS server already participating in the cluster is shut down while the cluster is starting

If an EADS server that is already participating in the cluster is shut down while the cluster is starting, that EADS server is isolated. If this happens, the other EADS servers stop their start processing and startup fails.

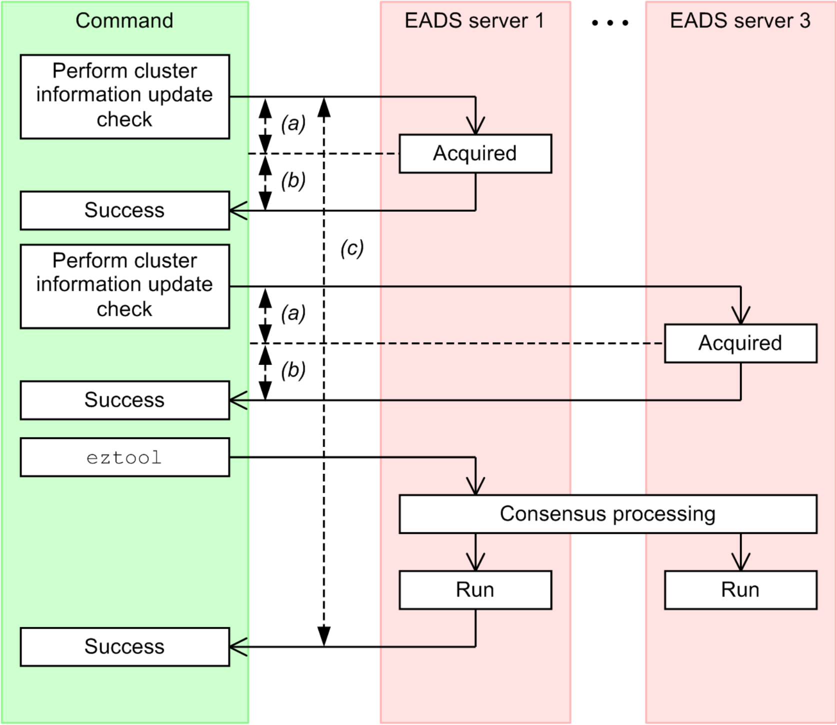
When at least half of the EADS servers in the cluster are shut down, a timeout occurs.

(c) If an EADS server that is not yet participating in the cluster is shut down while the cluster is starting

If an EADS server that is not yet participating in the cluster is shut down while the cluster is starting, any EADS server that has already started results in a timeout because the start processing of all EADS servers has not been completed.

(3) Running cluster operations

The following figure shows the timers used for running the cluster by executing the `eztool` command.



The alphabetical letters in the figure correspond to explanations provided in the following subsections of *9.3.3 Timeout-related parameters*:

(a): *9.3.3(2)(a) eads.command.connection.timeout*

(b): *9.3.3(2)(b) eads.command.common.read.timeout*#1

(c): *9.3.3(2)(c) eads.command.common.execution.timeout*#2

#1

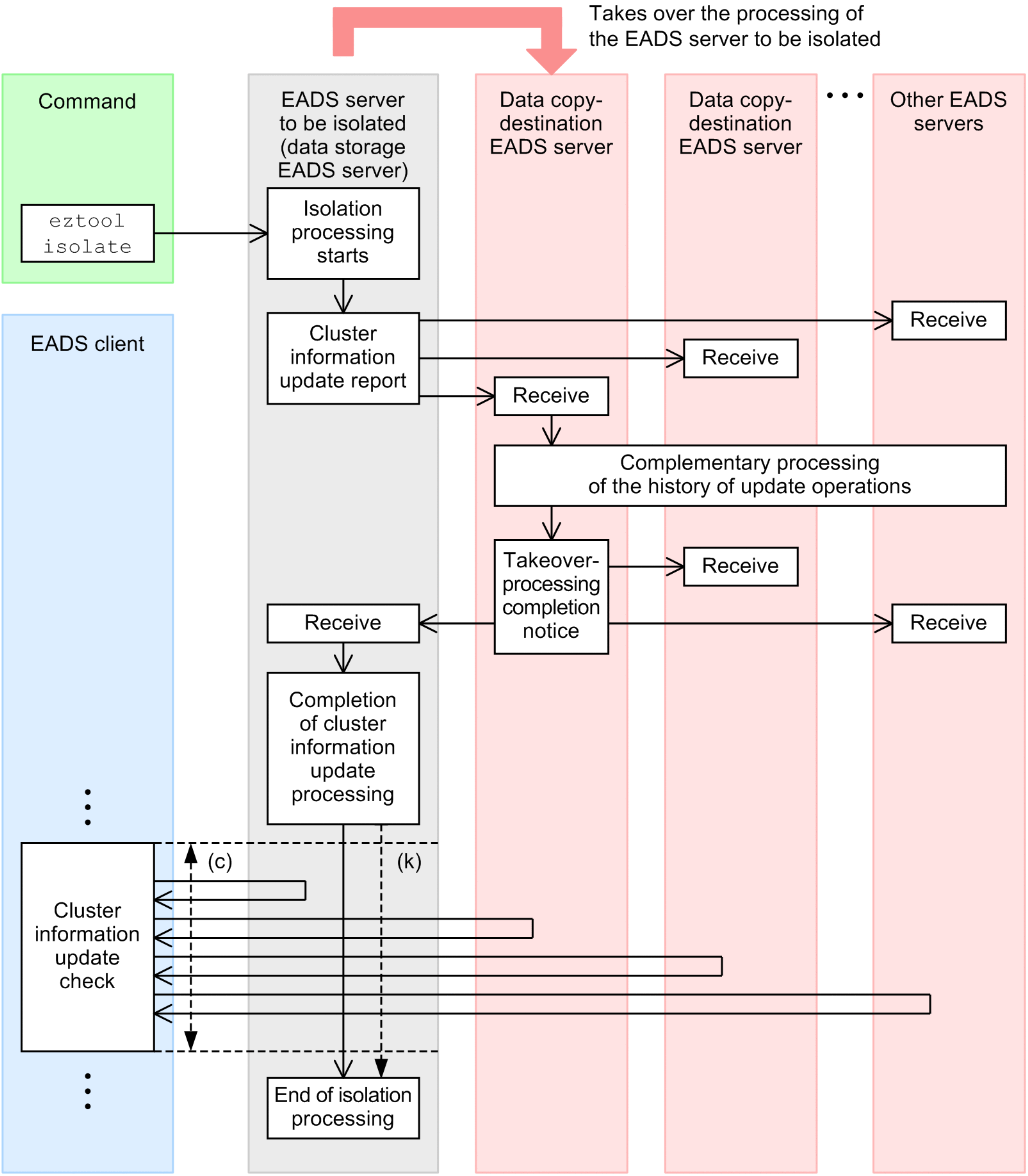
If the `eads.command.`*subcommand-name*`.read.timeout` parameter is specified in the command properties, its parameter value is used.

#2

If the `eads.command.`*subcommand-name*`.execution.timeout` parameter is specified in the command properties, its parameter value is used.

(4) EADS server isolation processing

The following figure shows the flow of EADS server isolation processing and the relation with timers:



Takes over the processing of the EADS server to be isolated
Command
eztool isolate
EADS server to be isolated (data storage EADS server)
Data copy-destination EADS server
Data copy-destination EADS server
Other EADS servers
Isolation processing starts
Cluster information update report
Receive
Receive
Receive
Complementary processing of the history of update operations
Takeover-processing completion notice
Receive
Receive
Receive
EADS client
Completion of cluster information update processing
Cluster information update check
(c)
(k)
End of isolation processing

The letters in the figure correspond to explanations provided in the following subsections of *9.3.3 Timeout-related parameters*:

(c): *9.3.3(3)(c) eads.client.clusterInfo.update.interval*

(k): *9.3.3(1)(k) eads.admin.operation.isolate.gracefulstop.waitTime*

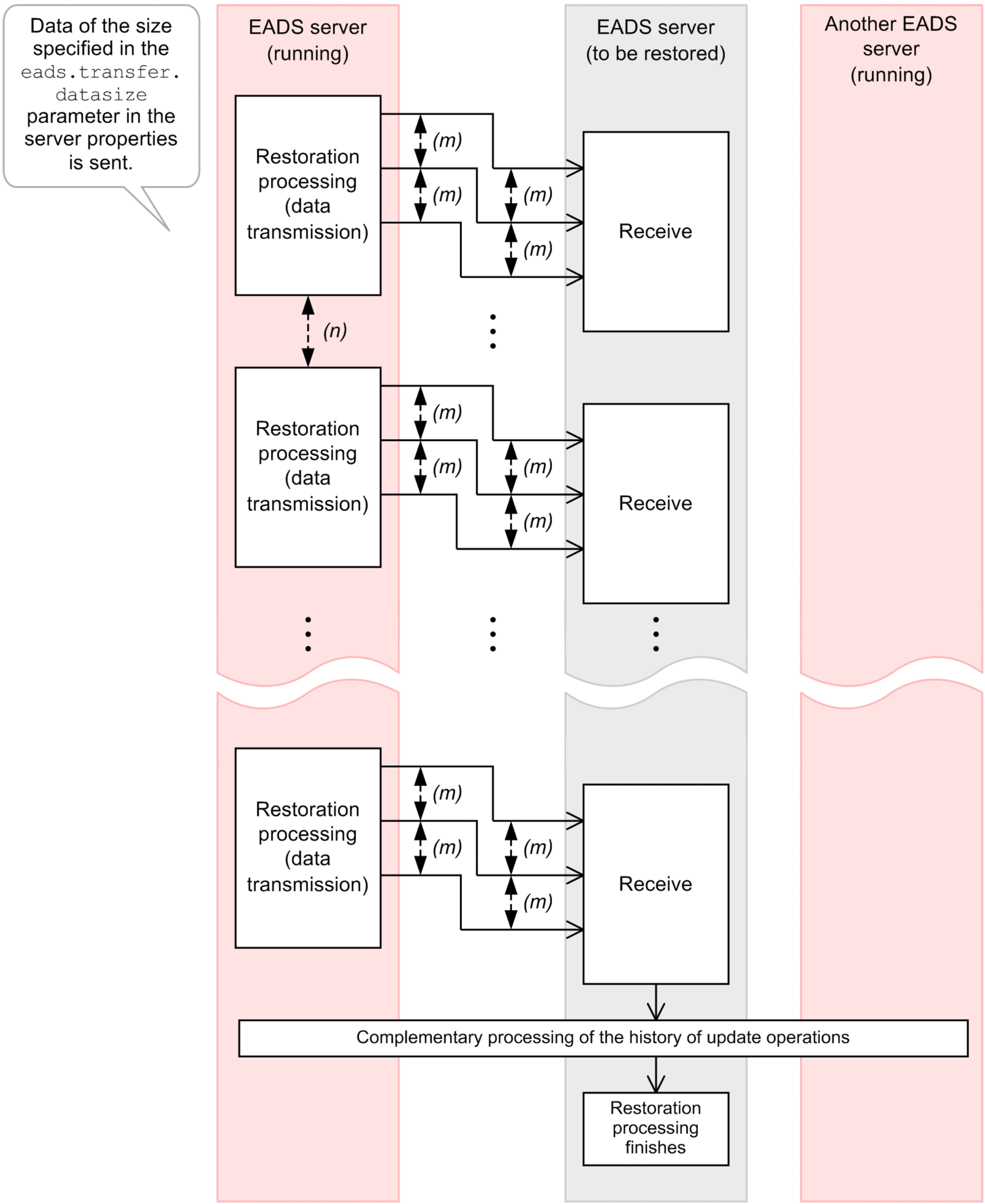
If you use the `eztool isolate` command to isolate an EADS server, you can specify in the `eads.admin.operation.isolate.gracefulstop.waitTime` parameter in the server properties the time allowed for completion of isolation processing since the cluster information update operation was completed. By specifying a value that is smaller than that value in the `eads.client.clusterInfo.update.interval` parameter in the client properties, you can isolate the EADS server after the cluster information update operation is completed on the EADS client.

Note that the specification of the `eads.admin.operation.isolate.gracefulstop.waitTime` parameter is invalid if the EADS server is isolated by cluster monitoring.

For details about the complementary processing of history of update operations, see *9.3.2(7) Complementary processing of the history of update operations*.

(5) Cluster recovery processing

The following figure shows the timers used when the cluster is recovered.


Data of the size specified in the eads.transfer.datasize parameter in the server properties is sent.
EADS server (running)
EADS server (to be restored)
Another EADS server (running)
Restoration processing (data transmission)
Receive
(m)
(n)
Complementary processing of the history of update operations
Restoration processing finishes

The alphabetical letters in the figure correspond to explanations provided in the following subsections of *9.3.3 Timeout-related parameters*:

(m): *9.3.3(1)(m) eads.transfer.timeout*

(n): *9.3.3(1)(n) eads.transfer.interval*

Data is sent in units of 10 kilobytes until the size specified in the `eads.transfer.datasize` parameter in the server properties for the EADS server subject to recovery is reached. For example, if a size of 25 kilobytes is specified, 30 kilobytes of data will be sent.

During restoration processing, the active EADS servers send data to the EADS server to be restored in order to recover data consistency.

Therefore, note the following:

- To restore an EADS server, it takes at least the time required for obtaining data.
- The EADS server that sends data is affected correspondingly by the amount of CPU resources and network bandwidth that are allocated for sending data.
- If the EADS server cannot keep up with the processing because both data operations and restoration processing must be performed, the EADS server might place data operations on hold to avoid a memory shortage.

Reference note

If you will be restoring disk caches and two-way caches, specify the size of the data to be transmitted during restoration processing in the `eads.cache.disk.transfer.datasize` parameter in the cache properties. Also specify the data transmission interval during restoration processing in the `eads.cache.disk.transfer.interval` parameter.

Even while the data is being updated, the isolated EADS server can be restored to the cluster with the data integrity recovered. For the general procedure for restoring one or more isolated EADS servers, see *12.2.1 If one or more EADS servers are isolated*.

For details about the complementary processing of history of update operations, see *9.3.2(7) Complementary processing of the history of update operations*.

(6) Cluster scale-out processing (adding EADS servers)

The timer used for scaling out the cluster (adding EADS servers to the cluster) is also used for recovering the cluster.

For details about the timer used for recovering the cluster, see *9.3.2(5) Cluster recovery processing* (replacing *recovery* with *scale-out*).

(7) Complementary processing of the history of update operations

During EADS server isolation processing, restoration processing, scale-out processing, and locking, the EADS servers check their histories of update operations with each other. If any difference is detected, complementary processing of the history of update operations is performed. This ensures the consistency of the order in which data is written.

Complementary processing of the history of update operations consists of the following two processes:

- Complementary processing of the history of update operations on the remote EADS server
- Complementary processing of the history of update operations on the local EADS server

Here, the *local EADS server* means the following EADS server.

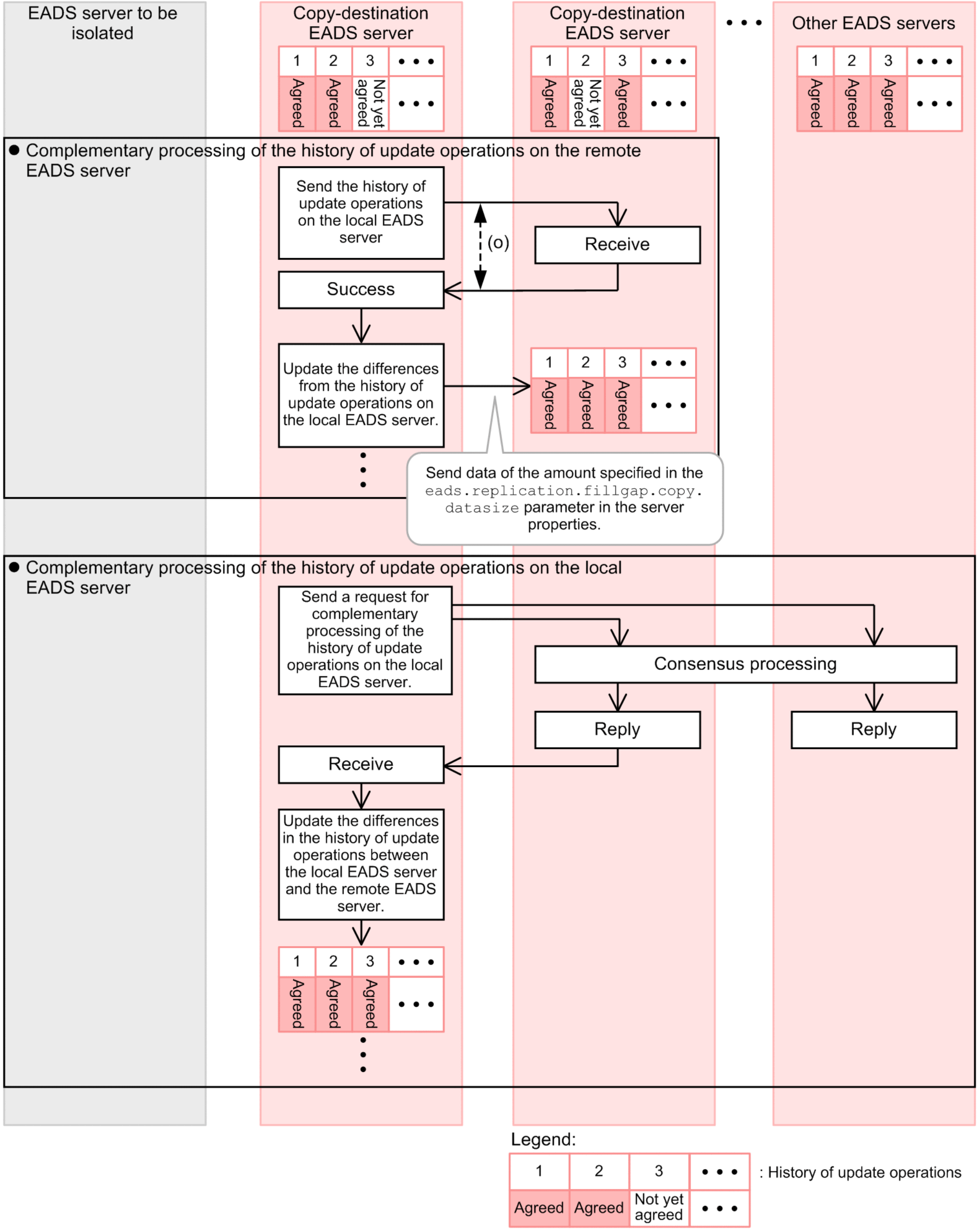
For isolation processing:

The EADS server (copy-destination EADS server) that takes over the processing of the EADS server to be isolated

For restoration and scale-out processing:

The EADS server to be restored and the EADS server to be added during scale-out processing

The following figure shows the flow of complementary processing of the history of update operations and the relation with timers:


EADS server to be isolated
Copy-destination EADS server
Copy-destination EADS server
Other EADS servers
1 2 3 • • •
Agreed Agreed Not yet agreed • • •
Agreed Not yet agreed Agreed • • •
Agreed Agreed Agreed • • •
● Complementary processing of the history of update operations on the remote EADS server
Send the history of update operations on the local EADS server
Receive
(o)
Success
Update the differences from the history of update operations on the local EADS server.
Send data of the amount specified in the eads.replication.fillgap.copy.datasize parameter in the server properties.
● Complementary processing of the history of update operations on the local EADS server
Send a request for complementary processing of the history of update operations on the local EADS server.
Consensus processing
Reply
Reply
Receive
Update the differences in the history of update operations between the local EADS server and the remote EADS server.
Legend:
: History of update operations

The alphabetical letters in the figure correspond to explanations provided in the following subsections of *9.3.3 Timeout-related parameters*:

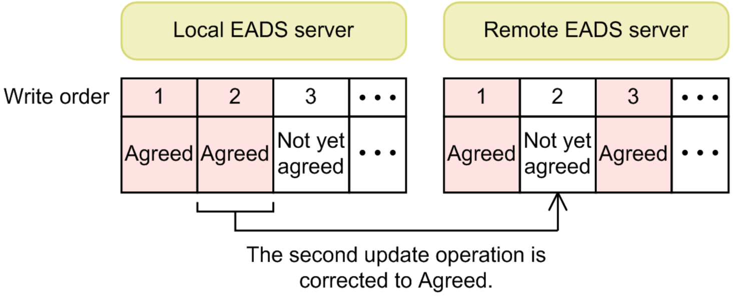
(o): *9.3.3(1)(o) eads.replication.fillgap.copy.timeout*

Complementary processing of the history of update operations on the remote EADS server

1. The history of update operations on the local EADS server is sent to each EADS server.

2. If the history of update operations on the local EADS server is different from that on a remote EADS server, the local EADS server sends the history of update operations to the remote EADS server. At this time, the EADS server sends the amount of data specified in the `eads.replication.fillgap.copy.datasize` parameter in the server properties.
3. The history of update operations on the remote EADS server is complemented based on the history of update operations sent from the local EADS server.

   The history of update operations on the remote EADS server is complemented as shown below.



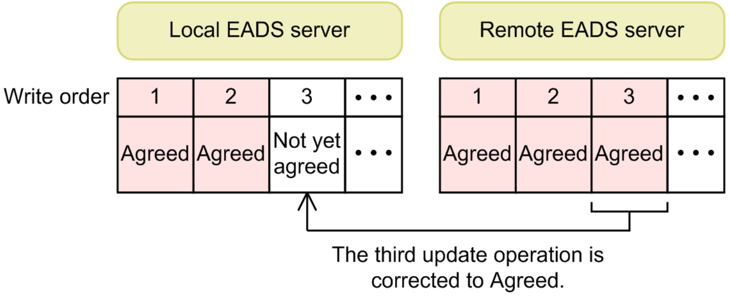
   Complementary processing of the history of update operations on the remote EADS server is performed for the number of differences in the history of update operations for the copy-destination EADS servers.

Complementary processing of the history of update operations on the local EADS server

1. A request for complementary processing of the history of update operations is sent to each EADS server to check whether the history of update operations on the local EADS server is different from other servers.
2. Consensus processing of the history of update operations is performed in response to the request.

   If the consensus processing does not finish within the time specified by the `eads.replication.consensus.timeout` parameter in the server properties, a timeout occurs, and then the consensus processing is performed again. This process is repeated until a consensus is built.
3. Via consensus processing, the remote EADS server sends the history of update operations to the local EADS server.
4. The history of update operations on the local EADS server is complemented based on the history of update operations sent by each EADS server.

   The history of update operations on the local EADS server is complemented as shown in the following diagram:



   Complementary processing of the history of update operations on the local EADS server is performed for the number of differences in the history of update operations on the local EADS server.

Complementary processing of the history of update operations might be performed more than once for one operation of restoration, isolation, or scale-out processing. The maximum number of times complementary processing can be performed is as follows: (*data multiplicity* - 1) × (*number of caches*).

The number of simultaneous threads for complementary processing of the history of update operations is the *number of redundant copies of data plus the original* - 1 (the value is 1 if the number of redundant copies of data plus the original is 1).

9.3.3 Timeout-related parameters

The following table lists the parameters related to timeouts.

Table 9–4: Timeout-related parameters

No.	Property file	Parameter name	Value to be specified	Default value
1	Server property file	eads.server.connection.timeout	100 to 3600000 (milliseconds)	60000
2		eads.admin.operation.connection.timeout	100 to 3600000 (milliseconds)	10000
3		eads.failureDetector.heartbeat.interval	10 to 60000 (milliseconds)	400
4		eads.failureDetector.heartbeat.timeout	10 to 86400000 (milliseconds)	2000
5		eads.failureDetector.connection.timeout	1 to 60000 (milliseconds)	500
6		eads.failureDetector.read.timeout	1 to 60000 (milliseconds)	500
7		eads.failureDetector.retry	0 to 100	0
8		eads.failureDetector.assertive.threshold	1 to 49	1
9		eads.admin.boot.timeout	1 to 86400 (seconds)	60
10		eads.server.connection.keepAlive.timeout	0 to 3600 (seconds)	3600
11		eads.admin.operation.isolate.gracefulstop.waitTime	0 to 60000 (milliseconds)	3000
12		eads.replication.consensus.timeout	10 to 3600000 (milliseconds)	800
13		eads.transfer.timeout	100 to 3600000 (milliseconds)	60000
14		eads.transfer.interval	0 to 60000 (milliseconds)	1000
15		eads.replication.fillgap.copy.timeout	100 to 3600000 (milliseconds)	2000
16		eads.admin.operation.resume.send.interval	0 to 2147483647 (milliseconds)	0
17	Command property file	eads.command.connection.timeout	0 to 2147483647 (milliseconds)	3000
18		eads.command.common.read.timeout	0 to 2147483647 (milliseconds)	60000
19		eads.command.common.execution.timeout	0 to 2147483647 (seconds)	600

No.	Property file	Parameter name	Value to be specified	Default value
20		eads.command.*subcommand-name*.read.timeout	0 to 2147483647 (milliseconds)	600000
21		eads.command.*subcommand-name*.execution.timeout	0 to 2147483647 (seconds)	600
22	Client property file	eads.client.connection.send.timeout	100 to 3600000 (milliseconds)	60000
23		eads.client.connection.receive.timeout	100 to 3600000 (milliseconds)	60000
24		eads.client.clusterInfo.update.interval	10 to 60000 (milliseconds)	1000

(1) Server property file

(a) eads.server.connection.timeout

This parameter specifies a timeout value (in milliseconds) for cluster information update checks and data transfer processing.

(b) eads.admin.operation.connection.timeout

This parameter specifies a timeout value (in milliseconds) for connection establishment and data transfer between EADS servers during execution of the following commands:

- eztool import
- eztool resume
- eztool importecf

(c) eads.failureDetector.heartbeat.interval

This parameter specifies a heartbeat transmission interval (in milliseconds).

As this parameter's value becomes greater, adverse effects on the restoration and scale-out processing increase and timeouts occur more frequently during cluster startup.

As this parameter's value decreases, the communication workload for processing increases.

Specify for this parameter a value that is smaller than the following parameter values:

- eads.admin.boot.timeout parameter in the server properties
- eads.transfer.timeout parameter in the server properties
- eads.command.common.read.timeout parameter in the command properties
- eads.command.common.execution.timeout parameter in the command properties
- eads.command.*subcommand-name*.read.timeout parameter in the command properties
- eads.command.*subcommand-name*.execution.timeout parameter in the command properties

(d) eads.failureDetector.heartbeat.timeout

This parameter specifies a heartbeat timeout value (in milliseconds).

As this parameter's value becomes greater, the time required for detecting failures increases. On the other hand, if this parameter value is too small, failures might be detected erroneously.

As this parameter's value becomes greater, the sizes of the Java heap (`eads.java.heapsize` parameter value in the shared properties) and explicit heap (`eads.java.external.heapsize` parameter value in the shared properties) required for managing and storing the history of update operations become larger.

To perform consensus processing without isolating EADS servers in the event of a temporary failure, specify in this parameter a value that is greater than the value of the `eads.replication.consensus.timeout` parameter in the server properties. If this parameter's value is smaller than the value of the `eads.replication.consensus.timeout` parameter in the server properties, a timeout might occur during isolation processing, resulting in extra time being required to complete the processing.

(e) eads.failureDetector.connection.timeout

This parameter specifies a connection timeout value (in milliseconds) for the check for live servers.

As this parameter's value becomes greater, the time required for detecting failures becomes longer when a connection cannot be established. On the other hand, if this parameter's value is too small, a timeout might occur before the connection is established successfully.

We recommend that you specify for this parameter a value that is close to the value set in the `eads.replication.consensus.timeout` parameter in the server properties.

(f) eads.failureDetector.read.timeout

This parameter specifies a reception timeout value (in milliseconds) for the check for live servers.

As this parameter's value becomes greater, the time required for detecting failures becomes longer. On the other hand, if this parameter's value is too small, failures might be detected erroneously.

We recommend that you specify for this parameter a value that is close to the value set in the `eads.replication.consensus.timeout` parameter in the server properties.

(g) eads.failureDetector.retry

This parameter specifies the number of retries if a check for live servers times out.

(h) eads.failureDetector.assertive.threshold

This parameter specifies the number of EADS servers that must agree before an EADS server can be shut down.

Specify a value less than or equal to the value that is obtained by rounding up (*number of EADS servers constituting the cluster* + 1) ÷ 2. For example, if a cluster consists of five EADS servers, specify `3` or a smaller value in this parameter.

If you specify a large value in this parameter, it might become impossible to isolate an EADS server that has been shut down. On the other hand, in the event of a temporary network failure (such as split-brain), the EADS server can be protected from being isolated prematurely.

(i) eads.admin.boot.timeout

This parameter specifies the maximum wait time (in seconds) until all EADS servers making up the cluster start up.

(j) eads.server.connection.keepAlive.timeout

This parameter specifies the length of time (in seconds) before the connection between the EADS client and EADS servers is released.

If request wait status (in which no communication takes place) lasts for the specified length of time or more, the connection is released.

If zero is specified, no timeout occurs (there is no limit).

(k) eads.admin.operation.isolate.gracefulstop.waitTime

This parameter specifies the time (in milliseconds) allowed for completion of isolation processing since the cluster information update operation was completed on the EADS server on which the `eztool isolate` command is executed.

When the EADS server that received an update confirmation of cluster information from the EADS client is isolated, if the connection is closed before the update of the cluster information on the EADS client is complete, a communication error occurs. By adjusting the value of this parameter, you can isolate the EADS server after updating of the cluster information on the EADS client has finished.

(l) eads.replication.consensus.timeout

This parameter specifies a timeout value (in milliseconds) for consensus processing.

If the consensus processing does not finish within the specified time, a timeout occurs, and then the consensus processing is performed again.

(m) eads.transfer.timeout

This parameter specifies a data transmission timeout value (in milliseconds) during restoration processing and scale-out processing.

Specify a value that is appropriate to the size of the data to be handled.

Specify this parameter to protect against the following events:

- If an EADS server that transfers data shuts down during restoration processing or scale-out processing, the EADS server subject to restoration or the EADS server that was added during scale-out processing is placed in wait status.
- If an EADS server subject to restoration or an EADS server that was added during scale-out processing hangs without releasing the connection, the EADS server that transfers data during restoration processing or scale-out processing is placed in wait status.

(n) eads.transfer.interval

This parameter specifies a data transmission interval (in milliseconds) during restoration processing and scale-out processing.

The value specified in this parameter is applied during restoration processing or scale-out processing to the EADS server subject to restoration or the EADS server that was added during scale-out processing.

Specify this parameter's value and the value of the `eads.transfer.datasize` parameter in the server properties in such a manner that the following condition is satisfied:

> *Bandwidth available for data transmission during restoration processing and scale-out processing* (bps) ≥
> (*size of data transmitted during restoration processing and scale-out processing* (bytes) × 8)
> ÷ {(*data transmission interval for restoration processing and scale-out processing* (milliseconds)
> + *time required for data transmission* (milliseconds)) ÷ 1,000}

Size of data transmitted during restoration processing and scale-out processing (bytes):

`eads.transfer.datasize` parameter value in the server properties

Data transmission interval for restoration processing and scale-out processing (milliseconds):

`eads.transfer.interval` parameter value in the server properties

Time required for data transmission (milliseconds):

Time required for transmitting the amount of data specified in the `eads.transfer.datasize` server property parameter (milliseconds)

This is the time required for *Restoration processing (data transmission)* in the figure in *9.3.2(5) Cluster recovery processing* (for scale-out processing, replace *recovery processing* with *scale-out processing*). This time value depends on the environment.

As the time required for restoration processing or scale-out processing becomes shorter, the communication workload for restoration processing increases. If the communication workload becomes large, the processing speed of an application program that updates caches might decrease. Conversely, as the communication workload for restoration processing or scale-out processing decreases, the time required for restoration processing or scale-out processing increases.

When disk caches and two-way caches are restored, this parameter is ignored and the value of the `eads.cache.disk.transfer.interval` cache property parameter is used.

(o) eads.replication.fillgap.copy.timeout

This parameter specifies the timeout period (in milliseconds) for transmission of the history of update operations during complementary processing.

Specify the greater of the following values as the time available for processing:

- The amount of data that is sent at one time to a copy destination EADS server (the value of the `eads.replication.fillgap.copy.datasize` parameter in the server properties)
- *maximum key size* + *maximum value size*# × MAX(2, *maximum number of data items that can be updated simultaneously*)

  #

  Maximum size that can be specified when `put`, `create`, `update`, or `replace` processing is performed.

  MAX:

  Choose the largest value in the parentheses that follow MAX.

  Example: For MAX(2, 10), the calculation result is 10.

  Maximum number of data items that can be updated simultaneously:

  If you will be performing batch data operations using a memory cache, specify `10`. Otherwise, specify `1`.

For details about the complementary processing of history of update operations, see *9.3.2(7) Complementary processing of the history of update operations*.

(p) eads.admin.operation.resume.send.interval

This parameter specifies a differential data transfer interval (in milliseconds) during execution of the `eztool resume` command.

(2) Command property file

(a) eads.command.connection.timeout

This parameter specifies a connection timeout value (in milliseconds) for communication with the EADS server during command execution.

If zero is specified in this parameter, no timeout occurs.

Note that the following commands ignore this parameter:

- `eztool threaddump`
- `eztool snapshot`
- `eztool forcestop`

(b) eads.command.common.read.timeout

This parameter specifies a reception timeout value (in milliseconds) for communication with the EADS server during command execution.

Specify in this parameter a value that is greater than the value of the `eads.admin.operation.isolate.gracefulstop.waitTime` parameter in the server properties.

If zero is specified in this parameter, no timeout occurs.

Note that the following commands ignore this parameter:

- `eztool threaddump`
- `eztool snapshot`
- `eztool forcestop`

When the following commands are executed, this parameter's value is applied in place of the value for the `eads.client.connection.receive.timeout` parameter in the client properties:

- `eztool put`
- `eztool get`
- `eztool remove`
- `eztool listgroup`
- `eztool listkey`
- `eztool removeall`
- `eztool execfunc`

This parameter's value is not applied in the following cases:

- Zero is specified in this parameter or this parameter's value is equal to or greater than the maximum value for the `eads.client.connection.receive.timeout` parameter in the client properties

  In such a case, the maximum value for the `eads.client.connection.receive.timeout` parameter is applied.
- This parameter's value is equal to or smaller than the minimum value set for the `eads.client.connection.receive.timeout` parameter in the client properties

  In such a case, the minimum value for the `eads.client.connection.receive.timeout` parameter is applied.

(c) eads.command.common.execution.timeout

This parameter specifies a timeout value (in seconds) from the start of command execution.

If zero is specified in this parameter, no timeout occurs.

Note that the following commands ignore this parameter:

- `eztool threaddump`
- `eztool snapshot`
- `eztool forcestop`

(d) eads.command.subcommand-name.read.timeout

This parameter specifies a reception timeout value (in milliseconds) for communication with the EADS server while a specified subcommand is executing.

For *subcommand-name*, specify the name of the subcommand to which you want to apply this parameter's value. When this subcommand executes, its value takes precedence over the value set for the `eads.command.common.read.timeout` parameter in the command properties.

Specify in this parameter a value that is greater than the value set for the `eads.admin.operation.isolate.gracefulstop.waitTime` parameter in the server properties.

If zero is specified in this parameter, no timeout occurs.

Note that the following commands ignore this parameter:

- `eztool threaddump`
- `eztool snapshot`
- `eztool forcestop`

When the following commands are executed, this parameter's value is applied in place of the value set for the `eads.client.connection.receive.timeout` parameter in the client properties:

- `eztool put`
- `eztool get`
- `eztool remove`
- `eztool listgroup`
- `eztool listkey`
- `eztool removeall`

- eztool execfunc

This parameter's value is not applied in the following cases:

- Zero is specified in this parameter or this parameter's value is equal to or greater than the maximum value for the eads.client.connection.receive.timeout parameter in the client properties

  In such a case, the maximum value for the eads.client.connection.receive.timeout parameter is applied.
- This parameter's value is equal to or smaller than the minimum value set for the eads.client.connection.receive.timeout parameter in the client properties

  In such a case, the minimum value for the eads.client.connection.receive.timeout parameter is applied.

(e) eads.command.subcommand-name.execution.timeout

This parameter specifies a timeout value (in seconds) since a specified subcommand started executing.

For *subcommand-name*, specify the name of the subcommand to which you want to apply this parameter's value. When this subcommand executes, its value takes precedence over the value set for the eads.command.common.execution.timeout parameter in the command properties.

If zero is specified in this parameter, no timeout occurs.

Note that the following commands ignore this parameter:

- eztool threaddump
- eztool snapshot
- eztool forcestop

(3) Client property file

(a) eads.client.connection.send.timeout

This parameter specifies a timeout value (in milliseconds) for cluster information update checks and data transmission.

(b) eads.client.connection.receive.timeout

This parameter specifies a timeout value (in milliseconds) for cluster information update checks and data reception.

Consider the following times:

- Execution times for the user functions allocated on the EADS server

(c) eads.client.clusterInfo.update.interval

This parameter specifies an interval (in milliseconds) at which the EADS client is to perform cluster information update checks on each EADS server in the cluster.

Specify in this parameter a value that is smaller than the value set for the eads.admin.operation.isolate.gracefulstop.waitTime parameter in the server properties.

When you specify this parameter, take into account the number of EADS servers that constitute the cluster.

As the number of EADS servers that constitute the cluster increases, the number of times the EADS client must communicate within the time specified in this parameter increases, resulting in an increase in the EADS client's workload.

9.4 Designing the command operation-related parameters

Design the command operation-related parameters.

9.4.1 Command operation-related parameters

The following table shows the command operation-related parameter.

Table 9–5: Command operation-related parameter

No.	Property file	Parameter name	Value to be specified	Default value
1	Command property file	`eads.command.compat`	`0300`	None

(1) Command property file

(a) eads.command.compat

This parameter specifies the version with which the commands are compatible.

If `0300` is specified, command display results and return codes are the same as for the commands of version 03-60 or earlier.

If any other value is specified or this parameter is omitted, the command display results and return codes depend on the version of the commands being used.

9.5 Designing application program operation-related parameters

Design the application program operation-related parameters.

9.5.1 Application program operation-related parameters

The following table lists the application program operation-related parameters.

Table 9–6: Application program operation-related parameters

No.	Property file	Parameter name	Value to be specified	Default value
1	Client property file	eads.client.batchOperation.unit	1 to 1024	10
2		eads.client.compat	0300	None

(1) Client property file

(a) eads.client.batchOperation.unit

This parameter specifies the number of data operations to be performed in response to a single communication event with the EADS server when a batch operation is performed on cache data. By specifying this parameter, you can adjust the usage amount and rate for each resource during the batch operation.

The following are the effects of increasing this parameter's value:

- Improvement of performance can be expected because the number of times communication is established with the EADS server is reduced.
- Performance might be affected adversely by locked resources, such as the network and memory, when a large amount of data is transmitted.

Important note

An appropriate value depends on the environment. Test the operation and then specify an appropriate value.

This parameter's value is used in the following API functions and methods:

- Methods available in Java client libraries
  - putAll() of the Cache class
  - getAll() (with Set specified) of the Cache class
  - getAll() (group specification) of the Cache class
- Functions available in C client libraries
  - ead_put_all()
  - ead_get_all()
  - ead_get_group()

The API functions and methods listed below do not include values for communication. Therefore, regardless of this parameter's value, these API functions and methods perform a maximum of 1,024 data operations in response to a single communication event with the EADS server.

- Method available in Java client libraries
  `removeAll()` (with `Set` specified) in the `Cache` class
- Function available in C client libraries
  `ead_remove_all()`

The API functions and methods listed below perform all target operations in response to a single communication event with the EADS server regardless of this parameter's value:

- Methods available in Java client libraries
  - `removeAll()` (group specification) of the `Cache` class
  - `removeAll()` (EADS server specification) of the `Cache` class
- Functions available in C client libraries
  - `ead_remove_group()`
  - `ead_remove_node()`

(b) eads.client.compat

This parameter specifies the version with which the EADS client is compatible.

If `0300` is specified, the same error codes as for EADS client version 03-60 or earlier are returned.

If any other value is specified or this parameter is omitted, the error codes for the EADS client version being used are returned.

9.6 Designing the compaction-related parameters

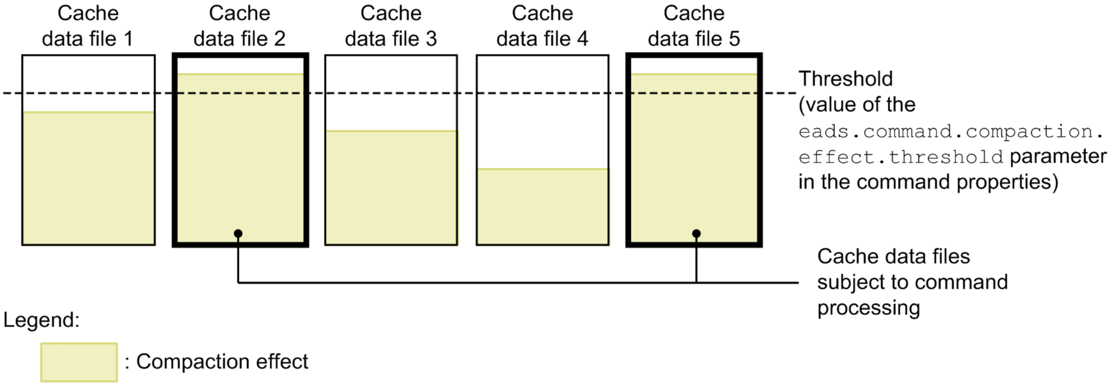
This section is applicable when you will be using disk caches or two-way caches.

Design the parameters related to compaction of cache data. For details about compaction, see *10.9 Reducing the data usage of cache data files (performing compaction on cache data files)*.

9.6.1 Specifying thresholds for compaction effects

You can specify thresholds for compaction effects on cache data files. The following figure provides an overview of thresholds for compaction effects.

Figure 9–3: Overview of thresholds for compaction effects


Cache data file 1
Cache data file 2
Cache data file 3
Cache data file 4
Cache data file 5
Threshold (value of the eads.command.compaction.effect.threshold parameter in the command properties)
Cache data files subject to command processing
Legend:
: Compaction effect

Approach

The compaction effects depend on the cache data file. By specifying a threshold when you execute the `eztool compaction` command, you can perform compaction on certain cache data files, for example, that will yield desired compaction effects.

You can use the `eads.command.compaction.effect.threshold` parameter in the command properties to specify a threshold of compaction effects that is to be set when the `--threshold` option is omitted from the following commands:

- `eztool listecf` command
- `eztool compaction` command

In this figure, cache data files 2 and 5 yield at least the threshold of compaction effects. If the `eztool compaction` command is executed with the `--threshold` option omitted, cache data files 2 and 5 become subject to compaction processing.

9.6.2 Compaction-related parameters

The following table lists the compaction-related parameters.

Table 9–7: Compaction-related parameters

No.	Property file	Parameter name	Value to be specified	Default value
1	Command property file	`eads.command.compaction.effect.threshold`	`1` to `100` (%)	`50`
2		`eads.command.compaction.effect.division`	`1` to `10`	`5`

(1) Command property file

(a) eads.command.compaction.effect.threshold

This parameter specifies a threshold (%) of compaction effects to be used in the following commands:

- `eztool listecf`
- `eztool compaction`

(b) eads.command.compaction.effect.division

This parameter specifies the number of distribution ranges to be used when indicating by means of the `eztool listecf` command the distribution of the numbers of files in each compaction effects range.

For example, if `4` is specified in this parameter, the distribution is divided into four ranges, and the numbers of files in the compaction effects ranges 0% to 25%, 26% to 50%, 51% to 75%, and 76% to 100% are displayed.

If the value indicating effects is not an integer, all digits following the decimal point are discarded.

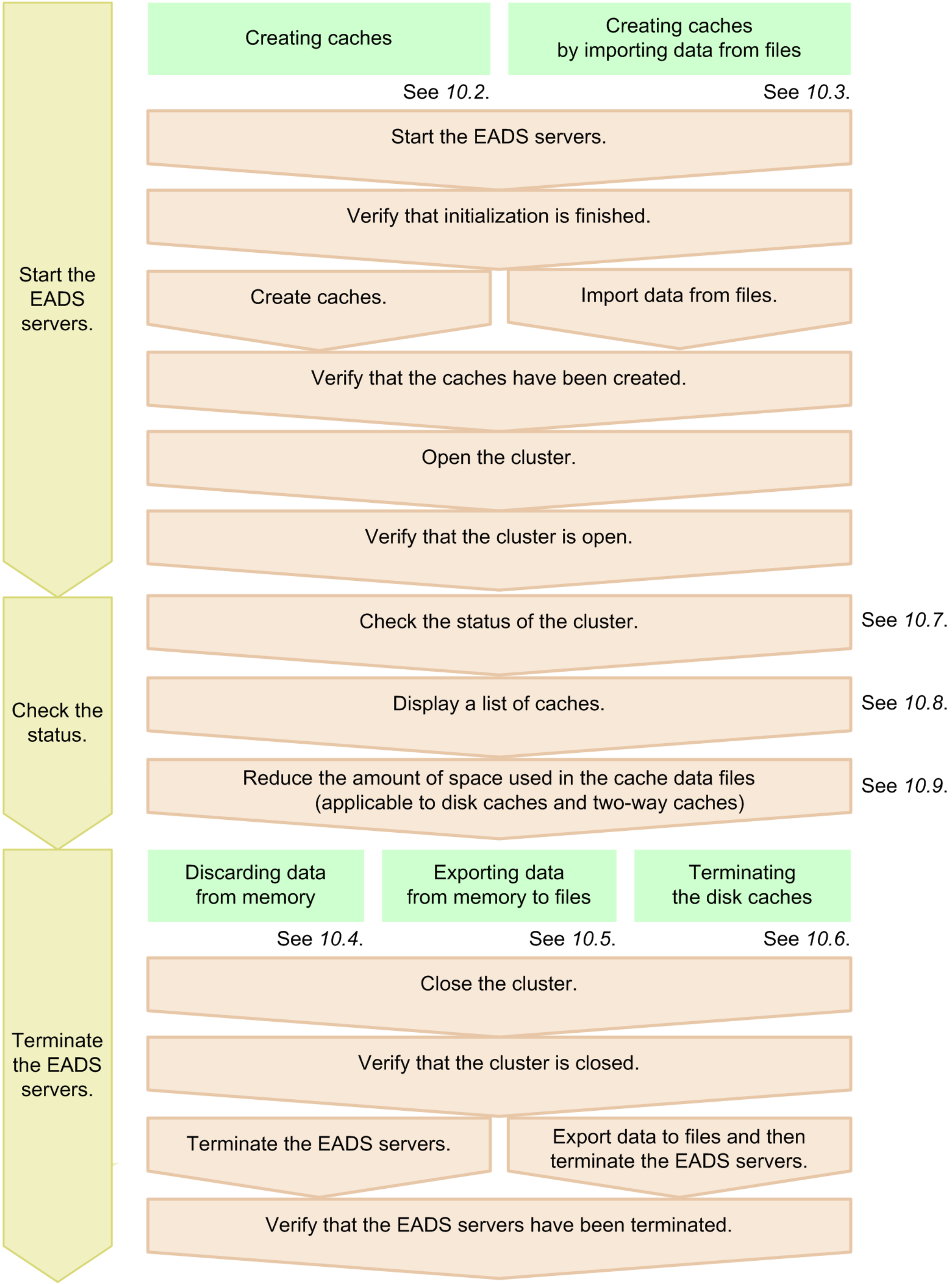
10 Normal Operations

This chapter explains the system operation administrator's main tasks and how to perform normal operations in EADS.

10.1 The system operation administrator's tasks

The following figure shows the general procedure for normal operations.

Figure 10–1: General procedure for normal operations


Start the EADS servers.
Creating caches
See 10.2.
Creating caches by importing data from files
See 10.3.
Start the EADS servers.
Verify that initialization is finished.
Create caches.
Import data from files.
Verify that the caches have been created.
Open the cluster.
Verify that the cluster is open.
Check the status.
Check the status of the cluster.
See 10.7.
Display a list of caches.
See 10.8.
Reduce the amount of space used in the cache data files (applicable to disk caches and two-way caches)
See 10.9.
Terminate the EADS servers.
Discarding data from memory
See 10.4.
Exporting data from memory to files
See 10.5.
Terminating the disk caches
See 10.6.
Close the cluster.
Verify that the cluster is closed.
Terminate the EADS servers.
Export data to files and then terminate the EADS servers.
Verify that the EADS servers have been terminated.

If system maintenance is needed, perform maintenance operations. For details about the maintenance operations, see *11. Maintenance Operations*.

In the event of a failure in the system, perform error handling operations. For details about the error handling operations, see *12. Error Handling Operations*.

10.2 Starting the EADS servers (and creating caches)

This section explains how to start EADS servers and then create caches.

10.2.1 How to start the EADS servers (creating caches in memory)

This subsection explains how to start the EADS servers and then create memory caches.

(1) Start the EADS servers

Log in to the host on which you plan to start an EADS server, and then execute the `ezstart` command to start the EADS server. You must execute this command for each EADS server.

```
ezstart
```

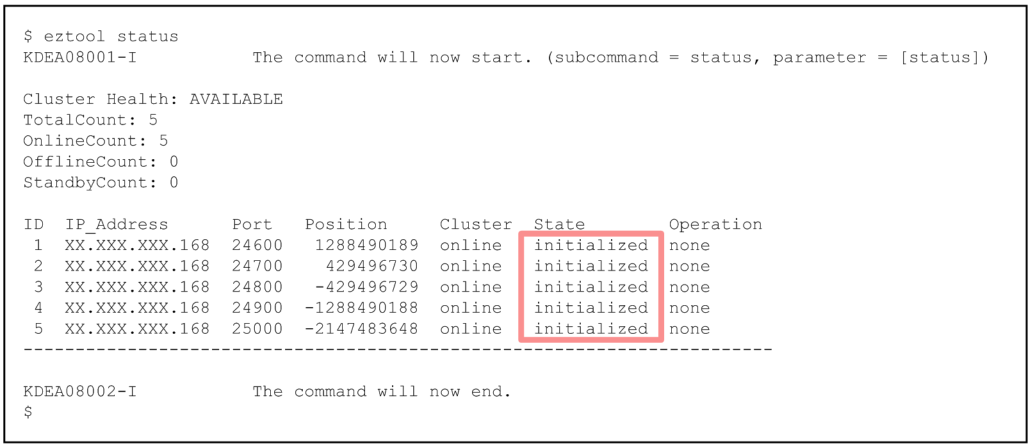
(2) Verify that initialization is finished

After an EADS server starts up successfully, it changes to initialized status.

Execute the `eztool status` command to verify that the initialization of all EADS servers is finished.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address       Port   Position     Cluster  State        Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   initialized  none
 2  XX.XXX.XXX.168  24700    429496730  online   initialized  none
 3  XX.XXX.XXX.168  24800   -429496729  online   initialized  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   initialized  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   initialized  none
------------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

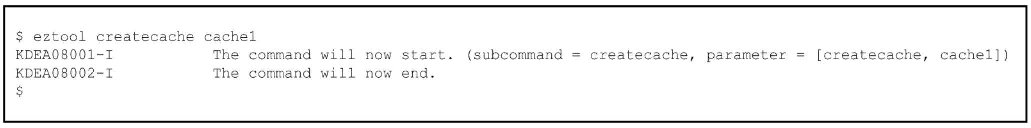
If the initialization is finished, `initialized` is displayed in the `State` column.

(3) Create caches

Execute the `eztool createcache` command to create a cache for storing keys and values.

```
eztool createcache cache-name
```

Command execution example



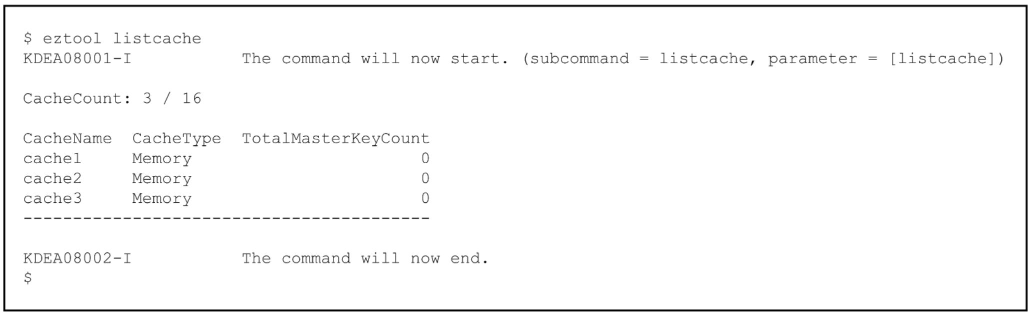
```
$ eztool createcache cache1
KDEA08001-I           The command will now start. (subcommand = createcache, parameter = [createcache, cache1])
KDEA08002-I           The command will now end.
$
```

(4) Verify that caches have been created

Execute the `eztool listcache` command to display a list of caches.

```
eztool listcache
```

Command execution example



```
$ eztool listcache
KDEA08001-I           The command will now start. (subcommand = listcache, parameter = [listcache])

CacheCount: 3 / 16

CacheName  CacheType  TotalMasterKeyCount
cache1     Memory                       0
cache2     Memory                       0
cache3     Memory                       0
-----------------------------------------

KDEA08002-I           The command will now end.
$
```

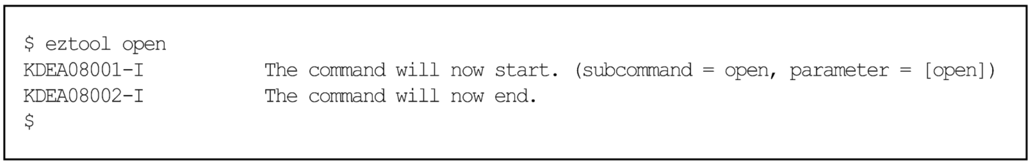
Check the cache names in the `CacheName` column and the cache types in the `CacheType` column to verify that the caches have been created as intended.

(5) Open the cluster

As is the case with closed status, in initialized status, the EADS servers do not accept requests from the EADS client. Execute the `eztool open` command to open the cluster.

```
eztool open
```

Command execution example



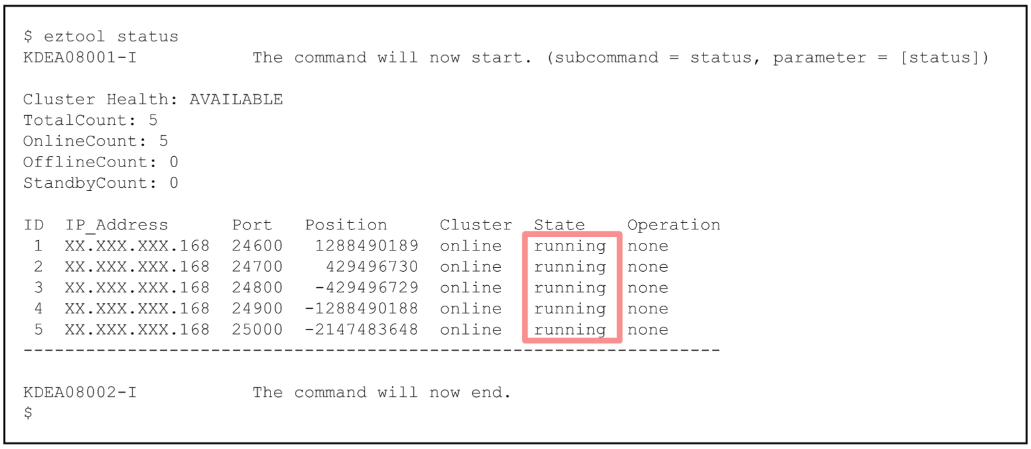
```
$ eztool open
KDEA08001-I           The command will now start. (subcommand = open, parameter = [open])
KDEA08002-I           The command will now end.
$
```

(6) Verify that the EADS servers have been opened

Execute the `eztool status` command to verify that all EADS servers have been opened.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      Port   Position     Cluster  State    Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   running  none
 2  XX.XXX.XXX.168  24700    429496730  online   running  none
 3  XX.XXX.XXX.168  24800   -429496729  online   running  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   running  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   running  none
--------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

If the EADS servers have been opened, `running` is displayed in the `State` column.

10.2.2 Starting the EADS servers (creating caches on disk)

This subsection is applicable when you will be using disk caches or two-way caches.

This subsection explains how to start the EADS servers and then create disk caches or two-way caches.

(1) Create cache property files

Create cache property files for specifying cache information, such as the cache types and cache storage locations.

The file name of a cache property file is `eads_cache.`*`cache-name`*`.properties`.

For details about the cache property parameters, see *7.7 Designing the cache operation-dependent parameters*.

Important note

The settings for the parameters shown below must be identical in the cache property files for all the EADS servers that make up the cluster. If any settings are different, caches cannot be created.

- `eads.cache.type`
- `eads.cache.disk.filesize`
- `eads.cache.disk.filenum`
- `eads.cache.disk.blocksize`

The following shows an example of a cache property file for creating a disk cache with the number of EADS servers set to `5` and the number of redundant copies of data plus the original set to `3`:

```
eads.cache.type=Disk
eads.cache.disk.info.dir=store
eads.cache.disk.1.dir=/hdd/cache_server01_range01
eads.cache.disk.2.dir=/hdd/cache_server01_range05
eads.cache.disk.3.dir=/hdd/cache_server01_range04
eads.cache.disk.filesize=128
eads.cache.disk.filenum=8
eads.cache.disk.blocksize=1
```

(2) Start the EADS servers

Log in to the host on which you plan to start an EADS server, and then execute the `ezstart` command to start the EADS server. You must execute this command for each EADS server.

```
ezstart
```

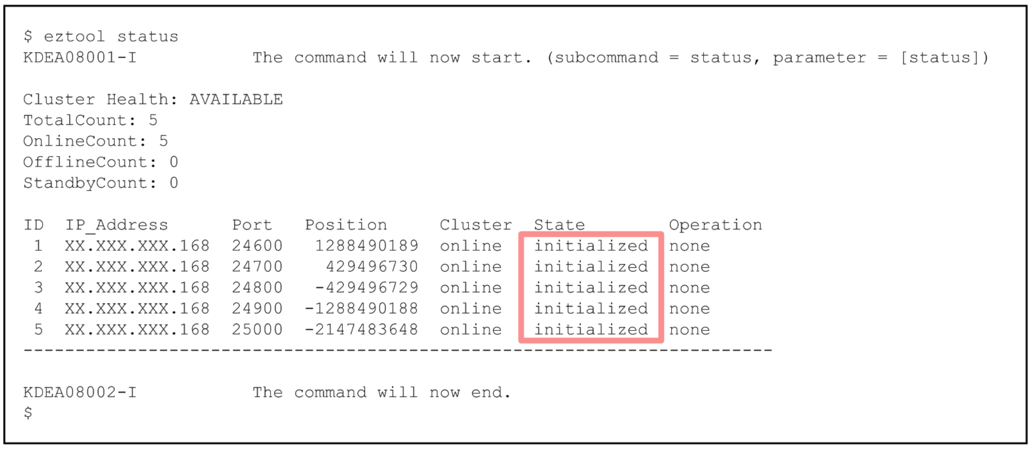
(3) Verify that initialization is finished

After an EADS server starts up successfully, it changes to initialized status.

Execute the `eztool status` command to verify that the initialization of all EADS servers is finished.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      Port   Position     Cluster  State        Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   initialized  none
 2  XX.XXX.XXX.168  24700    429496730  online   initialized  none
 3  XX.XXX.XXX.168  24800   -429496729  online   initialized  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   initialized  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   initialized  none
------------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

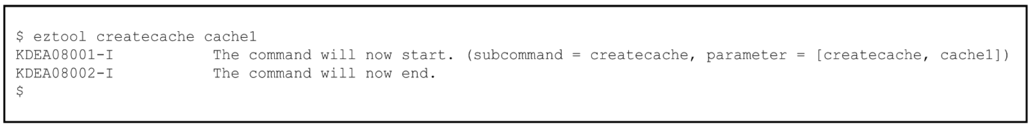
If the initialization is finished, `initialized` is displayed in the `State` column.

(4) Create caches

Execute the `eztool createcache` command to create a cache for storing keys and values.

```
eztool createcache cache-name
```

Command execution example



```
$ eztool createcache cache1
KDEA08001-I           The command will now start. (subcommand = createcache, parameter = [createcache, cache1])
KDEA08002-I           The command will now end.
$
```

When a cache is created, cache files are created in the following directories:

- Cache data file

directory-specified-in-the-eads.cache.disk.n.dir-parameter-in-the-cache-properties/*cache-name*

- Cache index file and cache information file

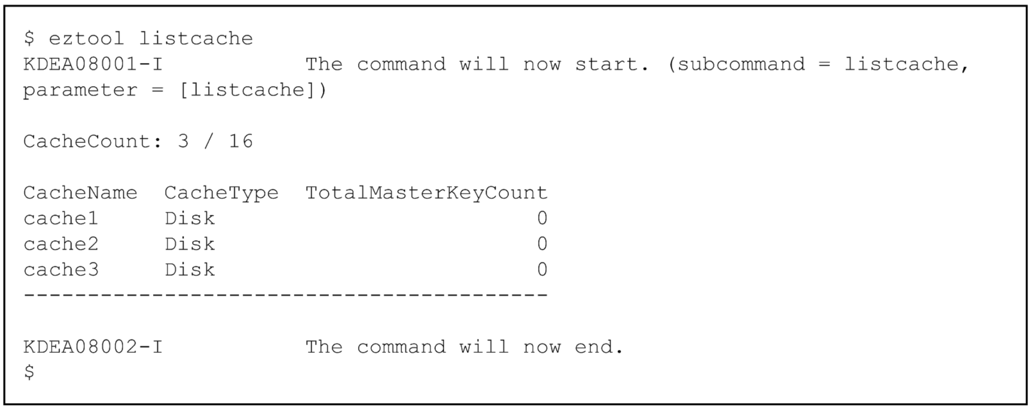
  directory-specified-in-the-eads.cache.disk.info.dir-parameter-in-the-cache-properties/*cache-name* (the default is *management-directory*/`store`/*cache-name*)

(5) Verify that caches have been created

Execute the `eztool listcache` command to display a list of caches.

```
eztool listcache
```

Command execution example



```
$ eztool listcache
KDEA08001-I           The command will now start. (subcommand = listcache,
parameter = [listcache])

CacheCount: 3 / 16

CacheName  CacheType  TotalMasterKeyCount
cache1     Disk                         0
cache2     Disk                         0
cache3     Disk                         0
-----------------------------------------

KDEA08002-I           The command will now end.
$
```

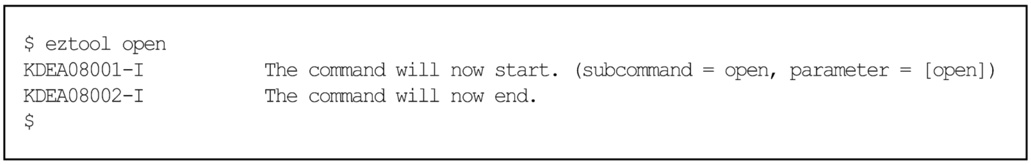
Check the cache names in the `CacheName` column and the cache types in the `CacheType` column to verify that the caches have been created as intended.

(6) Open the cluster

As is the case with closed status, in initialized status, the EADS servers do not accept requests from the EADS client. Execute the `eztool open` command to open the cluster.

```
eztool open
```

Command execution example



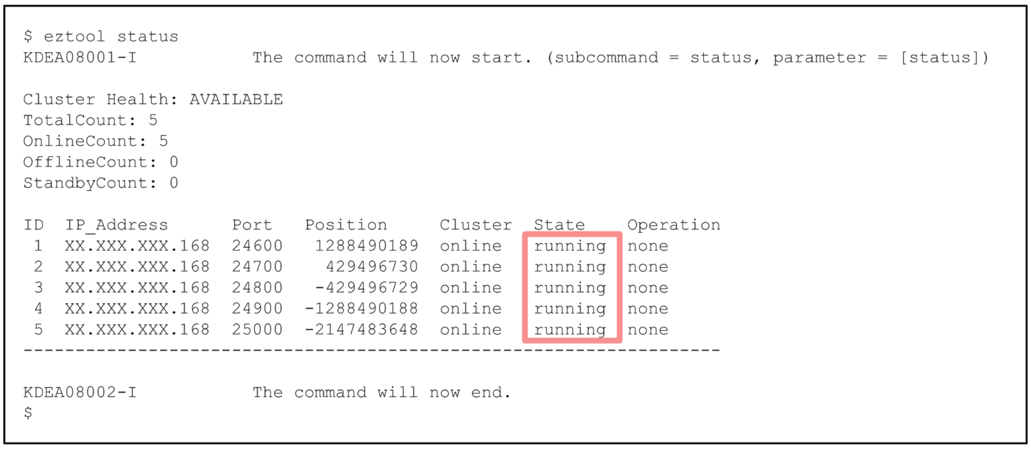
```
$ eztool open
KDEA08001-I           The command will now start. (subcommand = open, parameter = [open])
KDEA08002-I           The command will now end.
$
```

(7) Verify that the EADS servers have been opened

Execute the `eztool status` command to verify that all EADS servers have been opened.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      Port   Position     Cluster  State    Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   running  none
 2  XX.XXX.XXX.168  24700    429496730  online   running  none
 3  XX.XXX.XXX.168  24800   -429496729  online   running  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   running  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   running  none
-------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

If the EADS servers have been opened, `running` is displayed in the `State` column.

10.2.3 Notes about using multiple caches

Note the following when you use multiple caches:

- Fine-tuned operations cannot be performed for the caches individually, because the types of operations available to individual caches are limited.
- The number of cache-related resources increases.
- Performance might be affected adversely due to garbage collection because the size of the Java heap used by one process increases.

If there are too many caches, you can use the following methods to reduce the number of caches:

- Split the cluster instead of using multiple caches in the same cluster.
- When the same key name is used in different data items, use multiple groups, not multiple caches.

10.3 Starting the EADS servers (and creating caches by importing data from files)

After the EADS servers have started, data that was exported to the store data files during the previous session is imported back into memory. Alternatively, you can resume the caches after the EADS servers have started by accessing the cache files.

10.3.1 How to start the EADS servers (creating caches in memory)

This subsection explains how to start the EADS servers and then import back into memory cache the data that was exported to the store data files during the previous session.

(1) Start the EADS servers

Log in to the host on which you plan to start the EADS server, and then execute the `ezstart` command to start the EADS server. You must execute this command for each EADS server.

```
ezstart
```

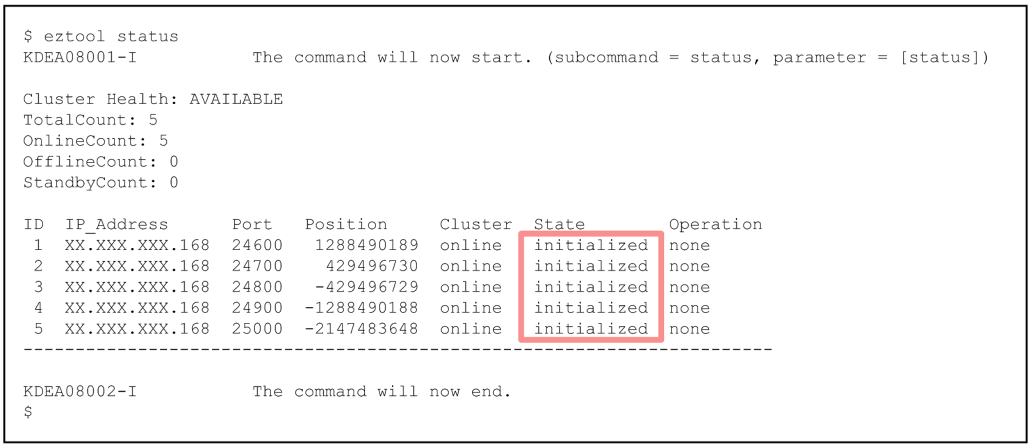
(2) Verify that initialization is finished

After an EADS server starts up successfully, it changes to initialized status.

Execute the `eztool status` command to verify that the initialization of all EADS servers is finished.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      Port   Position     Cluster  State        Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   initialized  none
 2  XX.XXX.XXX.168  24700    429496730  online   initialized  none
 3  XX.XXX.XXX.168  24800   -429496729  online   initialized  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   initialized  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   initialized  none
-------------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

If the initialization is finished, `initialized` is displayed in the `State` column.

(3) Import data into memory

Execute the `eztool import` command to import back into memory the most recent data that was exported to the store data files during the previous session.

```
eztool import
```

Command execution example

```
$ eztool import
KDEA08001-I          The command will now start. (subcommand = import, parameter = [import])
KDEA08054-I          The store data file was imported. (store data file key = stop_20130418100014)
KDEA08002-I          The command will now end.
$
```

As shown in this example, if a store data file key is omitted, the store data file with the most recent store data file key displayed in `latest` by the `eztool listesd` command is imported automatically.

You can specify the store data file key of any store data file.

Important note

If memory was almost full when the `eztool export` command was executed during the previous session, a data import operation might fail due to a memory shortage the next time the `eztool import` command is executed. This situation occurs, for example, if the `eztool export` command is executed while there is a memory shortage at the server on which a redundant copy of data is being created.

If this happens, increase the `eads.java.external.heapsize` parameter value in the shared properties, and then re-execute the `eztool import` command.

(4) Verify that caches have been created

Execute the `eztool listcache` command to display a list of caches.

```
eztool listcache
```

Command execution example

```
$ eztool listcache
KDEA08001-I          The command will now start. (subcommand = listcache, parameter = [listcache])

CacheCount: 3 / 16

CacheName  CacheType  TotalMasterKeyCount
cache1     Memory                       0
cache2     Memory                       0
cache3     Memory                       0
-----------------------------------------

KDEA08002-I          The command will now end.
$
```

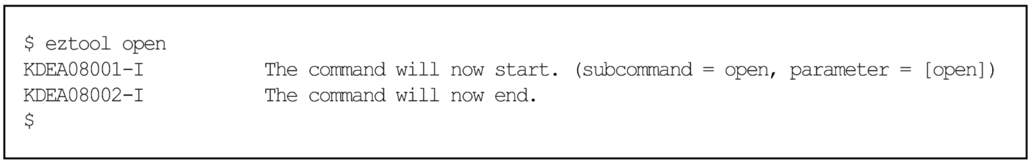
Check the cache names in the `CacheName` column and the cache types in the `CacheType` column to verify that the caches have been created as intended.

(5) Open the cluster

As is the case with closed status, in initialized status, the EADS servers do not accept requests from the EADS client. Execute the `eztool open` command to open the cluster.

```
eztool open
```

Command execution example



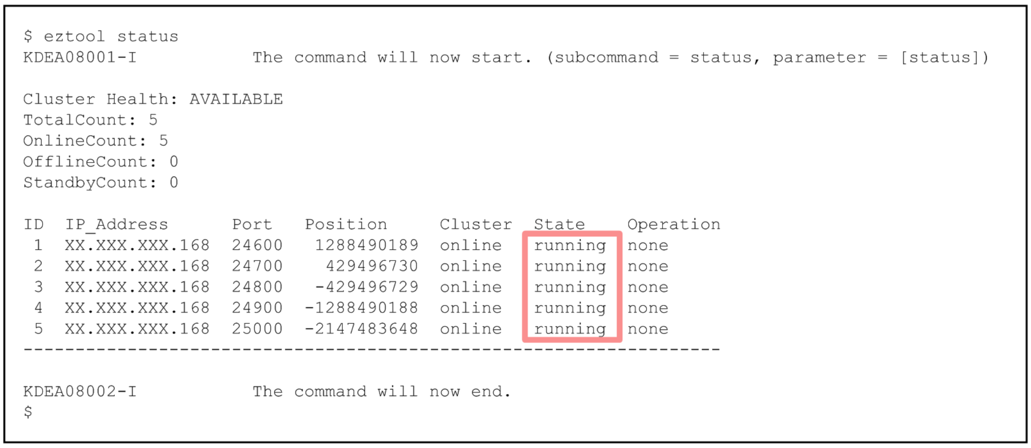
```
$ eztool open
KDEA08001-I           The command will now start. (subcommand = open, parameter = [open])
KDEA08002-I           The command will now end.
$
```

(6) Verify that the EADS servers have been opened

Execute the `eztool status` command to verify that all EADS servers have been opened.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      Port   Position     Cluster  State    Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   running  none
 2  XX.XXX.XXX.168  24700    429496730  online   running  none
 3  XX.XXX.XXX.168  24800   -429496729  online   running  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   running  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   running  none
-------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

If the EADS servers have been opened, `running` is displayed in the `State` column.

10.3.2 How to start the EADS servers (resuming caches on disk)

This subsection is applicable when you will be using disk caches or two-way caches.

This subsection explains how to start the EADS servers and then resume disk caches and two-way caches by accessing the cache files in the status they were in when the previous session terminated.

(1) Start the EADS servers

Log in to the host on which you plan to start the EADS server, and then execute the `ezstart` command to start the EADS server. You must execute this command for each EADS server.

```
ezstart
```

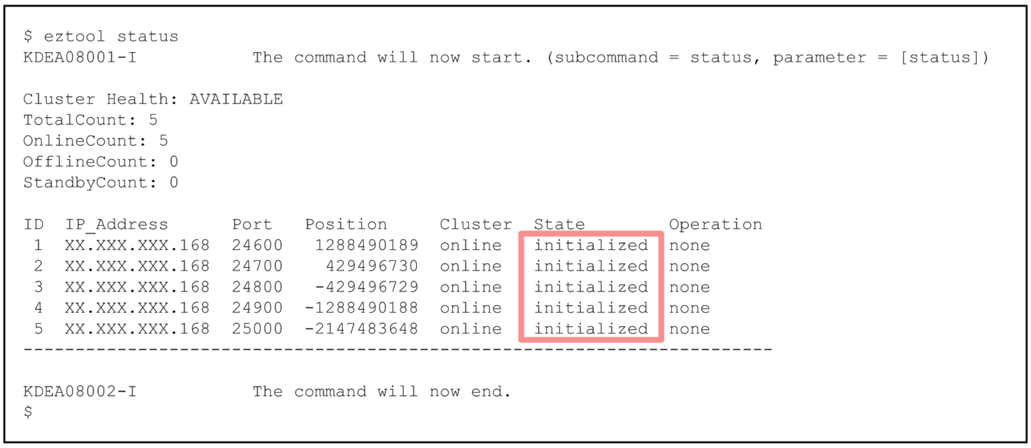
(2) Verify that initialization is finished

After an EADS server starts up successfully, it changes to initialized status.

Execute the `eztool status` command to verify that the initialization of all EADS servers is finished.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      Port   Position     Cluster  State         Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   initialized  none
 2  XX.XXX.XXX.168  24700    429496730  online   initialized  none
 3  XX.XXX.XXX.168  24800   -429496729  online   initialized  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   initialized  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   initialized  none
------------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

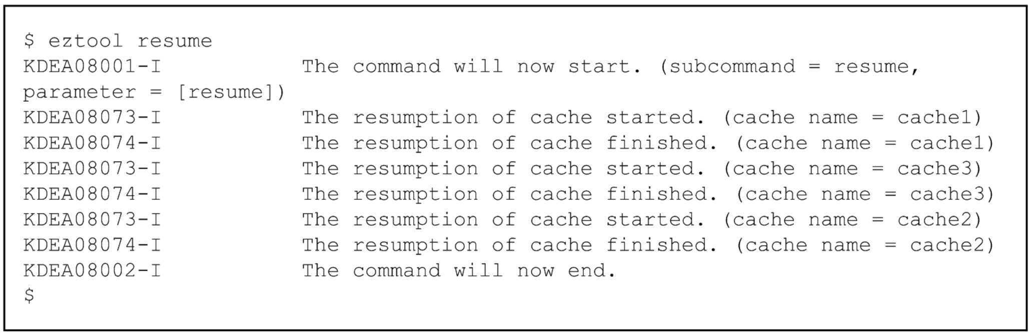
If the initialization is finished, `initialized` is displayed in the `State` column.

(3) Access cache files to resume the caches

Execute the `eztool resume` command to access the cache files and resume the caches.

```
eztool resume
```

Command execution example



```
$ eztool resume
KDEA08001-I           The command will now start. (subcommand = resume,
parameter = [resume])
KDEA08073-I           The resumption of cache started. (cache name = cache1)
KDEA08074-I           The resumption of cache finished. (cache name = cache1)
KDEA08073-I           The resumption of cache started. (cache name = cache3)
KDEA08074-I           The resumption of cache finished. (cache name = cache3)
KDEA08073-I           The resumption of cache started. (cache name = cache2)
KDEA08074-I           The resumption of cache finished. (cache name = cache2)
KDEA08002-I           The command will now end.
$
```

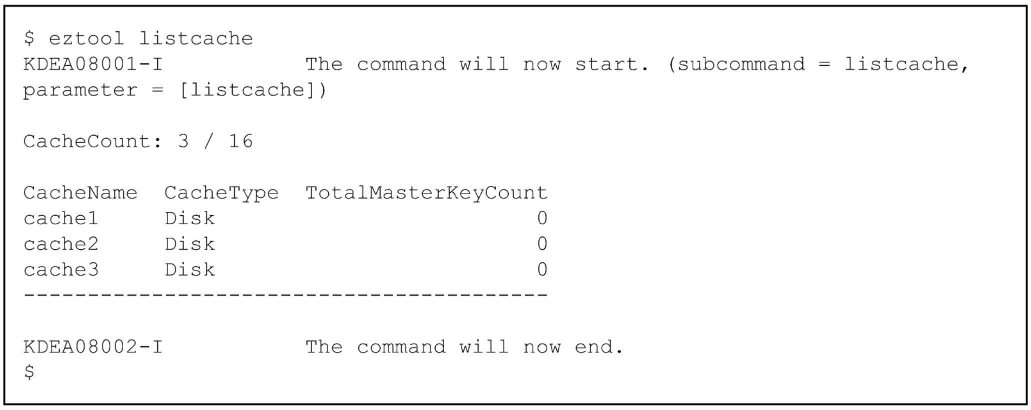
For a two-way cache, the contents of the cache data files are imported into memory when the cache is resumed.

(4) Verify that the caches have been resumed

Execute the `eztool listcache` command to display a list of caches.

```
eztool listcache
```

Command execution example



```
$ eztool listcache
KDEA08001-I           The command will now start. (subcommand = listcache,
parameter = [listcache])

CacheCount: 3 / 16

CacheName  CacheType  TotalMasterKeyCount
cache1     Disk                         0
cache2     Disk                         0
cache3     Disk                         0
-----------------------------------------

KDEA08002-I           The command will now end.
$
```

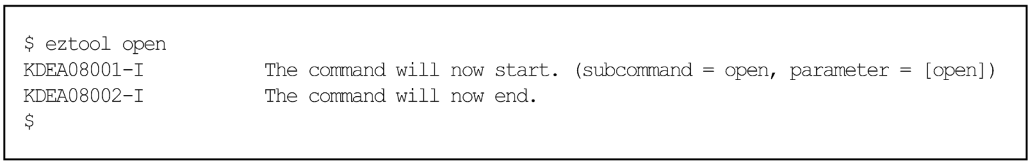
Check the cache names in the `CacheName` column and the cache types in the `CacheType` column to verify that the caches have been resumed as intended.

(5) Open the cluster

As is the case with closed status, in initialized status, the EADS servers do not accept requests from the EADS client. Execute the `eztool open` command to open the cluster.

```
eztool open
```

Command execution example



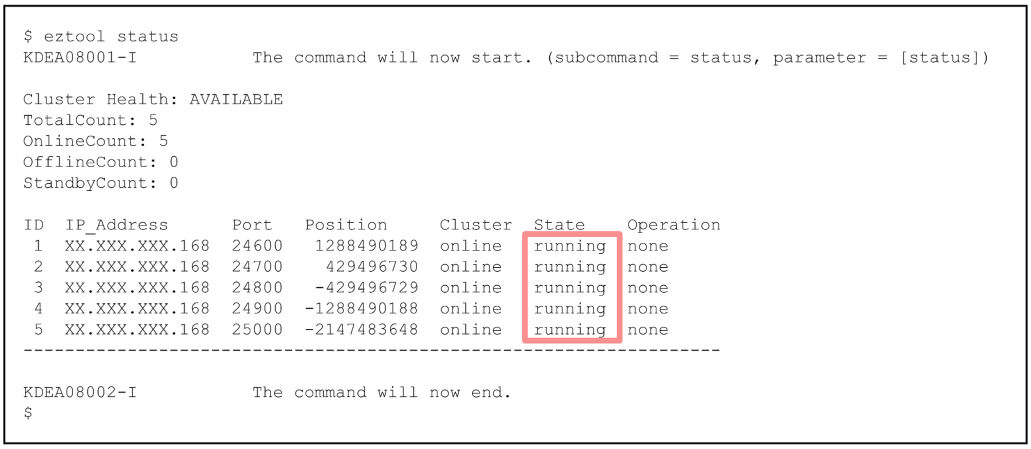
```
$ eztool open
KDEA08001-I           The command will now start. (subcommand = open, parameter = [open])
KDEA08002-I           The command will now end.
$
```

(6) Verify that the EADS servers have been opened

Execute the `eztool status` command to verify that all EADS servers have been opened.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      Port   Position     Cluster  State    Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   running  none
 2  XX.XXX.XXX.168  24700    429496730  online   running  none
 3  XX.XXX.XXX.168  24800   -429496729  online   running  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   running  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   running  none
-------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

If the EADS servers have been opened, `running` is displayed in the `State` column.

10.4 Terminating the EADS servers (and discarding data from memory)

The EADS servers are terminated without exporting data from memory caches to the store data files.

When the `--no_export` option is specified in the `eztool stop` command, all data is discarded from memory caches.

If you want to inherit the data in memory caches the next time you start the EADS servers, you must first export the data to store data files and then terminate the EADS servers. For details about this procedure, see *10.5 Terminating the EADS servers (after exporting data from memory to files)*.

10.4.1 How to terminate the EADS servers

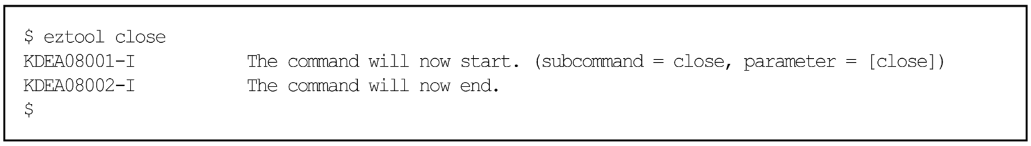
This subsection explains how to terminate the EADS servers without exporting data from memory caches to the store data files.

(1) Close the cluster

Execute the `eztool close` command to close the cluster.

```
eztool close
```

Command execution example



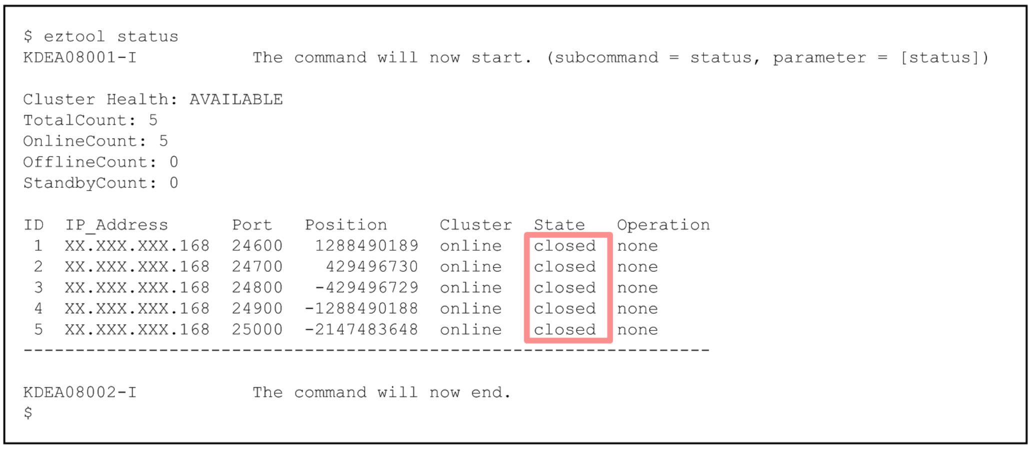
```
$ eztool close
KDEA08001-I          The command will now start. (subcommand = close, parameter = [close])
KDEA08002-I          The command will now end.
$
```

(2) Verify that the cluster is closed

After you have closed the EADS servers, execute the `eztool status` command to check the status of the cluster.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I          The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address       Port   Position     Cluster  State   Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   closed  none
 2  XX.XXX.XXX.168  24700    429496730  online   closed  none
 3  XX.XXX.XXX.168  24800   -429496729  online   closed  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   closed  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   closed  none
-------------------------------------------------------------------

KDEA08002-I          The command will now end.
$
```

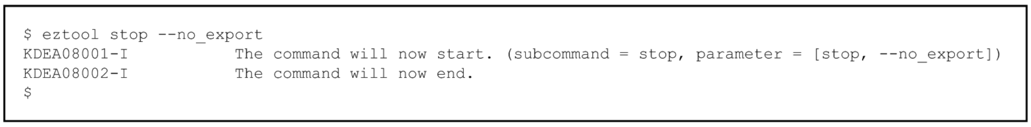
If the cluster is closed, `closed` is displayed in the `State` column.

(3) Terminate the EADS servers

Execute the `eztool stop --no_export` command to terminate all EADS servers in the cluster without exporting data from memory caches to the store data files.

```
eztool stop --no_export
```

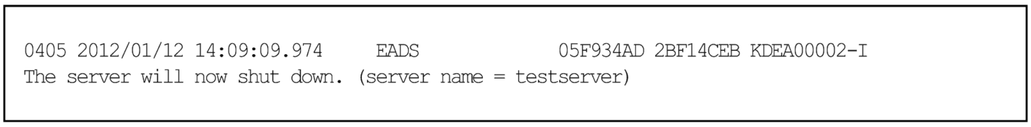
Command execution example



```
$ eztool stop --no_export
KDEA08001-I          The command will now start. (subcommand = stop, parameter = [stop, --no_export])
KDEA08002-I          The command will now end.
$
```

(4) Verify that the EADS servers have been terminated

Check the message logs of the EADS servers.



```
0405 2012/01/12 14:09:09.974     EADS             05F934AD 2BF14CEB KDEA00002-I
The server will now shut down. (server name = testserver)
```

10.5 Terminating the EADS servers (after exporting data from memory to files)

Data is exported from the memory caches to the store data files before terminating the EADS servers so that the data can be reused the next time the EADS servers are started.

10.5.1 How to terminate the EADS servers

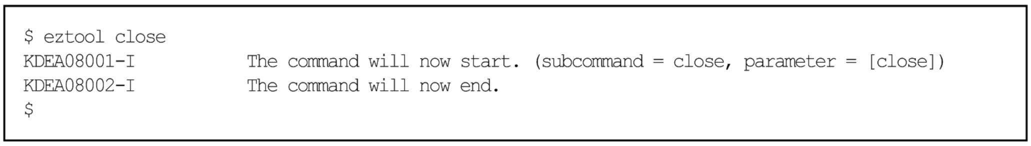
This subsection explains how to export data from memory caches to the store data files before terminating the EADS servers so that the data can be reused the next time the EADS servers are started.

(1) Close the cluster

Execute the `eztool close` command to close the cluster.

```
eztool close
```

Command execution example



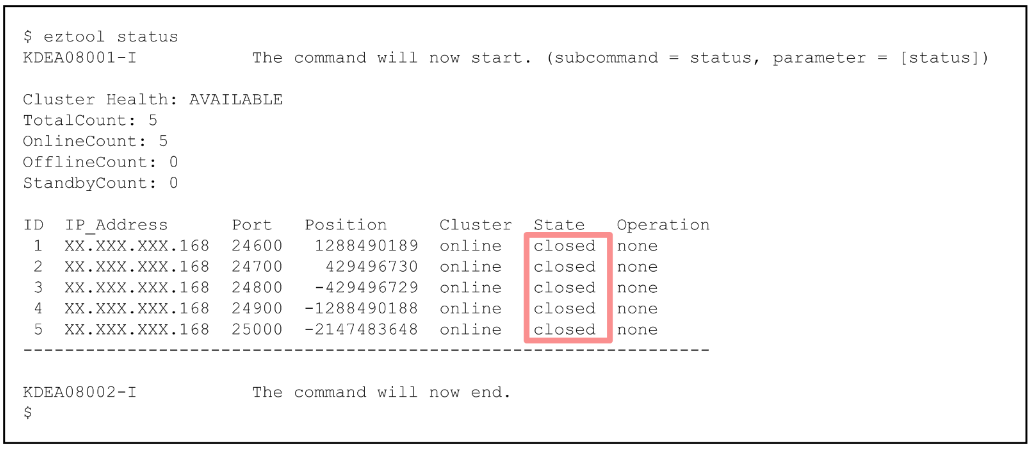
```
$ eztool close
KDEA08001-I          The command will now start. (subcommand = close, parameter = [close])
KDEA08002-I          The command will now end.
$
```

(2) Verify that the cluster is closed

After you have closed the EADS servers, execute the `eztool status` command to check the status of the cluster.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I          The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address       Port   Position     Cluster  State   Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   closed  none
 2  XX.XXX.XXX.168  24700    429496730  online   closed  none
 3  XX.XXX.XXX.168  24800   -429496729  online   closed  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   closed  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   closed  none
------------------------------------------------------------------

KDEA08002-I          The command will now end.
$
```

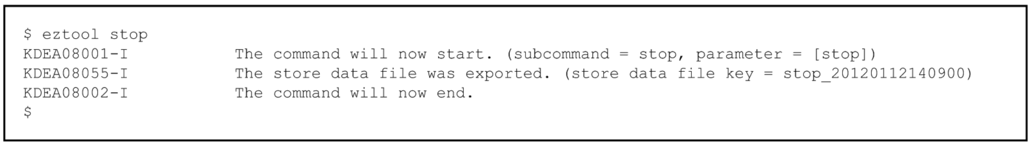
If the cluster is closed, `closed` is displayed in the `State` column.

(3) Terminate the EADS servers

Execute the `eztool stop` command to terminate all EADS servers in the cluster after exporting the most recent data to the store data files.

```
eztool stop
```

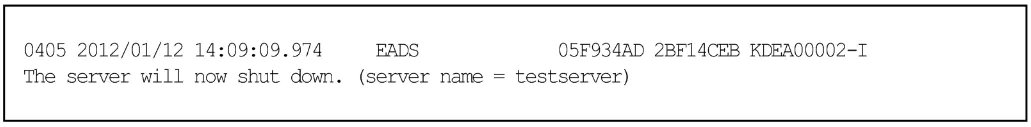
Command execution example



```
$ eztool stop
KDEA08001-I           The command will now start. (subcommand = stop, parameter = [stop])
KDEA08055-I           The store data file was exported. (store data file key = stop_20120112140900)
KDEA08002-I           The command will now end.
$
```

(4) Verify that the EADS servers have been terminated

Check the message logs of the EADS servers.



```
0405 2012/01/12 14:09:09.974     EADS           05F934AD 2BF14CEB KDEA00002-I
The server will now shut down. (server name = testserver)
```

10.6 Terminating the EADS servers (terminating caches on disk)

This section is applicable when you will be using disk caches or two-way caches.

The EADS servers that use disk caches or two-way caches are terminated.

10.6.1 How to terminate the EADS servers

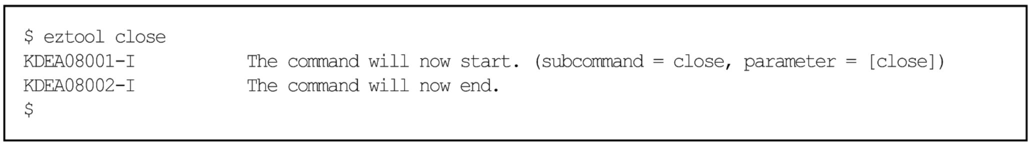
This subsection explains how to terminate disk caches and two-way caches.

(1) Close the cluster

Execute the `eztool close` command to close the cluster.

```
eztool close
```

Command execution example



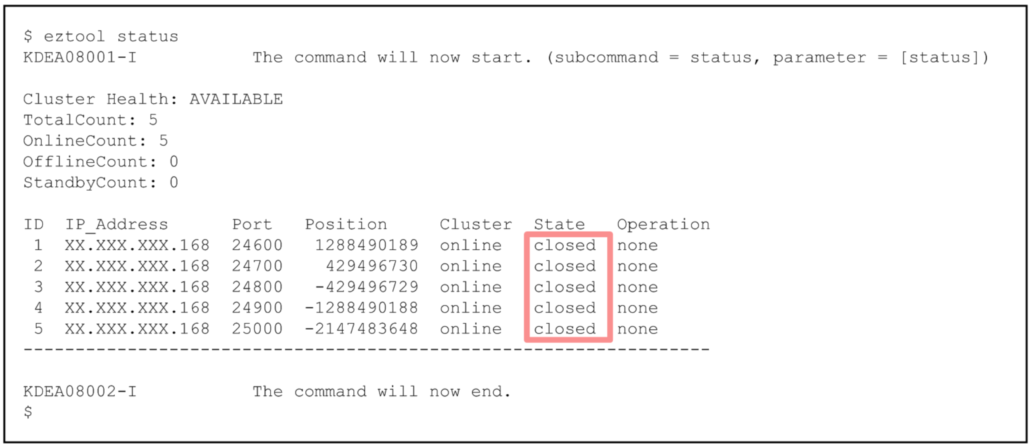
```
$ eztool close
KDEA08001-I          The command will now start. (subcommand = close, parameter = [close])
KDEA08002-I          The command will now end.
$
```

(2) Verify that the cluster is closed

After you have closed the EADS servers, execute the `eztool status` command to check the status of the cluster.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address       Port   Position     Cluster  State   Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   closed  none
 2  XX.XXX.XXX.168  24700    429496730  online   closed  none
 3  XX.XXX.XXX.168  24800   -429496729  online   closed  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   closed  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   closed  none
------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

If the cluster is closed, `closed` is displayed in the `State` column.

(3) Terminate the EADS servers

Execute the `eztool stop` command to terminate the disk caches and two-way caches and then terminate all EADS servers in the cluster.

```
eztool stop
```

Command execution example

```
$ eztool stop
KDEA08001-I           The command will now start. (subcommand = stop,
parameter = [stop])
KDEA08079-I           Exporting was not executed because no memory cache
exist on the server.
KDEA08002-I           The command will now end.
$
```

If there are no memory caches on the EADS servers, no store data files are created when the `eztool stop` command is executed.

(4) Verify that the EADS servers have been terminated

Check the message logs of the EADS servers.

```
0405 2012/01/12 14:09:09.974     EADS            05F934AD 2BF14CEB KDEA00002-I
The server will now shut down. (server name = testserver)
```

10.7 Checking the cluster and EADS server statuses

Execute the `eztool status` command to check the statuses of the cluster and the EADS servers.

See *2.11 Cluster and EADS server status transitions*.

```
eztool status [-v|--verbose]
```

Command execution example

```
$ eztool status
KDEA08001-I          The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      Port   Position     Cluster  State    Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   running  none
 2  XX.XXX.XXX.168  24700    429496730  online   running  none
 3  XX.XXX.XXX.168  24800   -429496729  online   running  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   running  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   running  none
-------------------------------------------------------------------

KDEA08002-I          The command will now end.
$
```

You can display the detailed information by specifying the `-v` or `--verbose` option.

```
$ eztool status -v
KDEA08001-I          The command will now start. (subcommand = status, parameter = [status, -v])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      ServerName  Port   Position     Cluster  State    Operation  Lock    KeyCount  UsedMemoryRatio  UsedMemory  MaxMemory  Version
 1  XX.XXX.XXX.168  server01    24600   1288490189  online   running  none       unlock       582               13           3         23  04-00-00
 2  XX.XXX.XXX.168  server02    24700    429496730  online   running  none       unlock       618               13           3         23  04-00-00
 3  XX.XXX.XXX.168  server03    24800   -429496729  online   running  none       unlock       636               13           3         23  04-00-00
 4  XX.XXX.XXX.168  server04    24900  -1288490188  online   running  none       unlock       591               13           3         23  04-00-00
 5  XX.XXX.XXX.168  server05    25000  -2147483648  online   running  none       unlock       573               13           3         23  04-00-00
---------------------------------------------------------------------------------------------------------------------------------------------------

KDEA08002-I          The command will now end.
$
```

10.8 Displaying a list of caches

Execute the `eztool listcache` command to display a list of caches.

```
eztool listcache [-v|--verbose]
```

Command execution example

```
$ eztool listcache
KDEA08001-I           The command will now start. (subcommand = listcache, parameter = [listcache])

CacheCount: 3 / 16

CacheName  CacheType  TotalMasterKeyCount
cache1     Memory                       0
cache2     Memory                       0
cache3     Memory                       0
-----------------------------------------

KDEA08002-I           The command will now end.
$
```

You can display the details by specifying the `-v` or `--verbose` option.

```
$ eztool listcache -v
KDEA08001-I           The command will now start. (subcommand = listcache, parameter =
[listcache, -v])

CacheCount: 3 / 16

CacheName  CacheType  TotalMasterKeyCount  Server                MasterKeyCount
cache1     Memory                       0  XX.XXX.XXX.168:24600               0
                                           XX.XXX.XXX.168:24700               0
                                           XX.XXX.XXX.168:24800               0
                                           XX.XXX.XXX.168:24900               0
                                           XX.XXX.XXX.168:25000               0
cache2     Memory                       0  XX.XXX.XXX.168:24600               0
                                           XX.XXX.XXX.168:24700               0
                                           XX.XXX.XXX.168:24800               0
                                           XX.XXX.XXX.168:24900               0
                                           XX.XXX.XXX.168:25000               0
cache3     Memory                       0  XX.XXX.XXX.168:24600               0
                                           XX.XXX.XXX.168:24700               0
                                           XX.XXX.XXX.168:24800               0
                                           XX.XXX.XXX.168:24900               0
                                           XX.XXX.XXX.168:25000               0
-------------------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

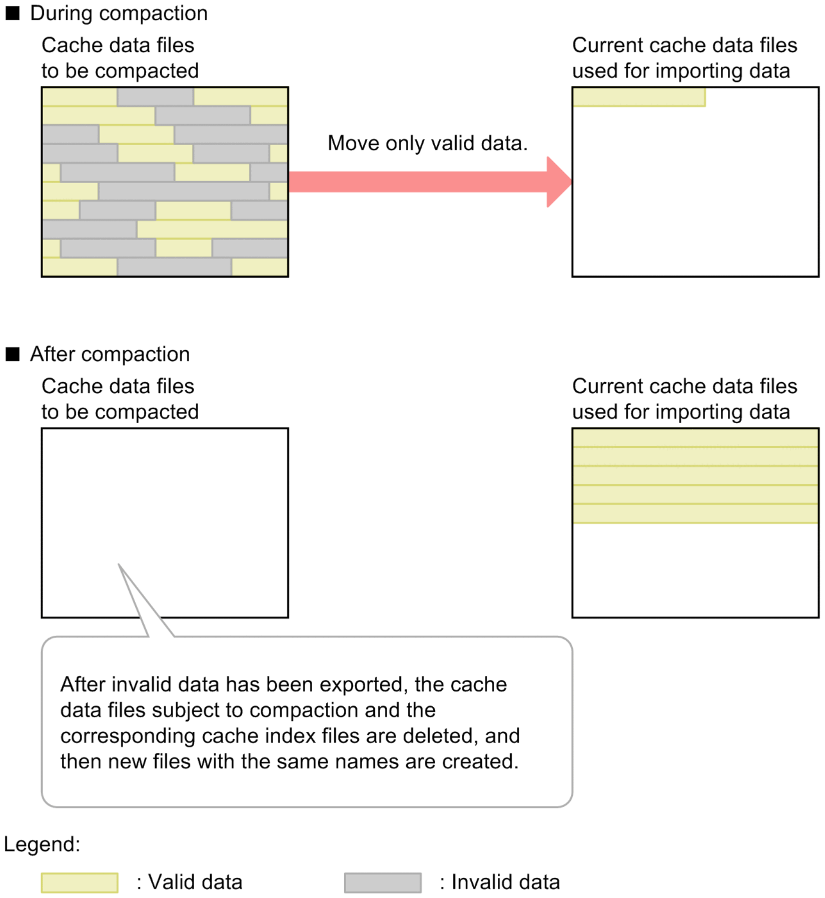
10.9 Reducing the data usage of cache data files (performing compaction on cache data files)

This section is applicable when you will be using disk caches or two-way caches.

With disk caches and two-way caches, data is stored in cache data files by using an appending methodology. This means that when data is updated or deleted, invalid data remains in the files. Deleting such invalid data to reduce the data usage of the cache data files is called *compaction*.

The following figure illustrates compaction processing.

Figure 10–2: Compaction processing


■ During compaction
Cache data files to be compacted
Move only valid data.
Current cache data files used for importing data
■ After compaction
Cache data files to be compacted
Current cache data files used for importing data
After invalid data has been exported, the cache data files subject to compaction and the corresponding cache index files are deleted, and then new files with the same names are created.
Legend:
: Valid data
: Invalid data

Note that the file size specified in the `eads.cache.disk.filesize` parameter in the cache properties remains unchanged after compaction.

Reference note

EADS employs an appending methodology to update cache data files. Data that is deleted using this method becomes invalid, but it is not actually deleted from the disk. When data is updated, the existing data that is updated becomes invalid and new data is added. The existing data that has become invalid is not deleted from the disk.

By performing compaction periodically, you can remove invalid data and use the disk efficiently.

10.9.1 Performing compaction on cache data files

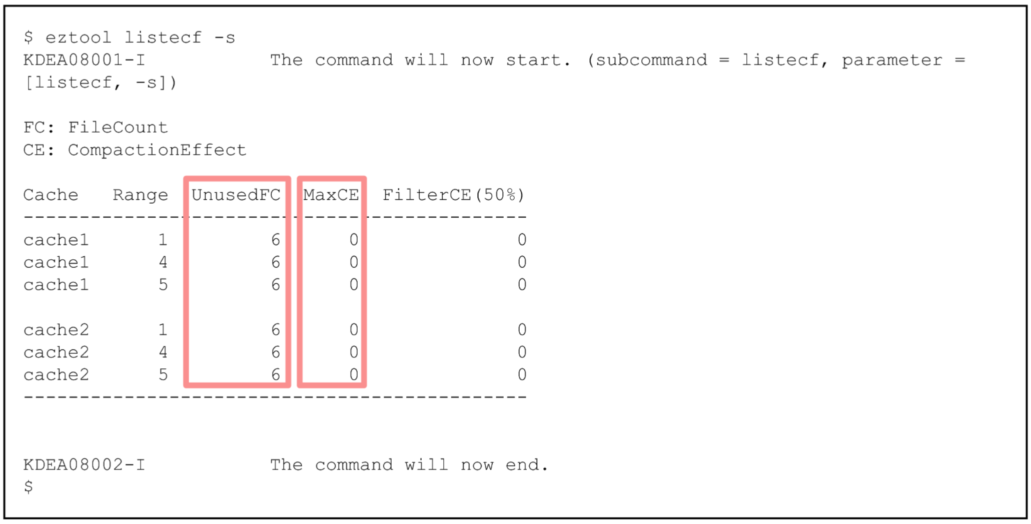
This subsection explains how to perform compaction on cache data files.

(1) Checking the compaction effects

Execute the `eztool listecf -s` command to check the compaction effects.

```
eztool listecf -s
```

Command execution example



```
$ eztool listecf -s
KDEA08001-I           The command will now start. (subcommand = listecf, parameter =
[listecf, -s])

FC: FileCount
CE: CompactionEffect

Cache   Range  UnusedFC  MaxCE  FilterCE(50%)
---------------------------------------------
cache1      1         6      0              0
cache1      4         6      0              0
cache1      5         6      0              0

cache2      1         6      0              0
cache2      4         6      0              0
cache2      5         6      0              0
---------------------------------------------


KDEA08002-I           The command will now end.
$
```

- The `UnusedFC` column displays the number of unused cache data files. Monitor this column when you store (`put`) or delete (`remove`) data randomly without regard to the order in which the data was stored (`put`).
- The `MaxCE` column displays the maximum compaction effects for the cache data files in each range. If you delete (`remove`) data in the order in which the data was stored (`put`) and this column displays 100% or a value close to 100%, you need to perform compaction.

(2) Close the cluster (optional)

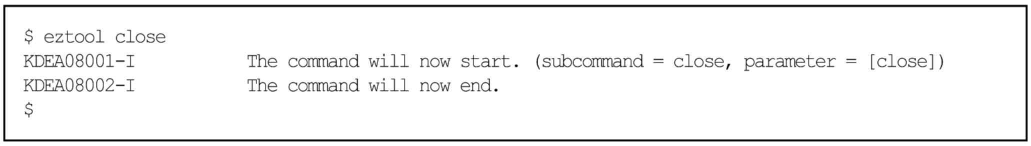
If necessary, execute the `eztool close` command to close the cluster.

Reference note

You can perform compaction while the EADS servers are running, but performance might be affected adversely.

```
eztool close
```

Command execution example



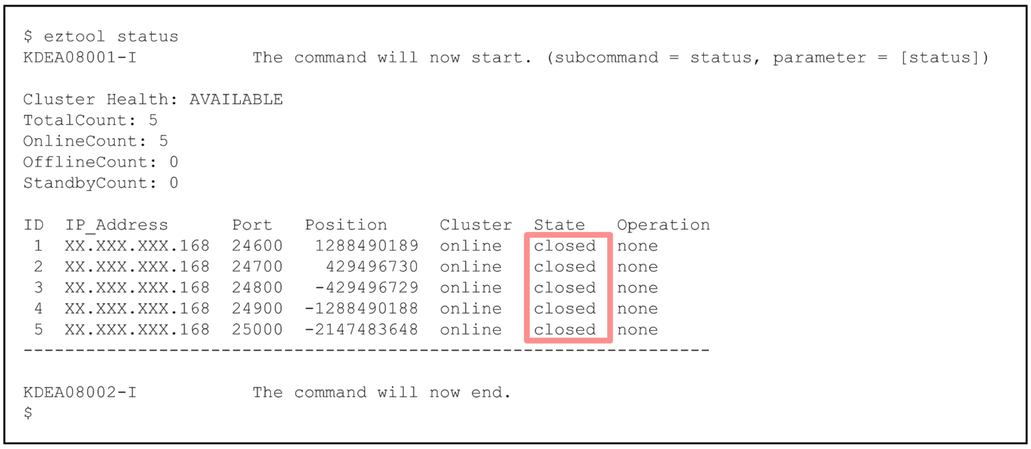
```
$ eztool close
KDEA08001-I           The command will now start. (subcommand = close, parameter = [close])
KDEA08002-I           The command will now end.
$
```

(3) Verify that the cluster is closed (optional)

If you closed the cluster in *10.9.1(2) Close the cluster (optional)*, execute the `eztool status` command to check the cluster status.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I          The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address       Port   Position     Cluster  State   Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   closed  none
 2  XX.XXX.XXX.168  24700    429496730  online   closed  none
 3  XX.XXX.XXX.168  24800   -429496729  online   closed  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   closed  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   closed  none
------------------------------------------------------------------

KDEA08002-I          The command will now end.
$
```

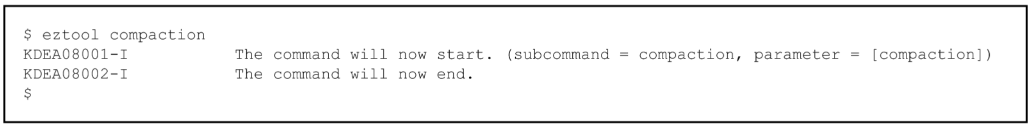
If the cluster is closed, `closed` is displayed in the `State` column.

(4) Perform compaction on the cache data files

Execute the `eztool compaction` command to perform compaction.

```
eztool compaction
```

Command execution example



```
$ eztool compaction
KDEA08001-I          The command will now start. (subcommand = compaction, parameter = [compaction])
KDEA08002-I          The command will now end.
$
```

Important note

Because compaction processing might take a long time, design a timeout value, if necessary.

(a) When you delete data in the order in which the data was stored

When you delete (`remove`) data in the order in which the data was stored (`put`), invalid data is created in cache data files each time data is deleted. In such a case, perform compaction on the cache data files whose entire contents are invalid data. Execute the `eztool compaction` command with the `--threshold` option specifying a threshold of 100%.

Reference note

If the order in which the data was stored (`put`) does not exactly match the order in which data is deleted (`remove`), set the threshold to a value that is smaller than 100%.

(b) When you store or delete data randomly without regard to the order in which the data was stored

When you store (`put`) or delete (`remove`) data randomly without regard to the order in which data is stored (`put`), the file usage rate will differ from one cache data file to another. For this reason, you need to allocate unused files so that data can be added and deleted. In this case, perform compaction periodically based on the number of unused files. Execute the `eztool compaction` command with the `--unused_fc` option specifying the number of unused files to be allocated.

If the planned number of unused files cannot be obtained after compaction processing, reduce the threshold with the `--threshold` option and then perform compaction again.

(c) Allocating space to cache data files immediately

When the number of unused files is very small and you need to allocate space to the cache data files immediately, perform compaction with a smaller threshold specified in the `--threshold` option. Execute the `eztool compaction` command with the priority for caches and ranges specified in the `--cache` and `--range` options.

If compaction is already underway, stop the compaction processing and then re-execute the `eztool compaction` command. For details about how to stop compaction processing, see *10.9.2 Stopping compaction processing*.

Reference note

If the amount of available space in the cache data files is very small, we recommend that you close the cluster before performing compaction processing, if possible.

(5) Open the cluster (optional)

If you closed the cluster in *10.9.1(2) Close the cluster (optional)*, execute the `eztool open` command to open the cluster.

```
eztool open
```

Command execution example

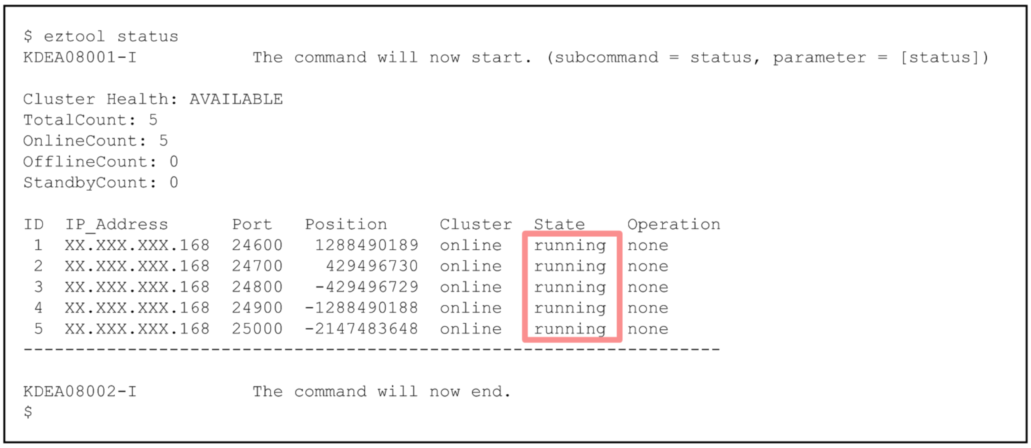
```
$ eztool open
KDEA08001-I         The command will now start. (subcommand = open, parameter = [open])
KDEA08002-I         The command will now end.
$
```

(6) Verify that the EADS servers have been opened

If you closed the cluster in *10.9.1(2) Close the cluster (optional)*, execute the `eztool status` command to verify that all EADS servers have been opened.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      Port   Position     Cluster  State    Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   running  none
 2  XX.XXX.XXX.168  24700    429496730  online   running  none
 3  XX.XXX.XXX.168  24800   -429496729  online   running  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   running  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   running  none
--------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

If the EADS servers have been opened, `running` is displayed in the `State` column.

10.9.2 Stopping compaction processing

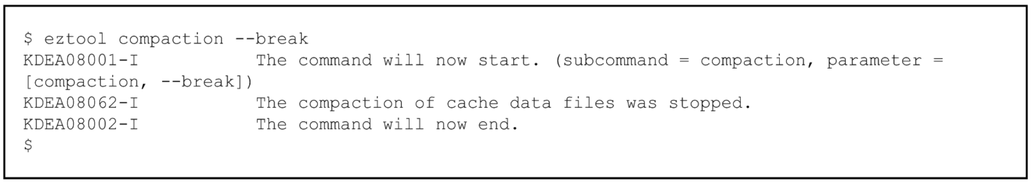
This subsection explains how to stop compaction processing that is underway.

(1) Stop compaction processing

While compaction processing is underway, execute the `eztool compaction --break` command.

```
eztool compaction --break
```

Command execution example



```
$ eztool compaction --break
KDEA08001-I           The command will now start. (subcommand = compaction, parameter =
[compaction, --break])
KDEA08062-I           The compaction of cache data files was stopped.
KDEA08002-I           The command will now end.
$
```

Compaction processing is stopped after the compaction processing on the current cache data file has been completed.

11 Maintenance Operations

This chapter explains how to run maintenance operations in EADS.

11.1 Adding EADS servers to a cluster

This section explains how to add EADS servers to a cluster.

11.1.1 How to add EADS servers to a cluster without stopping the cluster (scale-out processing)

This subsection explains how to add EADS servers to a cluster without stopping the cluster (scale-out processing).

To add EADS servers to a cluster without stopping the cluster, the following conditions must be satisfied:

- All caches in the cluster are memory caches.
- The number of redundant copies of data is 2 or more.

If there are disk caches or two-way caches or the number of redundant copies of data plus the original is 1, stop the cluster and then add EADS servers.

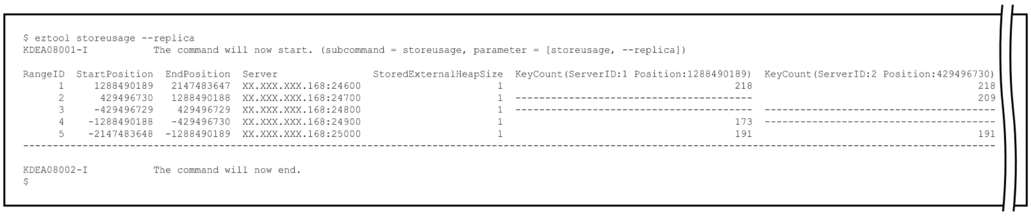
For details about how to add EADS servers after stopping a cluster, see *11.1.2 How to add EADS servers to a cluster after stopping the cluster (using only memory caches)* and *11.1.3 How to add EADS servers to a cluster after stopping the cluster (using only disk caches)*.

(1) Check the data distribution among the ranges

Execute the `eztool storeusage --replica` command to check the distribution of the data among the ranges.

```
eztool storeusage --replica
```

Command execution example



```
$ eztool storeusage --replica
KDEA08001-I          The command will now start. (subcommand = storeusage, parameter = [storeusage, --replica])

RangeID  StartPosition  EndPosition  Server                 StoredExternalHeapSize  KeyCount(ServerID:1 Position:1288490189)  KeyCount(ServerID:2 Position:429496730)
      1     1288490189   2147483647  XX.XXX.XXX.168:24600                        1                                       218                                      218
      2      429496730   1288490188  XX.XXX.XXX.168:24700                        1  ----------------------------------------                                      209
      3     -429496729    429496729  XX.XXX.XXX.168:24800                        1  ----------------------------------------  ---------------------------------------
      4    -1288490188   -429496730  XX.XXX.XXX.168:24900                        1                                       173  ---------------------------------------
      5    -2147483648  -1288490189  XX.XXX.XXX.168:25000                        1                                       191                                      191
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------

KDEA08002-I          The command will now end.
$
```

Based on the execution results of the `eztool storeusage --replica` command, determine the ranges within which EADS servers need to be added.

> **Tip**
>
> If keys are distributed evenly among the ranges, you can realize the benefits of scale-out processing over the entire cluster by adding one or more EADS servers to each range.
>
> On the other hand, sufficient benefits might not be obtained over the entire cluster in the following cases:
>
> - Keys are distributed evenly among the ranges and no EADS server is added to some of the ranges.
> - Keys are not distributed evenly among the ranges.

(2) Back up the cluster property files (optional)

Back up the cluster property files, if necessary.

When EADS servers are added to an active cluster, the cluster property files are updated automatically based on the information applicable after the EADS servers have been added.

A backup cluster property file containing the current information is created automatically at the path shown below. If another file with the same name already exists, the backup file is not output.

```
management-directory/conf/eads_cluster.properties.ebf
```

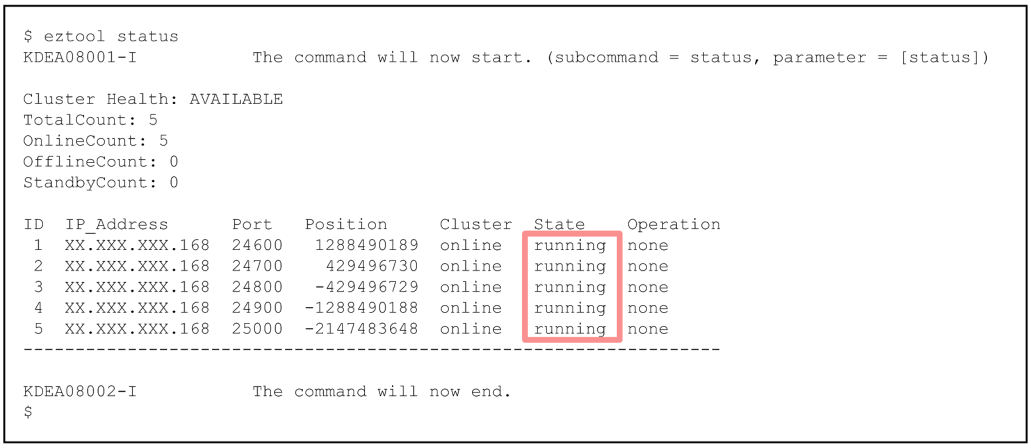
If a backup cluster property file already exists, the cluster property file is updated during scale-out processing to reflect the new information after the EADS servers have been added. Be aware that if you edit the cluster property file after scale-out processing has been performed and then execute scale-out processing again, the edited information is lost. If you need a backup copy of such an edited cluster property file, back up that cluster property file manually before you perform scale-out processing.

(3) Verify that there are no errors in any of the EADS servers in the cluster

Execute the `eztool status` command to verify that none of the EADS servers in the cluster is in isolated status (`isolated`) or stopped status (`-----------`).

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address       Port   Position     Cluster  State    Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   running  none
 2  XX.XXX.XXX.168  24700    429496730  online   running  none
 3  XX.XXX.XXX.168  24800   -429496729  online   running  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   running  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   running  none
--------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

If there are isolated or stopped EADS servers in the cluster, restore them.

For details about the restoration procedure, see *12.2.1 If one or more EADS servers are isolated*.

(4) Install and set up the EADS servers to be added

Install and set up the EADS servers you want to add. For details about how to install and set up EADS servers, see *5. Installing and Setting Up (EADS Servers)*.

There is no need to create cluster property files because these files are created automatically when EADS servers are added (if a cluster property file already exists for an EADS server to be added, that cluster property file is not imported).

(5) Add an EADS server to the cluster

Execute on an EADS server to be added the `ezstart` or `ezserver` command to add it to the cluster.

The following are the methods for adding an EADS server to a cluster:

- Adding an EADS server to a cluster by specifying its EADS server ID
  - Execute the `ezstart -ai` command on the EADS server to be added.
  - Execute the `ezserver -ai` command on the EADS server to be added.
- Adding an EADS servers to a cluster by specifying the EADS server position (hash value)
  - Execute the `ezstart -ap` command on the EADS server to be added.
  - Execute the `ezserver -ap` command on the EADS server to be added.

Adding an EADS server to a cluster by specifying its EADS server ID (executing the ezstart -ai command)

```
ezstart -ai EADS-server-ID
```

Adding an EADS servers to a cluster by specifying its EADS server ID (executing the ezserver -ai command)

```
ezserver -ai EADS-server-ID
```

Adding an EADS server to a cluster by specifying the EADS server position (executing the ezstart -ap command)

```
ezstart -ap EADS-server-location-(hash-value)
```

Adding an EADS server to a cluster by specifying the EADS server position (executing the ezserver -ap command)

```
ezserver -ap EADS-server-location-(hash-value)
```

If the cluster property files have not been updated for some reason after an EADS server has been added, an error message is displayed. In such a case, execute the `eztool listconf` command to check the most recent parameter values and apply those values manually to the cluster property files.

(6) Verify that the added EADS server is participating in the cluster

Execute the `eztool status` command to verify that the added EADS server is participating in the cluster.

```
eztool status
```

Command execution example

```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status,
parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 6
OnlineCount: 6
OfflineCount: 0
StandbyCount: 0

ID  IP_Address       Port   Position     Cluster  State    Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   running  none
 2  XX.XXX.XXX.168  24700    429496730  online   running  none
 6  XX.XXX.XXX.168  25100            0  online   running  none
 3  XX.XXX.XXX.168  24800   -429496729  online   running  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   running  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   running  none
-------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

If an EADS server is participating in the cluster, `online` is displayed in the `Cluster` column.

You add multiple EADS servers by performing for each EADS server to be added the procedure that follows *11.1.1(4) Install and set up the EADS servers to be added*.

11.1.2 How to add EADS servers to a cluster after stopping the cluster (using only memory caches)

This subsection explains how to add EADS servers to a cluster after stopping the cluster when using only memory caches.

(1) Terminate all EADS servers in the cluster (after exporting data to files)

Export data to files and then terminate all EADS servers in the cluster.

For details about this procedure, see *10.5 Terminating the EADS servers (after exporting data from memory to files)*.

(2) Install and set up the EADS servers to be added

Install and set up the EADS servers you want to add. For details about how to install and set up EADS servers, see *5. Installing and Setting Up (EADS Servers)*.

(3) Change the cluster properties

Because the number of EADS servers that make up the cluster is increasing, you must change the cluster properties.

For details about how to change the properties, see *11.4 Changing the properties*.

(4) Start all EADS servers in the cluster (and import data from files)

Start all EADS servers in the cluster, and then import back into memory the data that was exported to files during the previous session.

For details about the procedure, see *10.3 Starting the EADS servers (and creating caches by importing data from files)*.

11.1.3 How to add EADS servers to a cluster after stopping the cluster (using only disk caches)

This subsection is applicable when you will be using disk caches and two-way caches.

This subsection explains how to add EADS servers to a cluster after stopping the cluster when using disk caches or two-way caches.

(1) Terminate all EADS servers in the cluster

Terminate all EADS servers in the cluster.

For details about this procedure, see *10.6 Terminating the EADS servers (terminating caches on disk)*.

(2) Move cache files for all EADS servers

Move the cache files for all EADS servers to desired directories.

For each server, move all files under the following directories (copy all files under the following directories and then delete the source files):

- *directory-specified-in-the-eads.cache.disk.n.dir-parameter-in-the-cache-properties*/*cache-name*
- *directory-specified-in-the-eads.cache.disk.info.dir-parameter-in-the-cache-properties*/*cache-name* (the default is *management-directory*/`store`/*cache-name*)

Make sure that the paths of the target directories are identical on all the EADS servers.

(3) Install and set up the EADS servers to be added

Install and set up the EADS servers you want to add. For details about how to install and set up EADS servers, see *5. Installing and Setting Up (EADS Servers)*.

(4) Change the cluster properties

Because the number of EADS servers that make up the cluster is increasing, you must change the cluster properties.

For details about how to change the properties, see *11.4 Changing the properties*.

(5) Start all EADS servers in the cluster

Log in to the host on which you plan to start the EADS server, and then execute the `ezstart` command to start the EADS server. You must execute this command for each EADS server.

```
ezstart
```

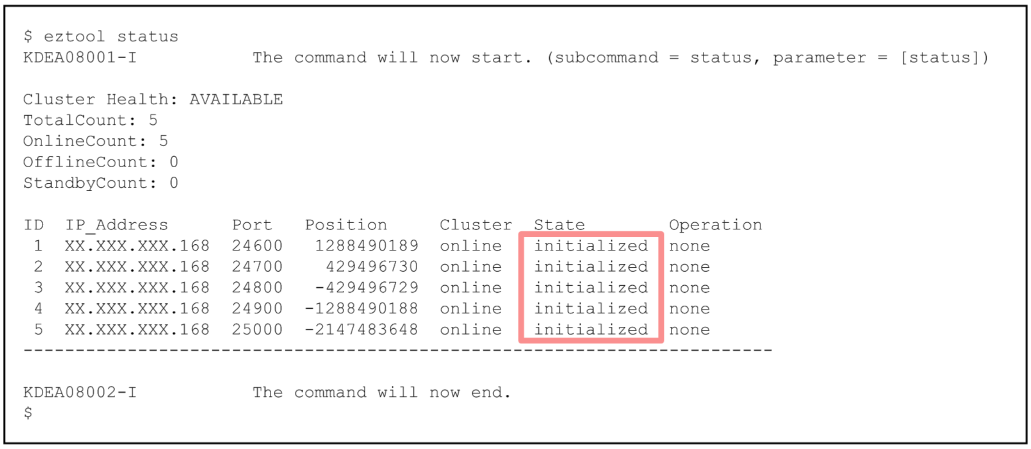
(6) Verify that initialization is finished

After an EADS server starts up successfully, it changes to initialized status.

Execute the `eztool status` command to verify that the initialization of all EADS servers is finished.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I          The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      Port   Position     Cluster  State        Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   initialized  none
 2  XX.XXX.XXX.168  24700    429496730  online   initialized  none
 3  XX.XXX.XXX.168  24800   -429496729  online   initialized  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   initialized  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   initialized  none
------------------------------------------------------------------------

KDEA08002-I          The command will now end.
$
```

If the initialization is finished, `initialized` is displayed in the `State` column.

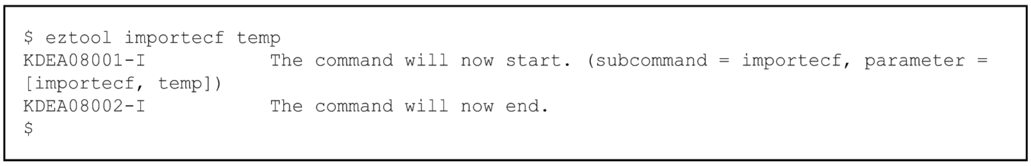
(7) Relocate data

Execute the `eztool importecf` command to relocate data. For *path-name-of-the-storage-for-cache-data-files-and-cache-index-files*, specify the directory to which cache data files and cache index files were moved in *11.1.3(2) Move cache files for all EADS servers*.

> **Important note**
>
> The following directories cannot be specified for *path-name-of-the-storage-for-cache-data-files-and-cache-index-files*:
>
> - *directory-specified-in-the-eads.cache.disk.n.dir-parameter-in-the-cache-properties*/*cache-name*
> - *directory-specified-in-the-eads.cache.disk.info.dir-parameter-in-the-cache-properties*/*cache-name* (the default is *management-directory*/`store`/*cache-name*)

```
eztool importecf path-name-of-the-storage-for-cache-data-files-and-cache-index-files
```

Command execution example



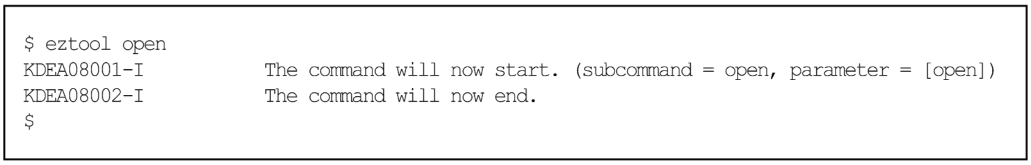
```
$ eztool importecf temp
KDEA08001-I          The command will now start. (subcommand = importecf, parameter =
[importecf, temp])
KDEA08002-I          The command will now end.
$
```

(8) Open the cluster

As is the case with closed status, in initialized status, the EADS servers do not accept requests from the EADS client. Execute the `eztool open` command to open the cluster.

```
eztool open
```

Command execution example



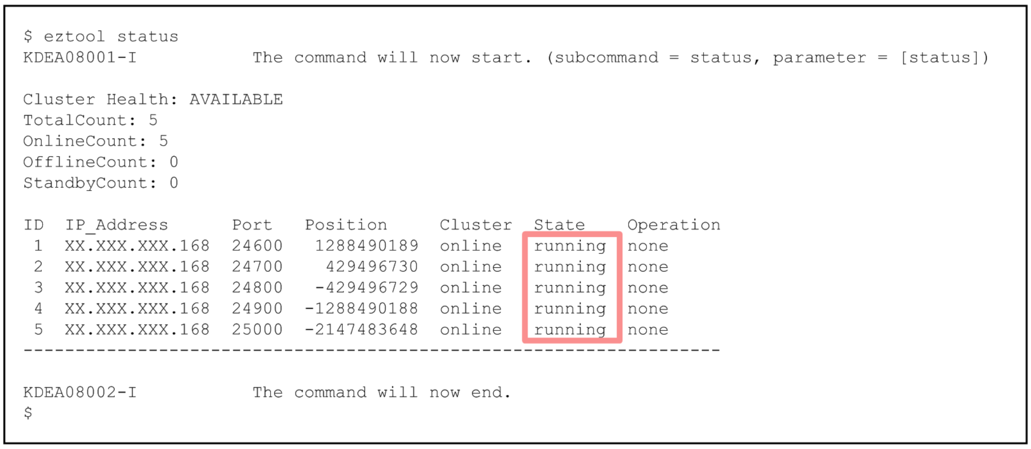
```
$ eztool open
KDEA08001-I          The command will now start. (subcommand = open, parameter = [open])
KDEA08002-I          The command will now end.
$
```

(9) Verify that the EADS servers have been opened

Execute the `eztool status` command to verify that all EADS servers have been opened.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address       Port   Position     Cluster  State    Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   running  none
 2  XX.XXX.XXX.168  24700    429496730  online   running  none
 3  XX.XXX.XXX.168  24800   -429496729  online   running  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   running  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   running  none
-------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

If the EADS servers have been opened, `running` is displayed in the `State` column.

11.2 Deleting EADS servers from a cluster

This section explains how to delete EADS servers from a cluster.

11.2.1 How to delete EADS servers from a cluster (using only memory caches)

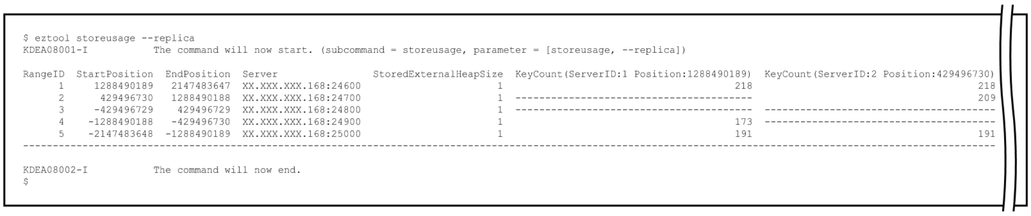
This subsection explains how to delete EADS servers from a cluster when only memory caches are used.

(1) Check the data distribution among the ranges

Execute the `eztool storeusage --replica` command to check the distribution of the data among the ranges.

```
eztool storeusage --replica
```

Command execution example



```
$ eztool storeusage --replica
KDEA08001-I           The command will now start. (subcommand = storeusage, parameter = [storeusage, --replica])

RangeID  StartPosition  EndPosition  Server                StoredExternalHeapSize  KeyCount(ServerID:1 Position:1288490189)  KeyCount(ServerID:2 Position:429496730)
      1     1288490189   2147483647  XX.XXX.XXX.168:24600                       1                                       218                                      218
      2      429496730   1288490188  XX.XXX.XXX.168:24700                       1  ----------------------------------------                                      209
      3     -429496729    429496729  XX.XXX.XXX.168:24800                       1  ----------------------------------------  ---------------------------------------
      4    -1288490188   -429496730  XX.XXX.XXX.168:24900                       1                                       173  ---------------------------------------
      5    -2147483648  -1288490189  XX.XXX.XXX.168:25000                       1                                       191                                      191
--------------------------------------------------------------------------------------------------------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

Based on the execution results of the `eztool storeusage --replica` command, determine the ranges from which EADS servers need to be deleted.

(2) Terminate all EADS servers in the cluster (after exporting data to files)

Export data to files and then terminate all EADS servers in the cluster.

For details about this procedure, see *10.5 Terminating the EADS servers (after exporting data from memory to files)*.

(3) Change the cluster properties

Because the number of EADS servers that make up the cluster is decreasing, you must change the cluster properties.

Delete from the cluster property files for all the EADS servers the following parameters for each EADS server that will be deleted:

- `eads.node.`*EADS-server-ID*`.address`
- `eads.node.`*EADS-server-ID*`.port`
- `eads.node.`*EADS-server-ID*`.position`

(4) Back up the store data files of the EADS servers to be deleted

Back up to any directory the store data files of the EADS servers you want to delete.

Copy the store data files of the EADS servers you want to be delete to a storage location containing the store data files of the EADS servers that you are not deleting.

(5) Start all EADS servers in the cluster (and importing data from files)

Start all EADS servers in the cluster, and then import back into memory the data that was exported to files during the previous session.

For details about the procedure, see *10.3 Starting the EADS servers (and creating caches by importing data from files)*.

11.2.2 How to delete EADS servers from a cluster (using disk caches)

This subsection is applicable when you will be using disk caches and two-way caches.

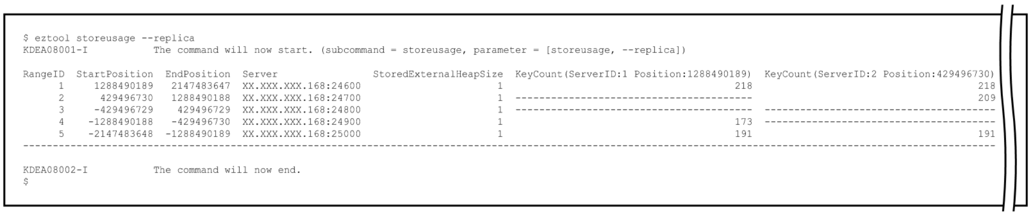
This subsection explains how to delete EADS servers from a cluster when disk caches or two-way caches are used.

(1) Check the data distribution among the ranges

Execute the `eztool storeusage --replica` command to check the distribution of the data among the ranges.

```
eztool storeusage --replica
```

Command execution example



```
$ eztool storeusage --replica
KDEA08001-I           The command will now start. (subcommand = storeusage, parameter = [storeusage, --replica])

RangeID  StartPosition  EndPosition  Server                StoredExternalHeapSize  KeyCount(ServerID:1 Position:1288490189)  KeyCount(ServerID:2 Position:429496730)
      1     1288490189   2147483647  XX.XXX.XXX.168:24600                       1                                       218                                      218
      2      429496730   1288490188  XX.XXX.XXX.168:24700                       1  ----------------------------------------                                      209
      3     -429496729    429496729  XX.XXX.XXX.168:24800                       1  ----------------------------------------  ---------------------------------------
      4    -1288490188   -429496730  XX.XXX.XXX.168:24900                       1                                       173  ---------------------------------------
      5    -2147483648  -1288490189  XX.XXX.XXX.168:25000                       1                                       191                                      191
-------------------------------------------------------------------------------------------------------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

Based on the execution results of the `eztool storeusage --replica` command, determine the ranges from which EADS servers need to be deleted.

(2) Terminate all EADS servers in the cluster

Terminate all EADS servers in the cluster.

For details about this procedure, see *10.6 Terminating the EADS servers (terminating caches on disk)*.

(3) Change the cluster properties

Because the number of EADS servers making up the cluster is to be decreased, you must change the cluster properties.

Delete from the cluster property files for all the EADS servers the following parameters for each EADS server that will be deleted:

- `eads.node.`*EADS-server-ID*`.address`
- `eads.node.`*EADS-server-ID*`.port`
- `eads.node.`*EADS-server-ID*`.position`

(4) Move the cache files of the EADS servers

Move to desired directories the cache files for all EADS servers except the EADS servers that are to be deleted.

For each server, move all files under the following directories (copy all files under the following directories and then delete the source files):

- *directory-specified-in-the-eads.cache.disk.n.dir-parameter-in-the-cache-properties*/*cache-name*
- *directory-specified-in-the-eads.cache.disk.info.dir-parameter-in-the-cache-properties*/*cache-name* (the default is *management-directory*/`store`/*cache-name*)

Make sure that the paths of the target directories are identical on all the EADS servers.

(5) Back up the cache files of the EADS servers to be deleted

Back up the cache files of the EADS servers to be deleted to the directories to which the cache files of one of the other EADS servers were moved in *11.2.2(4) Move the cache files of the EADS servers*.

Move all files under the following directories of the EADS servers to be deleted:

- *directory-specified-in-the-eads.cache.disk.n.dir-parameter-in-the-cache-properties*/*cache-name*
- *directory-specified-in-the-eads.cache.disk.info.dir-parameter-in-the-cache-properties*/*cache-name* (the default is *management-directory*/`store`/*cache-name*)

(6) Start all EADS servers in the cluster (relocate data)

Relocate data after all EADS servers in the cluster have started.

For details about this procedure, see *11.1.3(5) Start all EADS servers in the cluster* through *11.1.3(9) Verify that the EADS servers have been opened*.

11.3 Making the number of keys uniform in all ranges

If the number of keys is not uniform among all ranges for a reason such as scale-out processing, you can make the number of keys uniform by rearranging the cluster's EADS servers evenly on the consistent hashing circumference.

11.3.1 How to make the number of keys uniform in all ranges

This subsection explains how to arrange the EADS server positions (hash values) evenly to make the number of keys uniform in all ranges.

(1) Terminate all EADS servers in the cluster (after exporting data to files)

Export data to files and then terminate all EADS servers in the cluster.

For details about this procedure, see *10.5 Terminating the EADS servers (after exporting data from memory to files)*.

(2) Change the cluster properties

To adjust the EADS server positions (hash values) automatically, change each EADS server's cluster properties.

Delete from each EADS server's cluster property file all the `eads.node.`*EADS-server-ID*`.position` parameters.

(3) Start the EADS servers (import data from files)

Start all EADS servers in the cluster, and then import back into memory the data that was exported to files during the previous session.

For details about the procedure, see *10.3 Starting the EADS servers (and creating caches by importing data from files)*.

11.4 Changing the properties

This section explains how to change the properties.

11.4.1 How to change the properties

This subsection explains how to change the properties.

(1) Terminate the EADS servers (after exporting data to files)

Export data to files and then terminate the EADS servers.

For details about this procedure, see *10.5 Terminating the EADS servers (after exporting data from memory to files)* or *10.6 Terminating the EADS servers (terminating caches on disk)*.

> **Important note**
>
> There is no need to terminate the EADS servers when you are changing command properties.

(2) Change the properties

Change the properties.

If you change a server property parameter whose name begins with `eads.logger.`, either move all files and directories under the directory specified in the `eads.logger.dir` server property parameter to another directory or delete them.

Whenever you change cluster properties or shared properties, make sure all properties are the same on all EADS servers that make up the cluster.

Whenever you change cache properties, make sure the settings for the following parameters are the same on all EADS servers that make up the cluster:

- `eads.cache.type`
- `eads.cache.disk.filesize`
- `eads.cache.disk.filenum`
- `eads.cache.disk.blocksize`

If you change a command property parameter whose name begins with `eads.command.logger.`, either move all files and directories under the directory specified in the `eads.command.logger.dir` command property parameter to another directory or delete them.

(3) Start the EADS servers (and import data from files)

Start the EADS servers, and then import back into memory the data that was exported to files during the previous session. For details about the procedure, see *10.3 Starting the EADS servers (and creating caches by importing data from files)*.

11.4.2 Notes about changing properties

This subsection is applicable when you will be using disk caches or two-way caches.

If the parameters listed below are changed while you are using disk caches or two-way caches, the caches can no longer be resumed. If you want to change these parameters, first relocate the data. For details about relocating data, see *11.1.3 How to add EADS servers to a cluster after stopping the cluster (using only disk caches)*.

- Cluster property parameter
  - `eads.node.`*EADS-server-ID*`.position` parameter
- Shared property parameters
  - `eads.replication.factor` parameter
  - `eads.cache.key.maxsize` parameter

Do not change the parameters listed below while you are using disk caches or two-way caches. If this rule is violated, the caches can no longer be resumed.

- Cache property parameters
  - `eads.cache.type` parameter
  - `eads.cache.disk.filesize` parameter

  Note: The value of the `eads.cache.disk.filenum` parameter in the cache properties can be increased, but it cannot be decreased.

11.5 Adding and deleting caches

This section explains how to add and delete caches.

11.5.1 How to add and delete memory caches

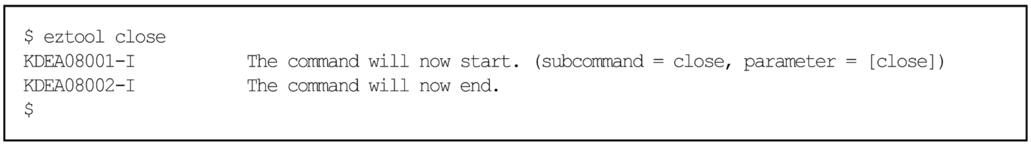
This subsection explains how to add and delete memory caches.

(1) Close the cluster

Execute the `eztool close` command to close the cluster.

```
eztool close
```

Command execution example



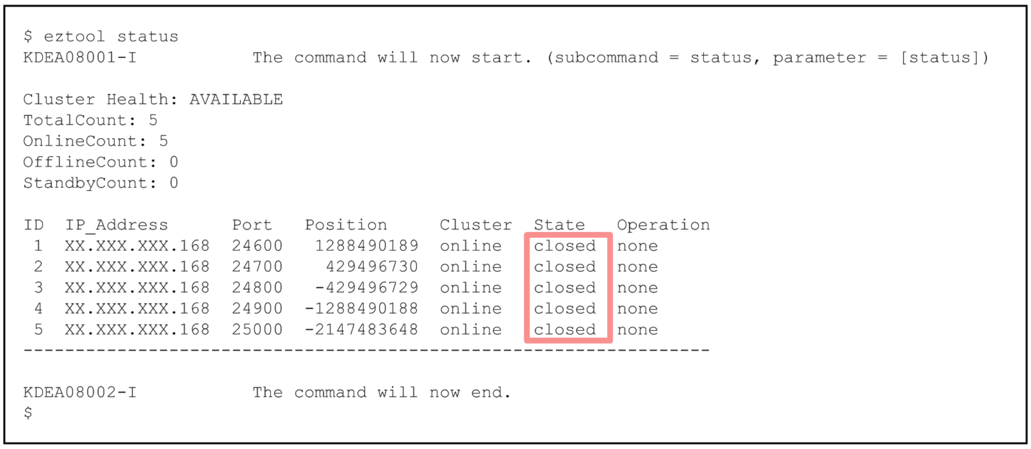
```
$ eztool close
KDEA08001-I          The command will now start. (subcommand = close, parameter = [close])
KDEA08002-I          The command will now end.
$
```

(2) Verify that the cluster is closed

After you have closed the EADS servers, execute the `eztool status` command to check the status of the cluster.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I          The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      Port   Position     Cluster  State   Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   closed  none
 2  XX.XXX.XXX.168  24700    429496730  online   closed  none
 3  XX.XXX.XXX.168  24800   -429496729  online   closed  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   closed  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   closed  none
------------------------------------------------------------------

KDEA08002-I          The command will now end.
$
```

If the cluster is closed, `closed` is displayed in the `State` column.

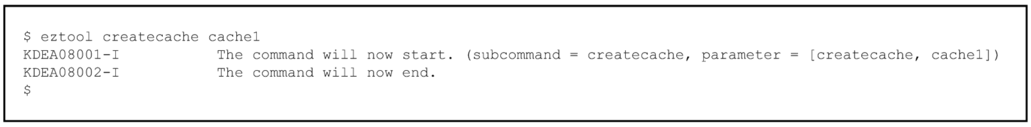
(3) Add or delete caches

(a) Adding caches

Execute the `eztool createcache` command to add a cache.

```
eztool createcache cache-name
```

Command execution example



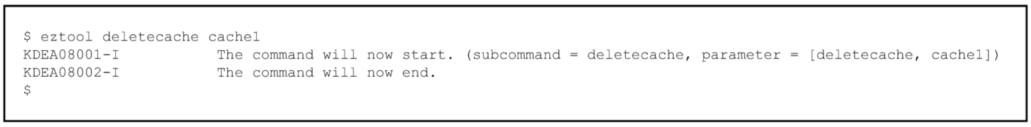
```
$ eztool createcache cache1
KDEA08001-I           The command will now start. (subcommand = createcache, parameter = [createcache, cache1])
KDEA08002-I           The command will now end.
$
```

(b) Deleting caches

Execute the `eztool deletecache` command to delete a cache.

```
eztool deletecache cache-name
```

Command execution example



```
$ eztool deletecache cache1
KDEA08001-I           The command will now start. (subcommand = deletecache, parameter = [deletecache, cache1])
KDEA08002-I           The command will now end.
$
```

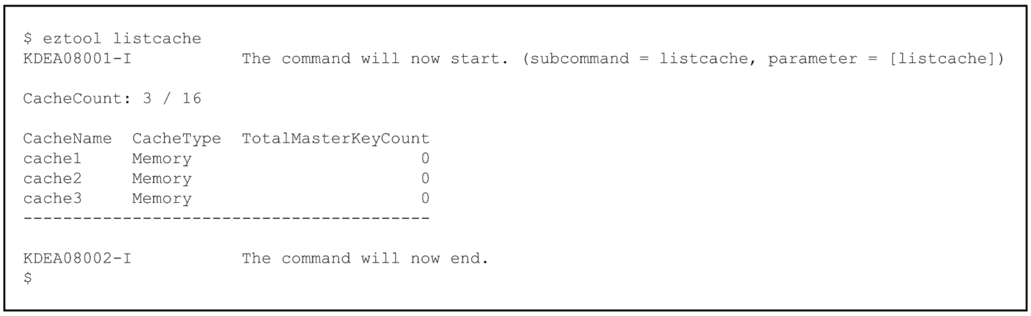
(4) Verify that the caches have been added or deleted

Execute the `eztool listcache` command to display a list of caches.

Verify that the caches have been added or deleted.

```
eztool listcache
```

Command execution example



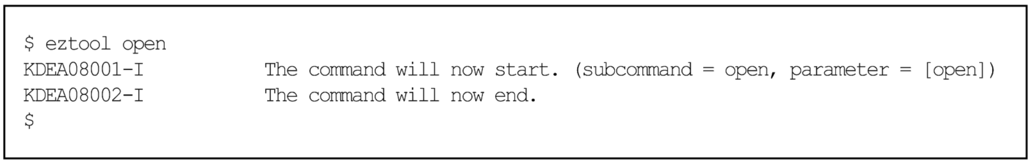
```
$ eztool listcache
KDEA08001-I           The command will now start. (subcommand = listcache, parameter = [listcache])

CacheCount: 3 / 16

CacheName  CacheType  TotalMasterKeyCount
cache1     Memory                       0
cache2     Memory                       0
cache3     Memory                       0
-----------------------------------------

KDEA08002-I           The command will now end.
$
```

(5) Open the cluster

Execute the `eztool open` command to open the cluster.

```
eztool open
```

Command execution example



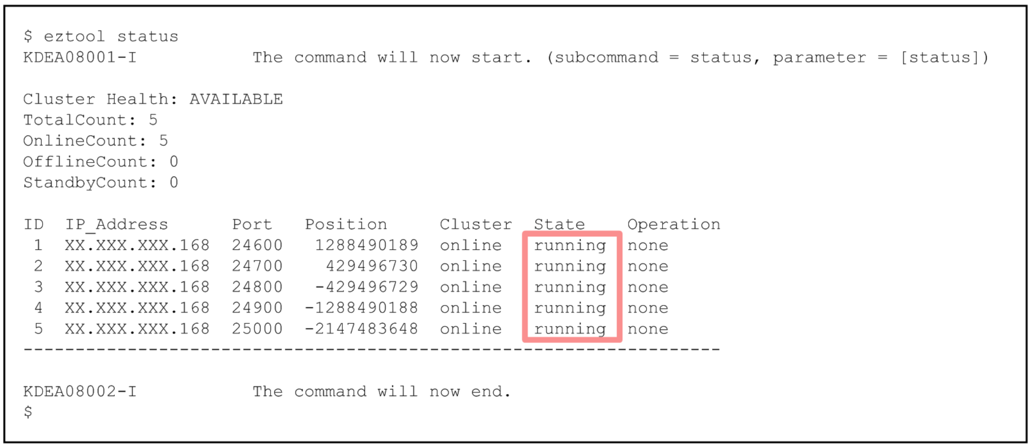
```
$ eztool open
KDEA08001-I          The command will now start. (subcommand = open, parameter = [open])
KDEA08002-I          The command will now end.
$
```

(6) Verify that the EADS servers have been opened

Execute the `eztool status` command to verify that all EADS servers have been opened.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I          The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      Port   Position     Cluster  State    Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   running  none
 2  XX.XXX.XXX.168  24700    429496730  online   running  none
 3  XX.XXX.XXX.168  24800   -429496729  online   running  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   running  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   running  none
--------------------------------------------------------------------

KDEA08002-I          The command will now end.
$
```

If the EADS servers have been opened, `running` is displayed in the `State` column.

11.5.2 How to add and delete disk caches and two-way caches

This subsection is applicable when you will be using disk caches and two-way caches.

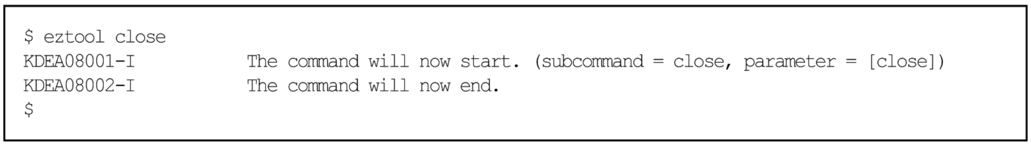
This subsection explains how to add and delete disk caches and two-way caches.

(1) Close the cluster

Execute the `eztool close` command to close the cluster.

```
eztool close
```

Command execution example



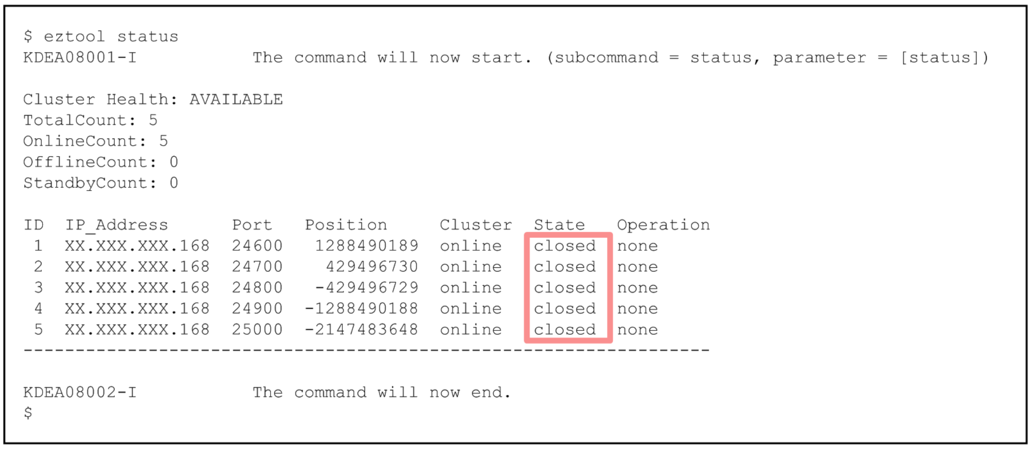
```
$ eztool close
KDEA08001-I          The command will now start. (subcommand = close, parameter = [close])
KDEA08002-I          The command will now end.
$
```

(2) Verify that the cluster is closed

After you have closed the EADS servers, execute the `eztool status` command to check the status of the cluster.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I          The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      Port   Position     Cluster  State   Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   closed  none
 2  XX.XXX.XXX.168  24700    429496730  online   closed  none
 3  XX.XXX.XXX.168  24800   -429496729  online   closed  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   closed  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   closed  none
------------------------------------------------------------------

KDEA08002-I          The command will now end.
$
```

If the cluster is closed, `closed` is displayed in the `State` column.

(3) Add or delete caches

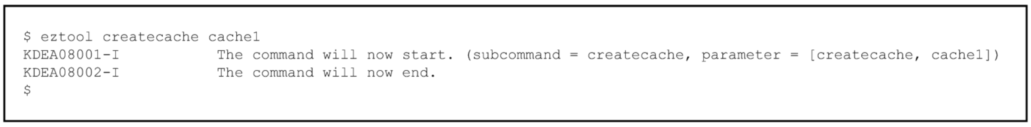
(a) Adding caches

Create a cache property file for a cache that is to be added. For details about this procedure, see *10.2.2(1) Create cache property files*.

Next, execute the `eztool createcache` command to add the cache.

```
eztool createcache cache-name
```

Command execution example



```
$ eztool createcache cache1
KDEA08001-I          The command will now start. (subcommand = createcache, parameter = [createcache, cache1])
KDEA08002-I          The command will now end.
$
```

When a cache is created, cache files are created in the following directories:

- Cache data files
  directory-specified-in-the-eads.cache.disk.n.dir-parameter-in-the-cache-properties / *cache-name*
- Cache index files and cache information files

directory-specified-in-the-eads.cache.disk.info.dir-parameter-in-the-cache-properties/*cache-name* (the default is *management-directory*/store/*cache-name*)

(b) Deleting caches

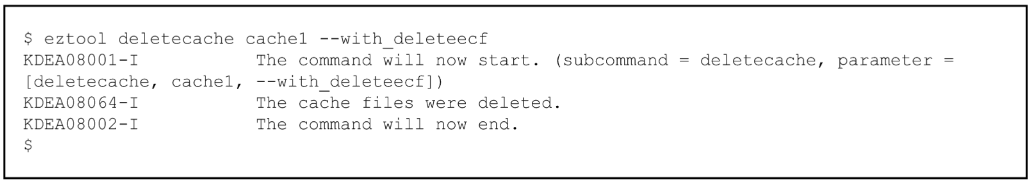
The following methods are provided for deleting a cache and its cache files:

- Execute the `eztool deletecache --with_deleteecf` command.
- Execute the `eztool deletecache` command and then execute the `eztool deleteecf` command.

Executing the eztool deletecache --with_deleteecf command

```
eztool deletecache cache-name --with_deleteecf
```

Command execution example

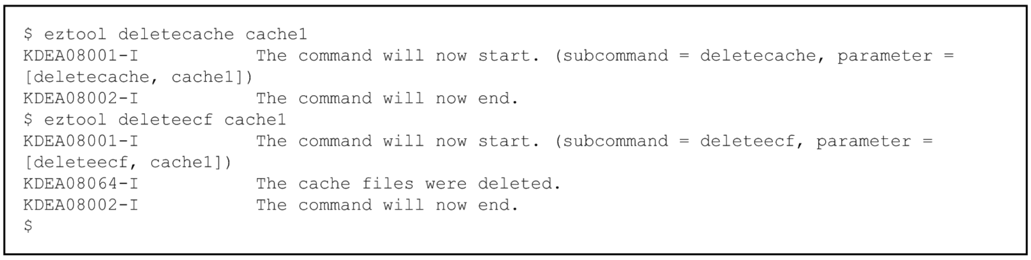


```
$ eztool deletecache cache1 --with_deleteecf
KDEA08001-I           The command will now start. (subcommand = deletecache, parameter =
[deletecache, cache1, --with_deleteecf])
KDEA08064-I           The cache files were deleted.
KDEA08002-I           The command will now end.
$
```

Executing the eztool deletecache command and then the eztool deleteecf command

```
eztool deletecache cache-name

eztool deleteecf cache-name
```

Command execution example



```
$ eztool deletecache cache1
KDEA08001-I           The command will now start. (subcommand = deletecache, parameter =
[deletecache, cache1])
KDEA08002-I           The command will now end.
$ eztool deleteecf cache1
KDEA08001-I           The command will now start. (subcommand = deleteecf, parameter =
[deleteecf, cache1])
KDEA08064-I           The cache files were deleted.
KDEA08002-I           The command will now end.
$
```

(4) Verify that the caches have been added or deleted

If you have added caches, execute the `eztool listcache` command to display a list of caches. Verify that the caches have been added.

```
eztool listcache
```

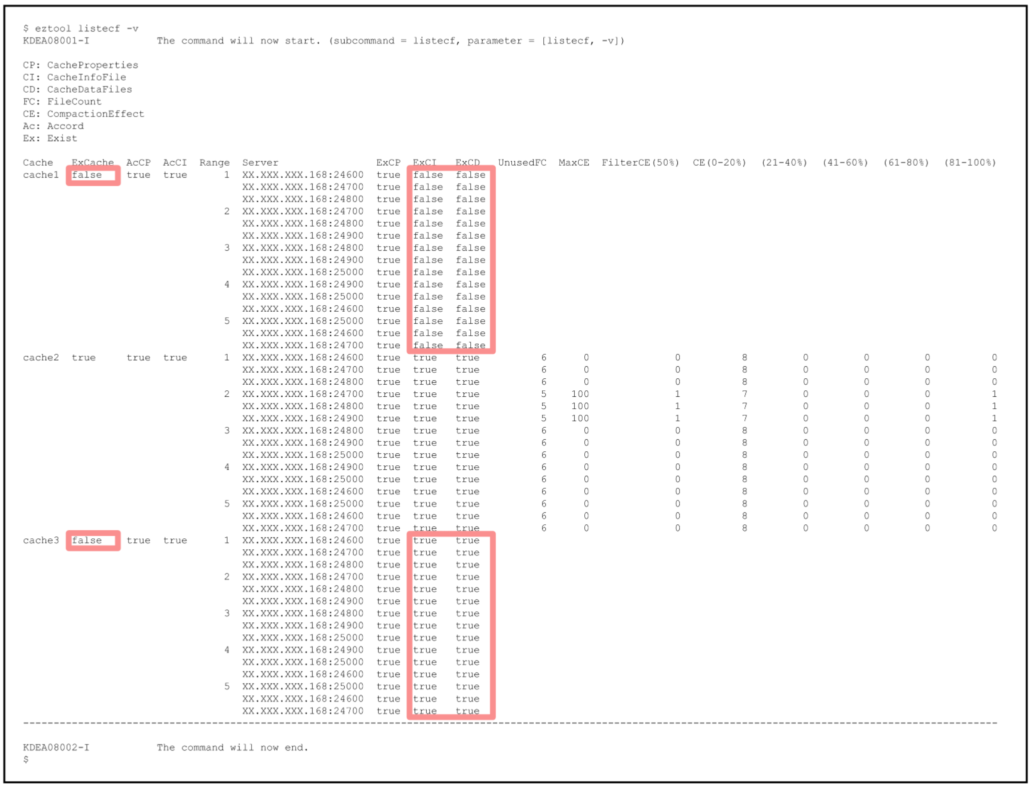
Command execution example

```
$ eztool listcache
KDEA08001-I           The command will now start. (subcommand = listcache,
parameter = [listcache])

CacheCount: 3 / 16

CacheName  CacheType  TotalMasterKeyCount
cache1     Disk                         0
cache2     Disk                         0
cache3     Disk                         0
-----------------------------------------

KDEA08002-I           The command will now end.
$
```

If you have deleted caches, execute the eztool listecf -v command to display information about persistent data.

```
eztool listecf -v
```

Command execution example



```
$ eztool listecf -v
KDEA08001-I           The command will now start. (subcommand = listecf, parameter = [listecf, -v])

CP: CacheProperties
CI: CacheInfoFile
CD: CacheDataFiles
FC: FileCount
CE: CompactionEffect
Ac: Accord
Ex: Exist

Cache   ExCache  AcCP  AcCI  Range  Server                ExCP  ExCI   ExCD   UnusedFC  MaxCE  FilterCE(50%)  CE(0-20%)  (21-40%)  (41-60%)  (61-80%)  (81-100%)
cache1  false    true  true      1  XX.XXX.XXX.168:24600  true  false  false
                                    XX.XXX.XXX.168:24700  true  false  false
                                    XX.XXX.XXX.168:24800  true  false  false
                                 2  XX.XXX.XXX.168:24700  true  false  false
                                    XX.XXX.XXX.168:24800  true  false  false
                                    XX.XXX.XXX.168:24900  true  false  false
                                 3  XX.XXX.XXX.168:24800  true  false  false
                                    XX.XXX.XXX.168:24900  true  false  false
                                    XX.XXX.XXX.168:25000  true  false  false
                                 4  XX.XXX.XXX.168:24900  true  false  false
                                    XX.XXX.XXX.168:25000  true  false  false
                                    XX.XXX.XXX.168:24600  true  false  false
                                 5  XX.XXX.XXX.168:25000  true  false  false
                                    XX.XXX.XXX.168:24600  true  false  false
                                    XX.XXX.XXX.168:24700  true  false  false
cache2  true     true  true      1  XX.XXX.XXX.168:24600  true  true   true          6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24700  true  true   true          6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24800  true  true   true          6      0              0          8         0         0         0          0
                                 2  XX.XXX.XXX.168:24700  true  true   true          5    100              1          7         0         0         0          1
                                    XX.XXX.XXX.168:24800  true  true   true          5    100              1          7         0         0         0          1
                                    XX.XXX.XXX.168:24900  true  true   true          5    100              1          7         0         0         0          1
                                 3  XX.XXX.XXX.168:24800  true  true   true          6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24900  true  true   true          6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:25000  true  true   true          6      0              0          8         0         0         0          0
                                 4  XX.XXX.XXX.168:24900  true  true   true          6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:25000  true  true   true          6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24600  true  true   true          6      0              0          8         0         0         0          0
                                 5  XX.XXX.XXX.168:25000  true  true   true          6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24600  true  true   true          6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24700  true  true   true          6      0              0          8         0         0         0          0
cache3  false    true  true      1  XX.XXX.XXX.168:24600  true  true   true
                                    XX.XXX.XXX.168:24700  true  true   true
                                    XX.XXX.XXX.168:24800  true  true   true
                                 2  XX.XXX.XXX.168:24700  true  true   true
                                    XX.XXX.XXX.168:24800  true  true   true
                                    XX.XXX.XXX.168:24900  true  true   true
                                 3  XX.XXX.XXX.168:24800  true  true   true
                                    XX.XXX.XXX.168:24900  true  true   true
                                    XX.XXX.XXX.168:25000  true  true   true
                                 4  XX.XXX.XXX.168:24900  true  true   true
                                    XX.XXX.XXX.168:25000  true  true   true
                                    XX.XXX.XXX.168:24600  true  true   true
                                 5  XX.XXX.XXX.168:25000  true  true   true
                                    XX.XXX.XXX.168:24600  true  true   true
                                    XX.XXX.XXX.168:24700  true  true   true
-----------------------------------------------------------------------------------------------------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

- Executing the eztool deletecache --with_deleteecf or eztool deletecache command and then executing the eztool deleteecf command

  If caches and cache files have been deleted successfully, false is displayed in the following columns:

  - ExCache column
  - ExCI column
  - ExCD column

  In the figure, cache1's cache and cache files have been deleted successfully.
- Executing only the eztool deletecache command

If caches have been deleted successfully, false is displayed in the ExCache column and true is displayed in the ExCI and ExCD columns. No values are displayed in the columns starting with UnusedFC.

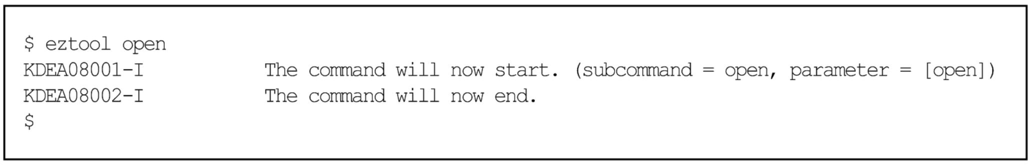
In the figure, cache3's cache has been deleted successfully.

(5) Open the cluster

Execute the eztool open command to open the cluster.

```
eztool open
```

Command execution example



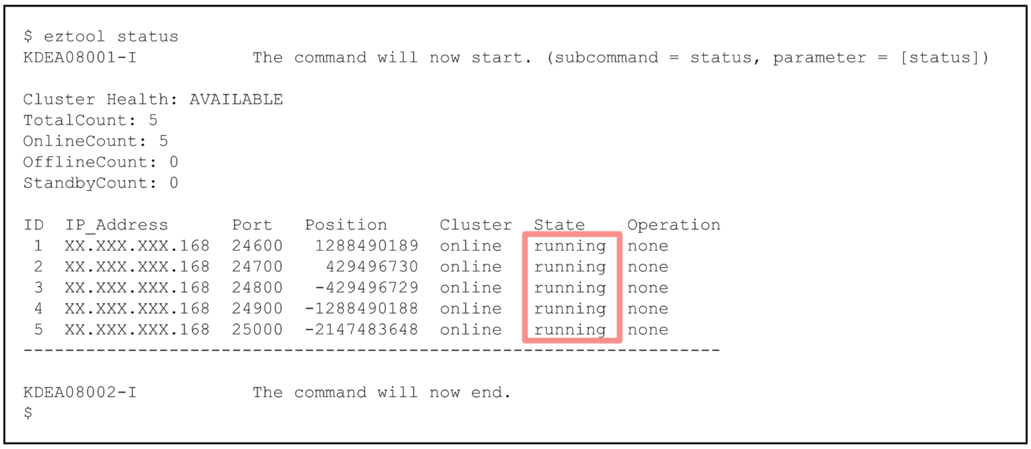
```
$ eztool open
KDEA08001-I          The command will now start. (subcommand = open, parameter = [open])
KDEA08002-I          The command will now end.
$
```

(6) Verify that the EADS servers have been opened

Execute the eztool status command to verify that all EADS servers have been opened.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      Port   Position     Cluster  State    Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   running  none
 2  XX.XXX.XXX.168  24700    429496730  online   running  none
 3  XX.XXX.XXX.168  24800   -429496729  online   running  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   running  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   running  none
-------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

If the EADS servers have been opened, running is displayed in the State column.

11.6 Making a backup

This section explains how to back up data.

11.6.1 How to back up data

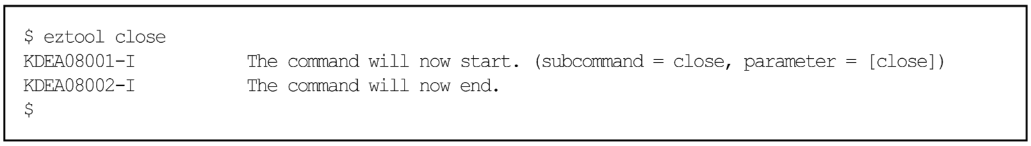
This subsection explains how to back up data.

(1) Close the cluster

Execute the `eztool close` command to close the cluster.

```
eztool close
```

Command execution example



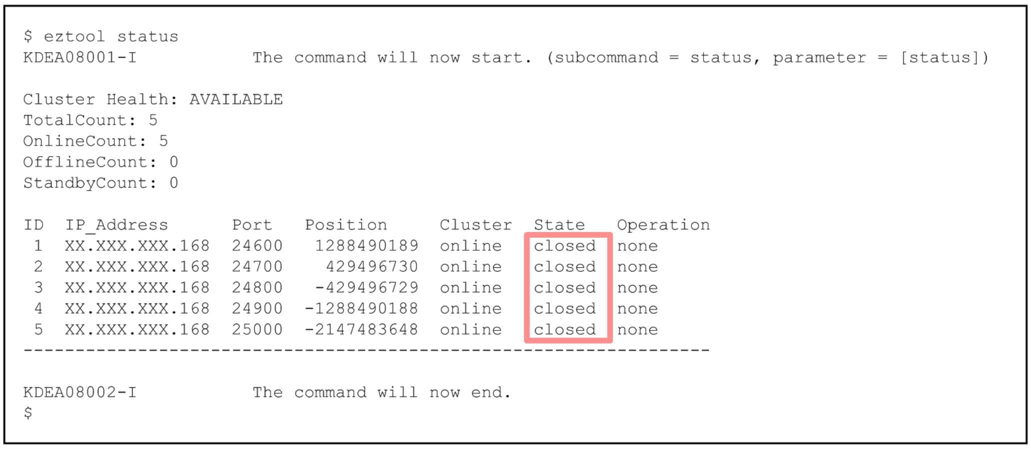
```
$ eztool close
KDEA08001-I          The command will now start. (subcommand = close, parameter = [close])
KDEA08002-I          The command will now end.
$
```

(2) Verify that the cluster is closed

After you have closed the EADS servers, execute the `eztool status` command to check the status of the cluster.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      Port   Position     Cluster  State   Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   closed  none
 2  XX.XXX.XXX.168  24700    429496730  online   closed  none
 3  XX.XXX.XXX.168  24800   -429496729  online   closed  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   closed  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   closed  none
------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

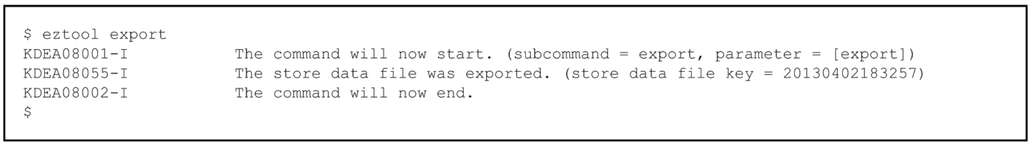
If the cluster is closed, `closed` is displayed in the `State` column.

(3) Export data to files

Execute the `eztool export` command to export data to a store data file.

```
eztool export
```

Command execution example



```
$ eztool export
KDEA08001-I          The command will now start. (subcommand = export, parameter = [export])
KDEA08055-I          The store data file was exported. (store data file key = 20130402183257)
KDEA08002-I          The command will now end.
$
```

As shown in this example, the command execution date and time become the store data file key when no store data file key is specified.

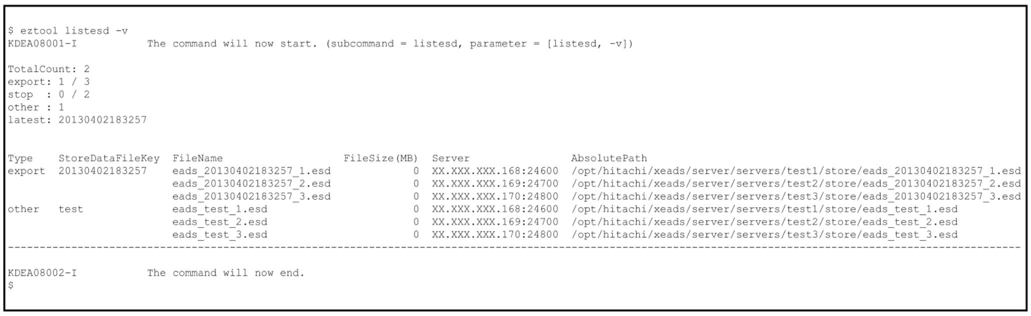
You can specify any store data file key.

(4) Back up the store data files

Execute the `eztool listesd -v` command to check the storage location for the acquired store data file.

```
eztool listesd -v
```

Command execution example



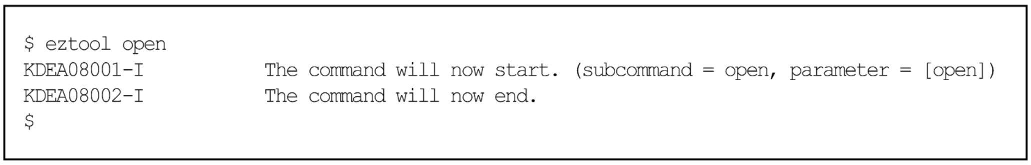
```
$ eztool listesd -v
KDEA08001-I          The command will now start. (subcommand = listesd, parameter = [listesd, -v])

TotalCount: 2
export: 1 / 3
stop  : 0 / 2
other : 1
latest: 20130402183257


Type    StoreDataFileKey  FileName                        FileSize(MB)  Server                  AbsolutePath
export  20130402183257    eads_20130402183257_1.esd                  0  XX.XXX.XXX.168:24600  /opt/hitachi/xeads/server/servers/test1/store/eads_20130402183257_1.esd
                          eads_20130402183257_2.esd                  0  XX.XXX.XXX.169:24700  /opt/hitachi/xeads/server/servers/test2/store/eads_20130402183257_2.esd
                          eads_20130402183257_3.esd                  0  XX.XXX.XXX.170:24800  /opt/hitachi/xeads/server/servers/test3/store/eads_20130402183257_3.esd
other   test              eads_test_1.esd                            0  XX.XXX.XXX.168:24600  /opt/hitachi/xeads/server/servers/test1/store/eads_test_1.esd
                          eads_test_2.esd                            0  XX.XXX.XXX.169:24700  /opt/hitachi/xeads/server/servers/test2/store/eads_test_2.esd
                          eads_test_3.esd                            0  XX.XXX.XXX.170:24800  /opt/hitachi/xeads/server/servers/test3/store/eads_test_3.esd
--------------------------------------------------------------------------------------------------------------------------------------------------------------------

KDEA08002-I          The command will now end.
$
```

(5) Open the cluster

Execute the `eztool open` command to open the cluster.

```
eztool open
```

Command execution example



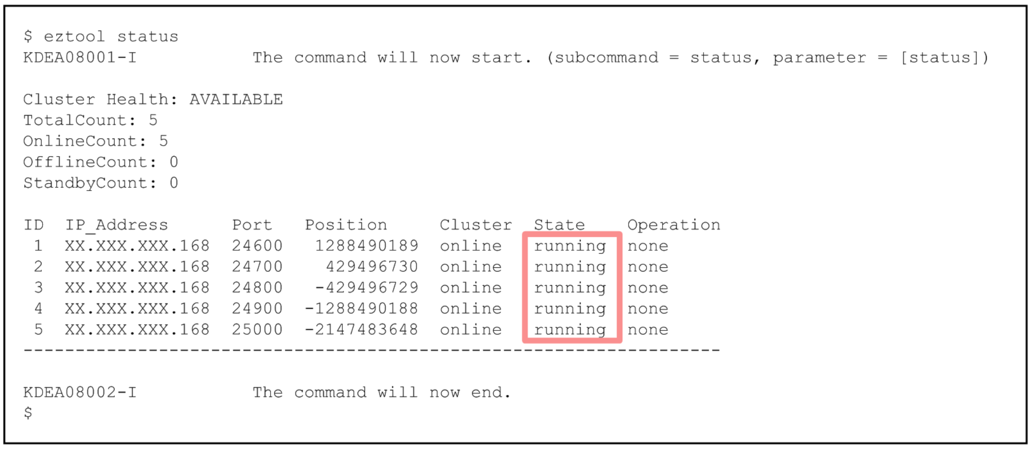
```
$ eztool open
KDEA08001-I          The command will now start. (subcommand = open, parameter = [open])
KDEA08002-I          The command will now end.
$
```

(6) Verify that the EADS servers have been opened

Execute the `eztool status` command to verify that all EADS servers have been opened.

```
eztool status
```

Command execution example



```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      Port   Position     Cluster  State    Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   running  none
 2  XX.XXX.XXX.168  24700    429496730  online   running  none
 3  XX.XXX.XXX.168  24800   -429496729  online   running  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   running  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   running  none
-------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

If the EADS servers have been opened, `running` is displayed in the `State` column.

11.7 Managing store data files

Because there is a limit to the number of store data file generations, you must delete store data files before the maximum number of generations is reached.

For details about the number of store data file generations, see *7.6.2 Specifying the number of store data file generations*.

This section explains how to check the store data and delete the store data files.

11.7.1 How to check and delete store data files

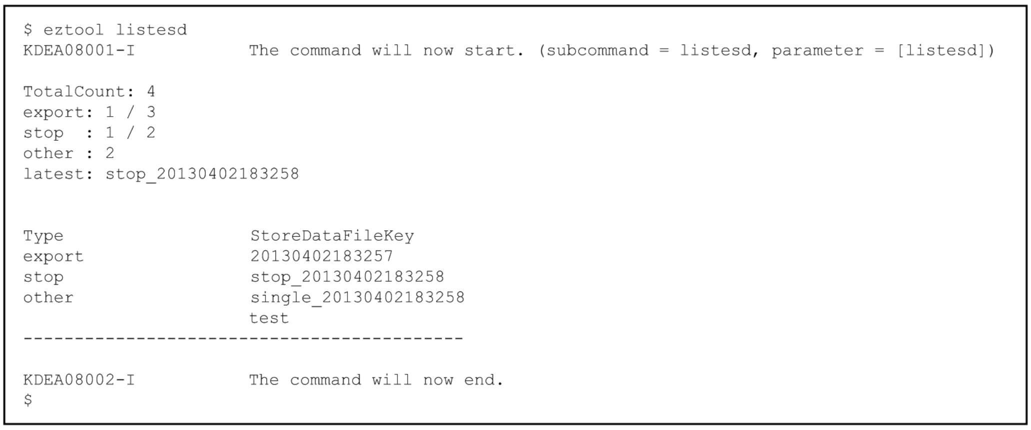
This subsection explains how to check and delete store data files.

(1) Display a list of store data files

Execute the `eztool listesd` command to display a list of the store data files in the cluster.

```
eztool listesd
```

Command execution example



```
$ eztool listesd
KDEA08001-I           The command will now start. (subcommand = listesd, parameter = [listesd])

TotalCount: 4
export: 1 / 3
stop  : 1 / 2
other : 2
latest: stop_20130402183258


Type                  StoreDataFileKey
export                20130402183257
stop                  stop_20130402183258
other                 single_20130402183258
                      test
-------------------------------------------

KDEA08002-I           The command will now end.
$
```

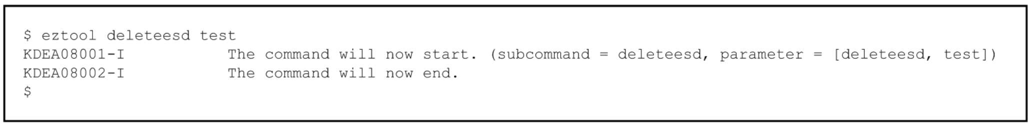
The `StoreDataFileKey` column displays a list of the store data file keys of the store data files.

(2) Delete unneeded store data files

Execute the `eztool deleteesd` command to delete an unneeded store data file from the cluster.

```
eztool deleteesd store-data-file-key
```

Command execution example



```
$ eztool deleteesd test
KDEA08001-I           The command will now start. (subcommand = deleteesd, parameter = [deleteesd, test])
KDEA08002-I           The command will now end.
$
```

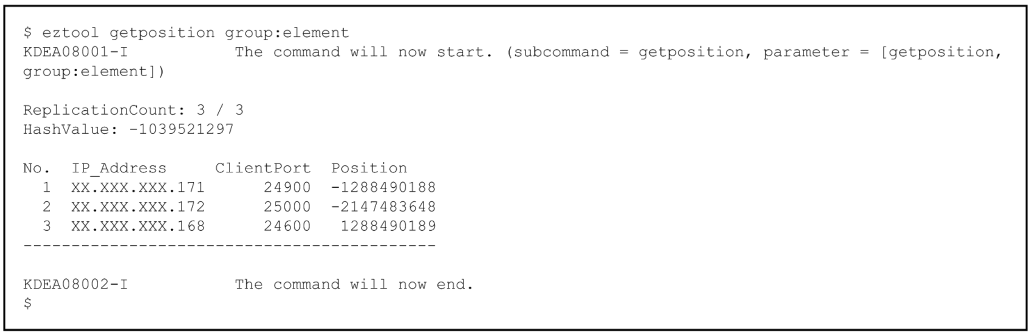
11.8 Checking the data storage location

Execute the `eztool getposition` command to display the EADS server that stores the specified key and group.

If you specify the `-l` or `--local` option, you can execute the command regardless of the status of the EADS server. In this case, the command imports the cluster property file of the EADS server on which the command was executed, not the cluster information currently in use.

```
eztool getposition key
```

Command execution example



```
$ eztool getposition group:element
KDEA08001-I           The command will now start. (subcommand = getposition, parameter = [getposition,
group:element])

ReplicationCount: 3 / 3
HashValue: -1039521297

No.  IP_Address     ClientPort  Position
  1  XX.XXX.XXX.171      24900  -1288490188
  2  XX.XXX.XXX.172      25000  -2147483648
  3  XX.XXX.XXX.168      24600   1288490189
-------------------------------------------

KDEA08002-I           The command will now end.
$
```

If redundant copies of data have been created, the EADS servers to which data has been copied are displayed following `No. 2` in the example above.

11.9 Checking a list of group names

Execute the `eztool listgroup` command to display a list of group names in the highest hierarchy in a specified cache.

```
eztool listgroup [-v|--verbose] cache-name
```

Command execution example

```
$ eztool listgroup cache1
KDEA08001-I          The command will now start. (subcommand = listgroup, parameter =
[listgroup, cache1])

GroupCount: 20

Server               Position     ID  GroupName
XX.XXX.XXX.168:24600   1288490189   1  group02
                                       group04
XX.XXX.XXX.168:24700    429496730   2  group01
                                       group09
                                       group11
                                       group17
                                       group18
                                       group19
XX.XXX.XXX.168:24800   -429496729   3  group05
                                       group10
                                       group12
                                       group13
                                       group15
XX.XXX.XXX.168:24900  -1288490188   4  group06
                                       group07
                                       group08
                                       group14
XX.XXX.XXX.168:25000  -2147483648   5  group03
                                       group16
                                       group20
------------------------------------------------

KDEA08002-I          The command will now end.
$
```

The `GroupName` column displays the group hierarchy names in the highest hierarchy for each EADS server.

If you specify the `-v` or `--verbose` option, you can check the locations of the groups in the highest hierarchy and the number of keys in each group.

```
$ eztool listgroup -v cache1
KDEA08001-I           The command will now start. (subcommand = listgroup, parameter =
[listgroup, -v, cache1])

GroupCount: 20

Server                Position     ID  GroupName  HashValue    KeyCount
XX.XXX.XXX.168:24600   1288490189   1  group02     2037266030       100
                                       group04     1388111144       100
XX.XXX.XXX.168:24700    429496730   2  group09     1269679203       100
                                       group11     1229725181       100
                                       group17     1172163628       100
                                       group19     1006380133       100
                                       group18      703078836       100
                                       group01      648506385       100
XX.XXX.XXX.168:24800   -429496729   3  group10      260938713       100
                                       group15      232103928       100
                                       group12      -56466136       100
                                       group05      -73411842       100
                                       group13     -163284496       100
XX.XXX.XXX.168:24900  -1288490188   4  group14     -503630070       100
                                       group07     -699511542       100
                                       group08     -992621419       100
                                       group06    -1203276328       100
XX.XXX.XXX.168:25000  -2147483648   5  group16    -1445454454       100
                                       group20    -1917239959       100
                                       group03    -1965919439       100
-----------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

11.10 Checking a list of keys

Execute the `eztool listkey` command to display a list of keys in a specified cache.

```
eztool listkey [-g group-name|--group group-name] cache-name
```

Command execution example

```
$ eztool listkey cache1
KDEA08001-I           The command will now start. (subcommand = listkey, parameter = [listkey,
cache1])

KeyCount: 25

Server                Position     ID  Key
XX.XXX.XXX.168:24600   1288490189   1  [1]groupA:element01
                                       [1]groupA:element02
                                       [1]groupA:element03
                                       [1]groupA:element04
                                       [1]groupA:element05
XX.XXX.XXX.168:24700    429496730   2  [2]groupB:element01
                                       [2]groupB:element02
                                       [2]groupB:element03
                                       [2]groupB:element04
                                       [2]groupB:element05
XX.XXX.XXX.168:24800   -429496729   3  [3]groupC:element01
                                       [3]groupC:element02
                                       [3]groupC:element03
                                       [3]groupC:element04
                                       [3]groupC:element05
XX.XXX.XXX.168:24900  -1288490188   4  [4]groupD:element01
                                       [4]groupD:element02
                                       [4]groupD:element03
                                       [4]groupD:element04
                                       [4]groupD:element05
XX.XXX.XXX.168:25000  -2147483648   5  [5]groupE:element01
                                       [5]groupE:element02
                                       [5]groupE:element03
                                       [5]groupE:element04
                                       [5]groupE:element05
----------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

The `Key` column displays the keys for each EADS server.

If you specify the `-g` or `--group` option, you can display a list of keys that belong to the specified group.

```
$ eztool listkey -g [1]groupA cache1
KDEA08001-I           The command will now start. (subcommand = listkey, parameter = [listkey, -g,
[1]groupA, cache1])

KeyCount: 5

Server                Position    ID  Key
XX.XXX.XXX.168:24600  1288490189   1  [1]groupA:element01
                                      [1]groupA:element02
                                      [1]groupA:element03
                                      [1]groupA:element04
                                      [1]groupA:element05
---------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

11.11 Checking whether user functions have been placed correctly on individual EADS servers and whether they can be executed

Execute the `eztool listfunc` command to display whether user functions have been placed correctly on the individual EADS servers and whether they can be executed.

```
eztool listfunc [-v|--verbose]
```

Command execution example

```
$ eztool listfunc
KDEA08001-I           The command will now start. (subcommand = listfunc, parameter = [listfunc])

FunctionCount: 3

FunctionName  Enable  Disable
FunctionA          3        0
FunctionB          2        1
FunctionC          0        1
-----------------------------
KDEA08002-I           The command will now end.
$
```

The `Enable` column displays the number of EADS servers on which each user function can be executed. If you specify the `-v` or `--verbose` option, you can check the user functions placed on each EADS server. You can also check whether they are executable.

```
$ eztool listfunc -v
KDEA08001-I           The command will now start. (subcommand = listfunc, parameter = [listfunc, -v])

FunctionCount: 3

Server                FunctionName  Status
-------------------------------------------
XX.XXX.XXX.168:24600  FunctionA     enable
XX.XXX.XXX.168:24600  FunctionB     enable
XX.XXX.XXX.168:24600  FunctionC     none

XX.XXX.XXX.169:24700  FunctionA     enable
XX.XXX.XXX.169:24700  FunctionB     enable
XX.XXX.XXX.169:24700  FunctionC     none

XX.XXX.XXX.170:24800  FunctionA     enable
XX.XXX.XXX.170:24800  FunctionB     disable
XX.XXX.XXX.170:24800  FunctionC     disable
-------------------------------------------
KDEA08002-I           The command will now end.
$
```

To make a user function for which `disable` is displayed executable, terminate all EADS servers, determine the cause of the problem from the message logs, and then eliminate the problem.

11.12 Applying EADS server patches while the cluster is running

This section explains how to apply EADS server patches while the cluster is running.

Normally, we recommend that you terminate all EADS servers and then apply patches.

Apply patches to one machine at a time.

If the system configuration does not consist of one EADS server per machine, the following conditions must be satisfied:

- The number of EADS servers per machine must be less than the data multiplicity.
- The number of EADS servers per machine must be less than half the total number of the EADS servers that constitute the cluster.

11.12.1 How to apply EADS server patches

This subsection explains how to apply EADS server patches.

(1) Verify that there are no errors in any of the EADS servers in the cluster

Execute the `eztool status -v` command to verify that there are no errors in any of the EADS servers in the cluster.

```
eztool status -v
```

(2) Isolate and terminate an EADS server to which patches are to be applied

Execute the `eztool isolate --stop` command to isolate and terminate an EADS server on the machine to which patches are to be applied.

```
eztool isolate --stop
```

If this machine contains other EADS servers, repeat step (2).

(3) Apply patches to the machine

Apply patches to the machine on which all EADS servers have been stopped.

(4) Restart the EADS servers

Execute the `ezstart -r` or `ezserver -r` command to restart the EADS servers to which patches were applied.

Executing the ezstart -r command

```
ezstart -r
```

Executing the ezserver -r command

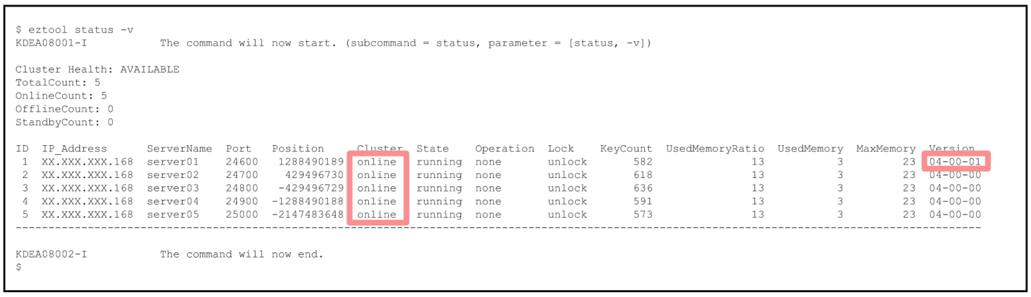
```
ezserver -r
```

(5) Verify that the restarted EADS servers are participating in the cluster

Execute the `eztool status -v` command to verify that the restarted EADS servers are participating in the cluster.

```
eztool status -v
```

Command execution example



```
$ eztool status -v
KDEA08001-I          The command will now start. (subcommand = status, parameter = [status, -v])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address       ServerName  Port   Position     Cluster  State     Operation  Lock     KeyCount  UsedMemoryRatio  UsedMemory  MaxMemory  Version
 1  XX.XXX.XXX.168   server01    24600   1288490189  online   running   none       unlock        582               13           3         23  04-00-01
 2  XX.XXX.XXX.168   server02    24700    429496730  online   running   none       unlock        618               13           3         23  04-00-00
 3  XX.XXX.XXX.168   server03    24800   -429496729  online   running   none       unlock        636               13           3         23  04-00-00
 4  XX.XXX.XXX.168   server04    24900  -1288490188  online   running   none       unlock        591               13           3         23  04-00-00
 5  XX.XXX.XXX.168   server05    25000  -2147483648  online   running   none       unlock        573               13           3         23  04-00-00
-----------------------------------------------------------------------------------------------------------------------------------------------------

KDEA08002-I          The command will now end.
$
```

If an EADS server is participating in the cluster, `online` is displayed in the `Cluster` column.

If there is any other EADS server that is stopped, repeat steps (4) and (5).

The `Version` column displays the version information. Verify that the correct version is displayed.

(6) Verify that all EADS servers in the cluster are participating in the cluster

Execute the `eztool status -v` command to verify that all EADS servers in the cluster are participating in the cluster.

```
eztool status -v
```

Repeat steps (1) through (6) for all machines.

11.13 Obtaining statistics

In EADS, you can obtain the following statistics:

- Statistics (`eads_stats.csv`)
- Cache statistics (`eads_cache_stats.csv`)
- User function statistics (`eads_function_stats.csv`)
- Statistics by range (`eads_store_stats.csv`)

This section explains the locations where statistics are stored, the information that is output as statistics, and the sources of the statistics.

11.13.1 Statistics storage locations

The statistics are stored under *directory-specified-in-the-eads.logger.dir-parameter-in-the-server-properties*`/stats`.

11.13.2 Statistics (eads_stats.csv)

This subsection explains the information that is output as statistics (`eads_stats.csv`) and their source.

(1) Information output as statistics (eads_stats.csv)

The following table lists and describes the information that is output as statistics (`eads_stats.csv`).

Table 11–1: Information output as statistics (eads_stats.csv)

No.	Purpose	Classification	Item	Column name	Output value	Related parameter
1	Checking status	Common item	Date statistics were output	`Date`	Current value	--
2			Time statistics were output	`Time`	Current value	
3			IP address of EADS server for which statistics are output and the EADS server's port number used for communication with EADS clients	`ThisNode`	Setting	• `eads.server.address` • `eads.server.port`
4	Re-evaluating tuning	Memory and buffer	Maximum size of receive data between EADS client and EADS servers (bytes)	`CSReadMaxSize`	Statistic	`eads.server.connection.buffersize`
5			Total size of receive data between EADS client and EADS servers (bytes)	`CSReadTotalSize`	Statistic	• `eads.server.connection.buffersize` • `eads.replication.connection.buffersize`
6			Maximum size of send data between EADS client and EADS servers (bytes)	`CSWriteMaxSize`	Statistic	

No.	Purpose	Classification	Item	Column name	Output value	Related parameter
7			Total size of send data between EADS client and EADS servers (bytes)	`CSWriteTotalSize`	Statistic	
8			Maximum size of receive data between EADS servers (bytes)	`SSReadMaxSize`	Statistic	
9			Total size of receive data between EADS servers (bytes)	`SSReadTotalSize`	Statistic	
10			Maximum size of send data between EADS servers (bytes)	`SSWriteMaxSize`	Statistic	
11			Total size of send data between EADS servers (bytes)	`SSWriteTotalSize`	Statistic	
12			Total size of data transmitted from source EADS server in order to copy data to EADS server to be restored or EADS server added by scale-out processing (bytes)	`TransferCopyDataWriteTotalSize`	Statistic	`eads.transfer.datasize`
13			Total size of all data[#3] sent from source EADS server to EADS server to be restored or EADS server added by scale-out processing (bytes)	`TransferWriteTotalSize`	Statistic	
14	Re-evaluating resource estimations	Number of data items	Number of keys stord in cache	`KeyCount`	Current value	`eads.node.`*EADS-server-ID*`.position`
15		Memory usage amount	Current memory usage amount (sum of values in memory caches) (megabytes)[#4]	`UsedMemorySize`	Current value	
16			Maximum memory size for storing values (megabytes)[#4]	`MaxMemorySize`	Setting	• `eads.java.heapsize` • `eads.java.external.heapsize`
17			Size of space being used in the area for storing the history of update operations (megabytes)[#4]	`HistoriesMemorySize`	Current value	`eads.replication.external.heapsize`
18			Maximum size of the area for storing the history of update operations (megabytes)[#4]	`MaxHistoriesMemorySize`	Setting	
19		Execution count	Total number of requests accepted (`put`, `create`, `update`, `replace`, `get`, `remove`)	`RequestCount`	Statistic	`eads.node.`*EADS-server-ID*`.position`
20			Total number of times individual user functions were executed	`AllFunctionExecuteCount`	Statistic	
21	EADS server performance measurement	Performance	Average internal processing time[#1] (microseconds)	`InternalProcessingAverageTime`	Statistic	

No.	Purpose	Classification	Item	Column name	Output value	Related parameter
22			Maximum request processing time[#2] (microseconds)	`RequestProcessingMaxTime`	Statistic	
23			Average request processing time[#2] (microseconds)	`RequestProcessingAverageTime`	Statistic	
24			Minimum request processing time[#2] (microseconds)	`RequestProcessingMinTime`	Statistic	

Legend:

--: There is no related parameter.

#1

This is the average time spent for the following request processing excluding creation of redundant copies of data:

- `put` and `remove` processing
- `create`, `update`, or `replace` processing to update data

#2

This is the time required to process the entire request (`put` or `remove` processing or `create`, `update`, or `replace` processing to update data) on the first EADS server whose data is manipulated.

#3

During restoration or scale-out processing, the following data is sent to the EADS server subject to restoration or the EADS server that was added during scale-out processing:

- Data to be copied
- If data was updated after restoration or scale-out processing started, a notification of the update operation

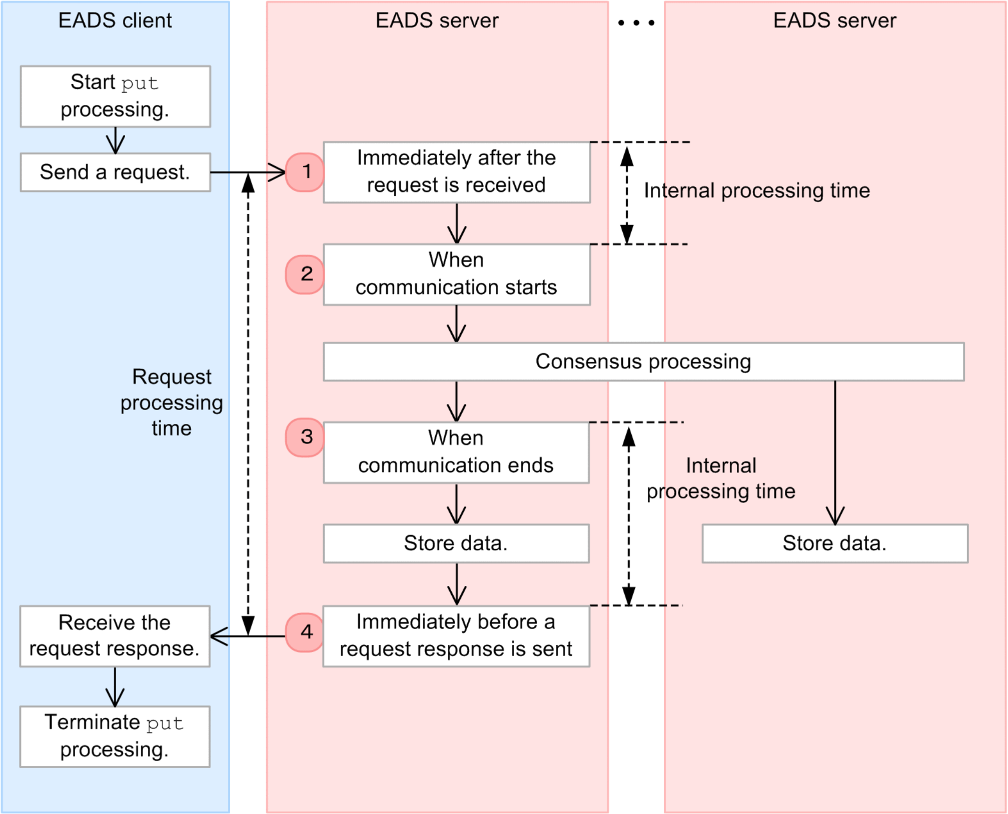
#4

The digits after the decimal point are truncated.

(2) Sources for statistics used for EADS server performance measurement

The following figure gives an example of `put` processing, which shows the sources of the statistics used for EADS server performance measurement.

Figure 11–1: Sources of statistics used for EADS server performance measurement (example of put processing)



Explanation

- The EADS server determines the request processing time based on the difference in time between Nos. 1 and 4. The EADS server also obtains the following statistics:
  - Maximum request processing time (`RequestProcessingMaxTime`)
  - Average request processing time (`RequestProcessingAverageTime`)
  - Minimum request processing time (`RequestProcessingMinTime`)
- The EADS server determines the time spent on consensus processing based on the difference in time between Nos. 2 and 3.
- The internal processing time equals the request processing time minus the time spent on consensus processing. The following statistics are obtained:
  - Average internal processing time (`InternalProcessingAverageTime`)

11.13.3 Cache statistics (eads_cache_stats.csv)

This subsection explains the information that is output as cache statistics (`eads_cache_stats.csv`).

(1) Information output as cache statistics (eads_cache_stats.csv)

The following table lists and describes the information that is output as cache statistics (`eads_cache_stats.csv`).

Table 11–2: Information output as cache statistics (eads_cache_stats.csv)

No.	Purpose	Classification	Item	Column name	Output value	Related parameter
1	Checking status	Common item	Date statistics were output	`Date`	Current value	--
2			Time statistics were output	`Time`	Current value	
3			IP address of EADS server for which statistics are output and the EADS server's port number used for communication with EADS clients	`ThisNode`	Setting	• `eads.server.address` • `eads.server.port`
4	Re-evaluating tuning	Cache settings	Cache name[#1]	`CacheName`	Setting	--
5			Cache type	`CacheType`	Setting	`eads.cache.type`
6			Range ID of the data stored in the cache data file output directory[#2]	`RangeID`	Setting	`eads.replication.factor`
7		Cache usage amount	Number of keys stored in the caches	`CacheKeyCount`	Current value	`eads.node.`*EADS-server-ID*`.position`
8			Memory capacity currently used by the caches (total of values) (megabytes)[#1]	`CacheUsedMemorySize`	Current value	
9			Disk capacity currently used by the caches (total of values) (megabytes)[#3]	`CacheUsedDiskSize`	Current value	
10			Maximum size of cache data files by cache (megabytes)[#3]	`CacheMaxFileSize`	Setting	`eads.cache.disk.filesize`
11			Maximum number of cache data files by cache	`CacheMaxFileCount`	Setting	`eads.cache.disk.filenum`
12			Number of unused cache data files by range	`RangeUnusedFileCount`	Current value	• `eads.cache.disk.filesize` • `eads.cache.disk.filenum`
13		Compaction	File compression that has the largest compaction effects among all cache data files in use by range (%)	`RangeMaxFileCompactionEffect`	Current value	
14			Number of files whose compaction rate is within each range in all cache data files in use by range	`RangeFileCompactionEffectCount`	Current value	• `eads.cache.disk.filesize` • `eads.cache.disk.filenum` • `eads.statistics.compaction.effect.division`
15	EADS server performance measurement	Performance	Average internal processing time by cache[#4] (microseconds)	`CacheInternalProcessingAverageTime`	Statistic	--

No.	Purpose	Classification	Item	Column name	Output value	Related parameter
16			Maximum request processing time by cache#5 (microseconds)	CacheRequestProcessingMaxTime	Statistic	
17			Average request processing time by cache#5 (microseconds)	CacheRequestProcessingAverageTime	Statistic	
18			Minimum request processing time by cache#5 (microseconds)	CacheRequestProcessingMinTime	Statistic	
19			Total number of puts by cache	PutCount	Statistic	
20			Total number of successful puts by cache	PutSuccessCount	Statistic	
21			Total number of gets by cache	GetCount	Statistic	
22			Total number of successful gets by cache	GetSuccessCount	Statistic	
23			Total number of removes by cache	RemoveCount	Statistic	
24			Total number of successful removes by cache	RemoveSuccessCount	Statistic	
25			Total number of creates by cache	CreateCount	Statistic	
26			Total number of successful creates by cache	CreateSuccessCount	Statistic	
27			Total number of updates by cache	UpdateCount	Statistic	
28			Total number of successful updates by cache	UpdateSuccessCount	Statistic	
29			Total number of replaces by cache	ReplaceCount	Statistic	
30			Total number of successful replaces by cache	ReplaceSuccessCount	Statistic	
31			Average internal processing time#6 for putAll by cache (microseconds)	PutAllInternalProcessingAverageTime	Statistic	
32			Maximum time spent on entire putAll processing#7 by cache (microseconds)	PutAllProcessingMaxTime	Statistic	
33			Average time spent on entire putAll processing#7 by cache (microseconds)	PutAllProcessingAverageTime	Statistic	
34			Minimum time spent on entire putAll processing#7 by cache (microseconds)	PutAllProcessingMinTime	Statistic	

No.	Purpose	Classification	Item	Column name	Output value	Related parameter
35			Total number of `putAlls` by cache#8	`PutAllCount`	Statistic	
36			Total number of successful `putAlls` by cache#8	`PutAllSuccessCount`	Statistic	
37			Maximum number of data items#9 manipulated by `putAll` in batch mode by cache	`PutAllBatchDataMaxCount`	Statistic	
38			Average number of data items#9 manipulated by `putAll` in batch mode by cache	`PutAllBatchDataAverageCount`	Statistic	
39			Minimum number of data items#9 manipulated by `putAll` in batch mode by cache	`PutAllBatchDataMinCount`	Statistic	
40			Average internal processing time#6 for `getAll` by cache (microseconds)	`GetAllInternalProcessingAverageTime`	Statistic	
41			Maximum time spent on entire `getAll` processing#7 by cache (microseconds)	`GetAllProcessingMaxTime`	Statistic	
42			Average time spent on entire `getAll` processing#7 by cache (microseconds)	`GetAllProcessingAverageTime`	Statistic	
43			Minimum time spent on entire `getAll` processing#7 by cache (microseconds)	`GetAllProcessingMinTime`	Statistic	
44			Total number of `getAlls` by cache#8	`GetAllCount`	Statistic	
45			Total number of successful `getAlls` by cache#8	`GetAllSuccessCount`	Statistic	
46			Maximum number of data items#9 manipulated by `getAll` in batch mode by cache	`GetAllBatchDataMaxCount`	Statistic	
47			Average number of data items#9 manipulated by `getAll` in batch mode by cache	`GetAllBatchDataAverageCount`	Statistic	
48			Minimum number of data items#9 manipulated by `getAll` in batch mode by cache	`GetAllBatchDataMinCount`	Statistic	
49			Average internal processing time#6 for `removeAll` by cache (microseconds)	`RemoveAllInternalProcessingAverageTime`	Statistic	

No.	Purpose	Classification	Item	Column name	Output value	Related parameter
50			Maximum time spent on entire `removeAll` processing#7 by cache (microseconds)	`RemoveAllProcessingMaxTime`	Statistic	
51			Average time spent on entire `removeAll` processing#7 by cache (microseconds)	`RemoveAllProcessingAverageTime`	Statistic	
52			Minimum time spent on entire `removeAll` processing#7 by cache (microseconds)	`RemoveAllProcessingMinTime`	Statistic	
53			Total number of `removeAll`s by cache#8	`RemoveAllCount`	Statistic	
54			Total number of successful `removeAll`s by cache#8	`RemoveAllSuccessCount`	Statistic	
55			Maximum number of data items#9 manipulated by `removeAll` in batch mode by cache	`RemoveAllBatchDataMaxCount`	Statistic	
56			Average number of data items#9 manipulated by `removeAll` in batch mode by cache	`RemoveAllBatchDataAverageCount`	Statistic	
57			Minimum number of data items#9 manipulated by `removeAll` in batch mode by cache	`RemoveAllBatchDataMinCount`	Statistic	

Legend:

--: There is no related parameter.

#1

When there are multiple caches, the items are displayed in the order of the cache names.

#2

If the number of redundant copies of data plus the original is 2 or greater, this item is displayed from the top in the order of the target EADS servers into which data is stored, and then for the source EADS server from which the data is copied.

#3

The digits after the decimal point are truncated.

#4

This is the average time spent for the following request processing excluding creation of redundant copies of data:

- `put` and `remove` processing
- `create`, `update`, or `replace` processing to update data

#5

This is the time required to process the entire request (`put` or `remove` processing or `create`, `update`, or `replace` processing to update data) on the first EADS server whose data is manipulated.

#6

This is the time required for the entire processing minus the time spent on communication between EADS servers.

#7

This does not include the processing time if any part of a batch operation fails.

#8

This is the execution count for EADS servers. It does not match the execution count for EADS clients.

#9

This does not include the number of data items if any part of a batch operation fails.

11.13.4 User function statistics (eads_function_stats.csv)

This subsection explains the information that is output as user function statistics (`eads_function_stats.csv`) and their source.

(1) Information output as user function statistics (eads_function_stats.csv)

The table below lists and describes the information that is output as user function statistics (`eads_function_stats.csv`).

This information can be used for user function performance measurement.

Table 11–3: Information output as user function statistics (eads_function_stats.csv)

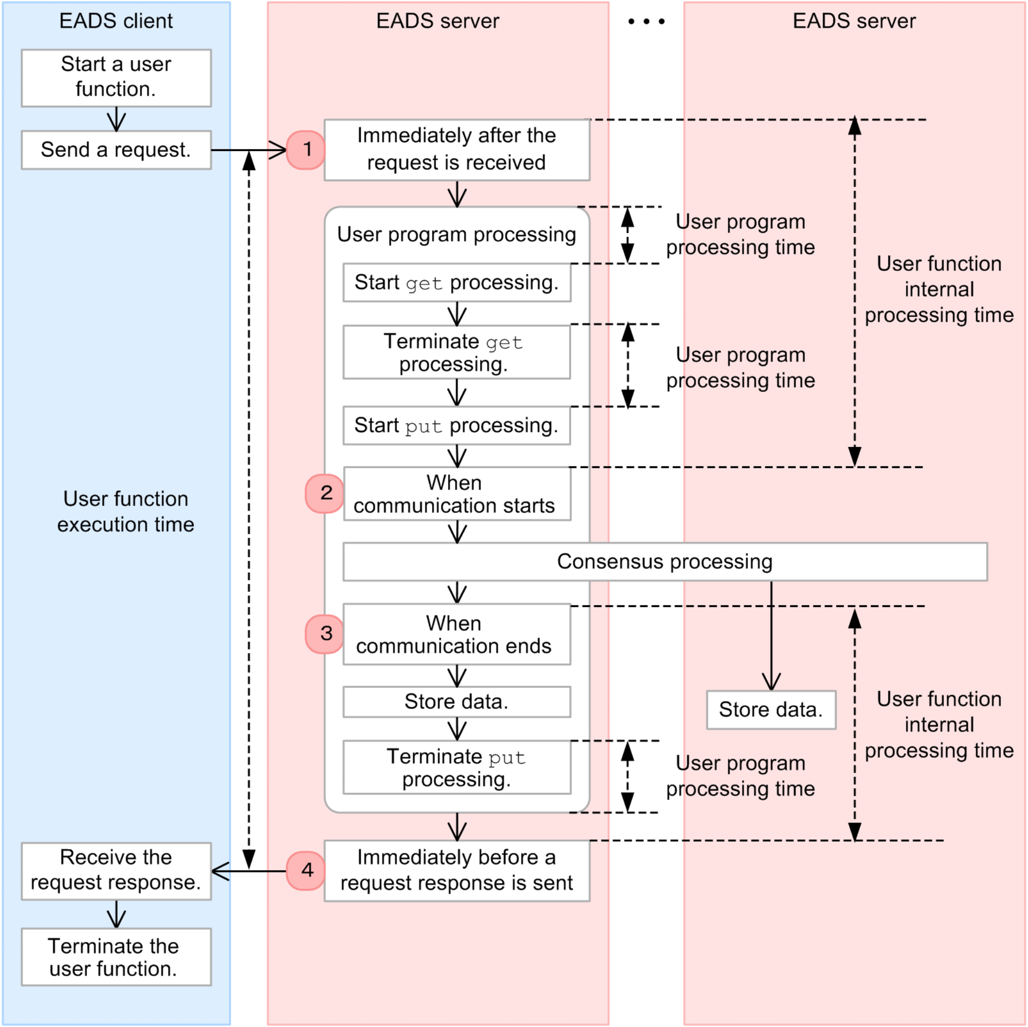
No.	Item	Column name	Output value
1	Date statistics were output	`Date`	Current value
2	Time statistics were output	`Time`	Current value
3	IP address of an EADS server for which statistics are output and the EADS server's port number used for communication with EADS clients	`ThisNode`	Setting
4	User function name	`FunctionName`	Setting
5	Total number of times user functions were executed	`FunctionExecuteCount`	Statistic
6	Maximum user function execution time (microseconds)	`FunctionExecuteMaxTime`	Statistic
7	Average user function execution time (microseconds)	`FunctionExecuteAverageTime`	Statistic
8	Minimum user function execution time (microseconds)	`FunctionExecuteMinTime`	Statistic
9	Maximum user function internal processing time (microseconds)	`FunctionInternalProcessingMaxTime`	Statistic
10	Average user function internal processing time (microseconds)	`FunctionInternalProcessingAverageTime`	Statistic
11	Minimum user function internal processing time (microseconds)	`FunctionInternalProcessingMinTime`	Statistic

No.	Item	Column name	Output value
12	Maximum user program processing time in the user function (microseconds)	`UserProgramProcessingMaxTime`	Statistic
13	Average user program processing time in the user function (microseconds)	`UserProgramProcessingAverageTime`	Statistic
14	Minimum user program processing time in the user function (microseconds)	`UserProgramProcessingMinTime`	Statistic
15	Total number of `puts` called by user functions	`PutCount`	Statistic
16	Total number of successful `puts` called by user functions	`PutSuccessCount`	Statistic
17	Total number of `gets` called by user functions	`GetCount`	Statistic
18	Total number of successful `gets` called by user functions	`GetSuccessCount`	Statistic
19	Total number of `removes` called by user functions	`RemoveCount`	Statistic
20	Total number of successful `removes` called by user functions	`RemoveSuccessCount`	Statistic
21	Total number of `creates` called by user functions	`CreateCount`	Statistic
22	Total number of successful `creates` called by user functions	`CreateSuccessCount`	Statistic
23	Total number of `updates` called by user functions	`UpdateCount`	Statistic
24	Total number of successful `updates` called by user functions	`UpdateSuccessCount`	Statistic
25	Total number of `replaces` called by user functions	`ReplaceCount`	Statistic
26	Total number of successful `replaces` called by user functions	`ReplaceSuccessCount`	Statistic

(2) Sources of statistics

The following figure shows the sources of the statistics.

Figure 11–2: Sources of statistics (user function statistics)


EADS client
EADS server
• • •
EADS server
Start a user function.
Send a request.
User function execution time
Receive the request response.
Terminate the user function.
1
Immediately after the request is received
User program processing
Start get processing.
Terminate get processing.
Start put processing.
2
When communication starts
Consensus processing
3
When communication ends
Store data.
Terminate put processing.
4
Immediately before a request response is sent
User program processing time
User program processing time
User program processing time
User function internal processing time
Store data.
User function internal processing time

Explanation

- The user function execution time is the processing time between when a user function execution request is accepted and a response is sent. The EADS server determines the user function execution time based on the difference in time between Nos. 1 and 4. The EADS server also obtains the following statistics:
  - Maximum user function execution time (`FunctionExecuteMaxTime`)
  - Average user function execution time (`FunctionExecuteAverageTime`)
  - Minimum user function execution time (`FunctionExecuteMinTime`)
- The EADS server determines the time spent on consensus processing based on the difference in time between Nos. 2 and 3.
- The user function internal processing time equals the user function execution time minus the time spent on consensus processing. The EADS server also obtains the following statistics:
  - Maximum user function internal processing time (`FunctionInternalProcessingMaxTime`)
  - Average user function internal processing time (`FunctionInternalProcessingAverageTime`)
  - Minimum user function internal processing time (`FunctionInternalProcessingMinTime`)
- The user program processing time equals the total time spent on user program processing minus the time spent on consensus processing minus the API function processing time. The EADS server also obtains the following statistics:

- Maximum user program processing time in the user function (UserProgramProcessingMaxTime)
- Average user program processing time in the user function (UserProgramProcessingAverageTime)
- Minimum user program processing time in the user function (UserProgramProcessingMinTime)

11.13.5 Statistics by range (eads_store_stats.csv)

This subsection explains the information that is output as statistics by range (eads_store_stats.csv).

Note that the statistics by range (eads_store_stats.csv) are output when the total data restriction function is enabled.

(1) Information output as statistics by range (eads_store_stats.csv)

The following table lists and describes the information that is output as statistics by range (eads_store_stats.csv).

Table 11–4: Information output as statistics by range (eads_store_stats.csv)

No.	Purpose	Classification	Item	Column name	Output value	Related parameter
1	Checking status	Common item	Date statistics were output	Date	Current value	--
2			Time statistics were output	Time	Current value	
3			IP address of an EADS server for which statistics are output and the EADS server's port number used for communication with EADS clients	ThisNode	Setting	• eads.server.address • eads.server.port
4	Restricting the total amount of data	Range information	Data range ID[#1]	RangeID	Setting	eads.replication.factor
5			Cache name[#2]	CacheName	Setting	--
6			Maximum number of keys that can be stored in ranges	KeyCountLimit	Setting	eads.cache.keyCount
7			Number of keys stored in ranges	StoredKeyMaxCount	Statistic	--
8			Number of keys that have not been stored in ranges yet	ReservedKeyMaxCount	Statistic	• eads.server.cache.maxExecuteThreads • eads.server.function.maxExecuteThreads
9		Memory usage amount	Maximum memory size available to the values that can be stored in ranges (megabytes)	ExternalHeapSizeLimit	Setting	eads.java.external.heapsize

No.	Purpose	Classification	Item	Column name	Output value	Related parameter
10			Memory size in use by the values that are stored in ranges (megabytes)[#3]	UsedExternalHeapMaxSize	Statistic	--
11			Memory size available to the values that have not been stored in ranges yet (bytes)	ReservedExternalHeapMaxSize	Statistic	• eads.server.cache.maxExecuteThreads • eads.server.function.maxExecuteThreads
12		Disk usage amount	Maximum disk capacity available to the records that can be stored in ranges (megabytes)	DiskSizeLimit	Setting	• eads.cache.disk.filesize • eads.cache.disk.filenum
13			Disk capacity in use by the records that are stored in ranges (megabytes)[#3]	UsedDiskMaxSize	Statistic	--
14			Disk capacity available to the records that have not been stored in ranges yet (bytes)	ReservedDiskMaxSize	Statistic	• eads.server.cache.maxExecuteThreads • eads.server.function.maxExecuteThreads

Legend:

--: There is no related parameter.

#1

If the number of redundant copies of data plus the original is 2 or greater, this item is displayed from the top in the order of the target EADS servers into which data is stored, and then for the source EADS server from which the data is copied.

#2

When there are multiple caches, this item is displayed in the order of the cache names.

#3

The digits after the decimal point are truncated.

11.14 Managing available space in the data storage

When `true` is specified in the `eads.cache.limiter.enable` parameter, the amount of data that can be stored in EADS servers (number of stored keys, available space in the explicit heap, available space in cache files) is monitored. If a shortage of storage space is foreseen, you can set an error in the corresponding processing to prevent the EADS servers from being shut down. This is called the *total data restriction function*.

When the total data restriction function is enabled, the number of keys, the capacity of the explicit heap, and the capacity of cache files are monitored so that the following maximum limits are not exceeded:

- Maximum number of keys

  `eads.cache.keyCount` parameter value in the shared properties

- Maximum capacity of the explicit heap (megabytes)

Maximum capacity of the explicit heap (megabytes) =
(*eads.java.external.heapsize parameter value in the shared properties* × 0.97
- *eads.replication.external.heapsize parameter value in the shared properties)*
÷ *eads.replication.factor parameter value in the shared properties*

Notes:

A value of less than 1 megabyte is truncated.

- Maximum capacity of cache files (megabytes)

Maximum capacity of cache files (megabytes) =
(*eads.cache.disk.filenum parameter value in the cache properties* - 2)
× (*eads.cache.disk.filesize parameter value in the cache properties* × 1,024
- *eads.cache.disk.blocksize parameter value in the cache properties* × 2) ÷ 1,024

Notes:

A value of less than 1 megabyte is truncated.

Tip

When the total data restriction function is enabled and any of these maxima is expected to be exceeded as a result of storing data (a shortage of space might occur in the storage), the processing results in an error. Therefore, if an attempt is made to perform multiple processes that store data concurrently when available storage space is limited, the processes might result in an error.

When you use the total data restriction function, your estimates of the number of keys, the capacity of the explicit heap, and the capacity of cache files need to be set to values that are larger than the actual values.

For details about the estimation method when using the total data restriction function, see *4. Checking the Required Resources*.

Note that the total data restriction function is disabled temporarily while any of the following types of processing is being performed:

- Scale-out processing
- Importing data (`eztool import` command)
- Relocating persistent data (`eztool importecf` command)
- Resuming caches (`eztool resume` command)

11.14.1 How to manage available space in the data storage (using only memory caches)

This subsection explains how to monitor available space in the data storage and how to increase available space when using only memory caches.

(1) Monitor available space by using statistics

Estimate the available space in the data storage from the information that is output as statistics by range (`eads_store_stats.csv`). Monitor these values.

Statistics by range (`eads_store_stats.csv`) are output only when the total data restriction function is used.

For details about the statistics by range (`eads_store_stats.csv`), see *11.13.5 Statistics by range (eads_store_stats.csv)*.

If the available space in the data storage is low, take the actions described in the following subsections.

(a) Deleting unneeded data

Increase the available space in the data storage by deleting unneeded data.

(b) Adding EADS servers to the cluster

If enough space cannot be obtained from (a), add EADS servers to the cluster. Adding EADS servers to the cluster enables you to reduce the number of data items stored in each EADS server.

For details about how to add EADS servers to a cluster, see *11.1.1 How to add EADS servers to a cluster without stopping the cluster (scale-out processing)* or *11.1.2 How to add EADS servers to a cluster after stopping the cluster (using only memory caches)*.

Note that because the total data restriction function is disabled during scale-out processing, sufficient space (at least twice the number of data items and capacity that are added during scale-out processing) is required.

(2) If processing results in an error due to a space shortage

This subsection explains how to handle processing errors that result from a shortage of space.

(a) Check the storage that has reached the maximum amount of data that can be stored on EADS servers

Execute the `eztool storeusage` command to determine from the number of keys and amount of memory usage the storage that has reached the maximum amount of data that can be stored on EADS servers.

(b) Terminate all EADS servers in the cluster (after exporting data to files)

Export data to files and then terminate all EADS servers in the cluster.

For details about this procedure, see *10.5 Terminating the EADS servers (after exporting data from memory to files)*.

(c) Change the properties

Re-estimate the Java heap size and the explicit heap size, and then change the properties based on the estimation results. If necessary, add server machines and memory.

For details about how to estimate these values, see the following:

- Estimating the Java heap size
  4.1.2 Estimating the Java heap size
- Estimating the explicit heap size
  4.1.3 Estimating the explicit heap size

For details about how to change the properties, see *11.4 Changing the properties*.

(3) Start all EADS servers in the cluster (and import data from files)

Start all EADS servers in the cluster, and then import back into memory the data that was exported to files during the previous session.

For details about the procedure, see *10.3 Starting the EADS servers (and creating caches by importing data from files)*.

11.14.2 How to manage available space in the data storage (using disk caches)

This subsection is applicable when you will be using disk caches and two-way caches.

> **Reference note**
>
> The range of applicability of the total data restriction function on cache files is only the source EADS servers from which data is copied. If a shortage of space occurs on a target EADS server, the target EADS server is isolated. If you will be performing compaction processing on cache files, make sure that the available space is the same on all EADS servers.
>
> Note that the total data restriction function is disabled on an EADS server while it is under restoration processing.

This subsection explains how to monitor available space in the data storage and how to increase available space when using disk caches and two-way caches.

(1) Monitor available space by using statistics

Estimate the available space in the data storage from the information that is output as statistics by range (`eads_store_stats.csv`). Monitor these values.

Statistics by range (`eads_store_stats.csv`) are output only when the total data restriction function is used.

For details about the statistics by range (`eads_store_stats.csv`), see *11.13.5 Statistics by range (eads_store_stats.csv)*.

If the available space in the data storage is low, take the actions described in the following subsections.

(a) Deleting unneeded data

Increase the available space in the data storage by deleting unneeded data.

(b) Performing compaction

Perform compaction to reduce the amount of data stored in cache data files.

For details about how to perform compaction on cache data files, see *10.9.1 Performing compaction on cache data files*.

(c) Adding EADS servers to the cluster

If enough space cannot be obtained from (a) and (b), add EADS servers to the cluster. Adding EADS servers to the cluster enables you to reduce the number of data items stored in each EADS server.

For details about how to add EADS servers to a cluster, see *11.1.3 How to add EADS servers to a cluster after stopping the cluster (using only disk caches)*.

(2) If processing results in an error due to a space shortage

This subsection explains how to handle processing errors that result from a shortage of space.

(a) Check the storage that has reached the maximum amount of data that can be stored on EADS servers

Execute the `eztool storeusage --cache` command to determine from the number of keys, amount of memory usage, and amount of cache file usage the storage that has reached the maximum amount of data that can be stored on EADS servers.

(b) Terminate all EADS servers in the cluster

Terminate all EADS servers in the cluster.

For details about this procedure, see *10.6 Terminating the EADS servers (terminating caches on disk)*.

(c) Change the properties

Re-estimate the Java heap size, the explicit heap size, and the capacity of cache files, and then change the properties based on the estimation results. If necessary, add server machines, memory, and disks.

For details about how to estimate these values, see the following:

- Estimating the Java heap size
  4.1.2 Estimating the Java heap size
- Estimating the explicit heap size
  4.1.3 Estimating the explicit heap size
- Estimating the capacity of cache files
  4.4 Estimating the sizes of cache files

For details about how to change the properties, see *11.4 Changing the properties*.

(d) Start all EADS servers in the cluster (relocate data)

Relocate data after all EADS servers in the cluster have started.

For details about this procedure, see *11.1.3(5) Start all EADS servers in the cluster* through *11.1.3(9) Verify that the EADS servers have been opened*.

11.15 Managing cache files

This section is applicable when you will be using disk caches and two-way caches.

Delete unneeded cache files, such as when caches are created using incorrect properties.

11.15.1 How to check and delete cache files

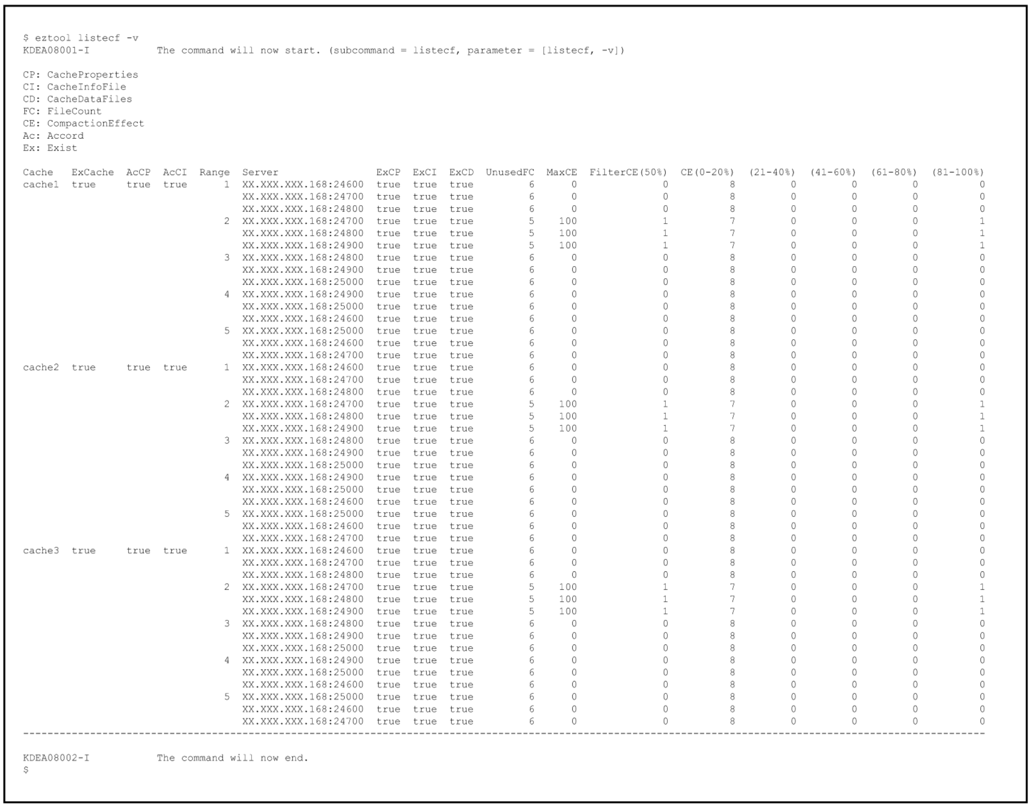
This subsection explains how to check and delete cache files.

(1) Check for caches that have cache files

Execute the `eztool listecf -v` command to check for caches that have cache files.

```
eztool listecf -v
```

Command execution example



```
$ eztool listecf -v
KDEA08001-I          The command will now start. (subcommand = listecf, parameter = [listecf, -v])

CP: CacheProperties
CI: CacheInfoFile
CD: CacheDataFiles
FC: FileCount
CE: CompactionEffect
Ac: Accord
Ex: Exist

Cache    ExCache  AcCP  AcCI  Range  Server                ExCP  ExCI  ExCD  UnusedFC  MaxCE  FilterCE(50%)  CE(0-20%)  (21-40%)  (41-60%)  (61-80%)  (81-100%)
cache1   true     true  true      1  XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24700  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24800  true  true  true         6      0              0          8         0         0         0          0
                                  2  XX.XXX.XXX.168:24700  true  true  true         5    100              1          7         0         0         0          1
                                     XX.XXX.XXX.168:24800  true  true  true         5    100              1          7         0         0         0          1
                                     XX.XXX.XXX.168:24900  true  true  true         5    100              1          7         0         0         0          1
                                  3  XX.XXX.XXX.168:24800  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24900  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                  4  XX.XXX.XXX.168:24900  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                  5  XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24700  true  true  true         6      0              0          8         0         0         0          0
cache2   true     true  true      1  XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24700  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24800  true  true  true         6      0              0          8         0         0         0          0
                                  2  XX.XXX.XXX.168:24700  true  true  true         5    100              1          7         0         0         0          1
                                     XX.XXX.XXX.168:24800  true  true  true         5    100              1          7         0         0         0          1
                                     XX.XXX.XXX.168:24900  true  true  true         5    100              1          7         0         0         0          1
                                  3  XX.XXX.XXX.168:24800  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24900  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                  4  XX.XXX.XXX.168:24900  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                  5  XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24700  true  true  true         6      0              0          8         0         0         0          0
cache3   true     true  true      1  XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24700  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24800  true  true  true         6      0              0          8         0         0         0          0
                                  2  XX.XXX.XXX.168:24700  true  true  true         5    100              1          7         0         0         0          1
                                     XX.XXX.XXX.168:24800  true  true  true         5    100              1          7         0         0         0          1
                                     XX.XXX.XXX.168:24900  true  true  true         5    100              1          7         0         0         0          1
                                  3  XX.XXX.XXX.168:24800  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24900  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                  4  XX.XXX.XXX.168:24900  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                  5  XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                     XX.XXX.XXX.168:24700  true  true  true         6      0              0          8         0         0         0          0
-----------------------------------------------------------------------------------------------------------------------------------------------------------

KDEA08002-I          The command will now end.
$
```

- For a cache that has a cache information file, `true` is displayed in the `ExCI` column.
- For a cache that has a cache data file, `true` is displayed in the `ExCD` column.

(2) Delete unneeded cache files

Execute the `eztool deleteecf` command to delete unneeded cache files from the cluster.

```
eztool deleteecf cache-name
```

Command execution example

```
$ eztool deleteecf cache1
KDEA08001-I           The command will now start. (subcommand = deleteecf, parameter =
[deleteecf, cache1])
KDEA08064-I           The cache files were deleted.
KDEA08002-I           The command will now end.
$
```

12 Error Handling Operations

This chapter explains the tasks that the system operation administrator must perform in the event of a failure.

12.1 Preventing failures (error monitoring and detection)

By using system operations management software such as JP1 to monitor messages issued by EADS and the EADS server processes, the system operation administrator can be notified of failures and of the status of the EADS server processes.

12.1.1 Monitoring messages

Monitoring messages involves monitoring the EADS system for messages and notifying the system operation administrator of failures.

Monitoring messages means to monitor the following logs:

Monitored program		Output destination	Rotation method	Log file name	Format	End-of-line code
Process	Log					
EADS server	Message log	Directory specified in the `eads.logger.dir` parameter in the server properties	`Wrap` (default)	`eads_server_message[`*n*`].log`	Wraparound file#1	`CR` + `LF`
			`Shift`	`eads_server_message.log`		
Commands		Directory specified in the `eads.command.logger.dir` parameter in the command properties	`Wrap` (default)	`eads_management_message[`*n*`].log`		
			`Shift`	`eads_command_message.log`		
EADS client		*directory-specified-in-eads.client.logger.dir-parameter-in-the-client-properties / EADS-client-name*#2	`Wrap`	`eads_client_message[`*n*`].log`		

Legend:

`[`*n*`]`: Sequence number of the file

`Wrap`: Wrap-around method

`Shift`: Shift method

#1

When the file wraps around, existing data is deleted and new data is overwritten from the beginning of the file.

#2

This is the EADS client name specified in the client API function. If the EADS client name is the null character string, the subdirectory of the EADS client name is omitted.

12.1.2 Monitoring the EADS server processes

Monitoring the EADS server processes involves monitoring the EADS server processes and notifying the system operation administrator of process statuses and failures.

You monitor the EADS server processes by periodically executing the `eztool status -s` command. After executing this command, check the return code to determine the EADS server status.

For details about the return codes of the `eztool status -s` command, see *14.3.4(6) Return code*.

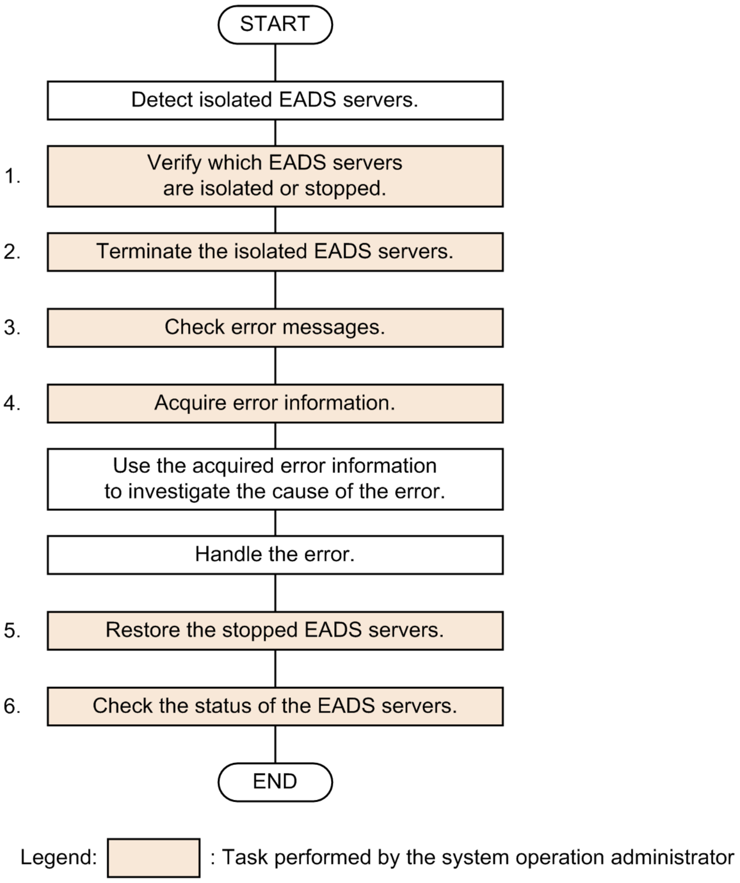
12.2 The system operation administrator's tasks in the event of a failure

In the event of a failure, the system operation administrator must check the contents of the error message and collect error information. This section explains the system operation administrator's tasks, corresponding to different failures.

12.2.1 If one or more EADS servers are isolated

The following figure shows the general procedure for restoring one or more EADS servers that have been isolated due to failures.

Figure 12–1: General procedure for restoring one or more EADS servers that have been isolated due to failures


START
Detect isolated EADS servers.
1. Verify which EADS servers are isolated or stopped.
2. Terminate the isolated EADS servers.
3. Check error messages.
4. Acquire error information.
Use the acquired error information to investigate the cause of the error.
Handle the error.
5. Restore the stopped EADS servers.
6. Check the status of the EADS servers.
END
Legend: ▨ : Task performed by the system operation administrator

Important note

In the following cases, the EADS servers cannot be restored using the procedures explained here:

- The cluster is unavailable (NOT_AVAILABLE) or is partially available (PARTIALLY_AVAILABLE).

  If the cluster status is AVAILABLE but at least half of the EADS servers in the cluster are isolated, the same measures are needed as when a cluster is unavailable (NOT_AVAILABLE).
- Online performance has degraded beyond what is allowed.
- An EADS server to be restored is not defined in the cluster properties.

- The cluster properties in effect when an EADS server was shut down do not match the cluster properties in effect during restoration.

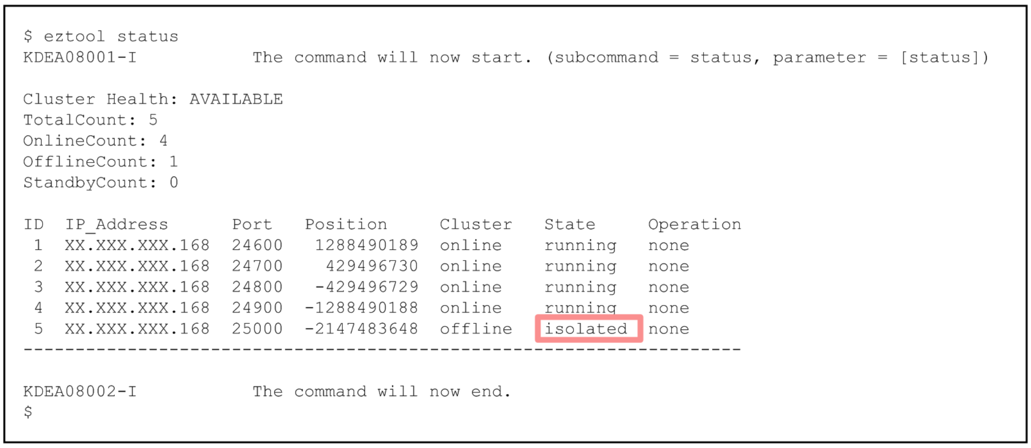
For details about the restoration procedure when the cluster is unavailable (NOT_AVAILABLE) or is partially available (PARTIALLY_AVAILABLE), see *12.2.2 If the cluster is unavailable (NOT_AVAILABLE) or is partially available (PARTIALLY_AVAILABLE)*.

Each of the system operation administrator's tasks is explained in more detail below.

(1) Verify which EADS servers are isolated or stopped

Execute the `eztool status` command to verify which EADS servers are isolated or stopped.

Command execution example



```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 4
OfflineCount: 1
StandbyCount: 0

ID  IP_Address       Port   Position     Cluster   State     Operation
 1  XX.XXX.XXX.168  24600   1288490189  online    running   none
 2  XX.XXX.XXX.168  24700    429496730  online    running   none
 3  XX.XXX.XXX.168  24800   -429496729  online    running   none
 4  XX.XXX.XXX.168  24900  -1288490188  online    running   none
 5  XX.XXX.XXX.168  25000  -2147483648  offline   isolated  none
----------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

In this example, the isolated EADS server is indicated as `isolated` in the `State` column. You must terminate this EADS server.

In this example, there is no stopped EADS server. If there is any stopped EADS server, it is indicated as `----------` in the `State` column.

(2) Terminate the isolated EADS servers

If an EADS server is isolated, use the `eztool isolate --stop` command to terminate it. If no EADS servers are isolated, skip this step.

Important note

Execute the `eztool isolate --stop` command on the isolated EADS server.

(3) Check error messages

Check the error message output to the message log of the EADS server that you terminated in (2) above.

(4) Acquire error information

You need information to investigate the cause of the error. Obtain the following information on all EADS servers:

- All files under the directory specified in the `eads.logger.dir` parameter in the server properties
- All property files under *management-directory*`/conf`
- Thread dumps

You can use the `eztool snapshot` command to collect logs and property files in a single batch operation.

For details about how to acquire error information, see *12.3 Acquiring error information*.

Determining the time of EADS server isolation

See the `KDEA04783-I` or `KDEA04799-E` message that has been output to the message log of an EADS server that was isolated. The time of this message is the time the EADS server was isolated.

Example message (KDEA04783-I)

```
        :
0385 2015/04/03 11:59:25.354 EADS 4A913FE2 276A38B5 KDEA04783-I Processing to isolate a
server will now start. (server ID = 1)
        :
```

In this example, the EADS server whose EADS server ID is `1` was isolated on 2015-04-03 at 11:59:25.

Example message (KDEA04799-E)

```
        :
0953 2015/04/21 11:55:46.116 EADS 6EF0EED6 45E67E6A KDEA04799-E Processing
to isolate a server will now start. (server ID = 3)
        :
```

In this example, the EADS server whose EADS server ID is `3` was isolated on 2015-04-21 at 11:55:46.

(5) Restore the stopped EADS servers

After handling the errors, restore the stopped EADS servers by using one of the following commands:

- `ezstart -r` command
- `ezserver -r` command

During restoration processing, an active EADS server sends data to the EADS servers being restored in order to recover data consistency.

Therefore, note the following:

- To restore an EADS server, it takes at least the time required for obtaining data.
- The EADS server that sends data is affected correspondingly by the amount of CPU resources and network bandwidth that are allocated for sending data.
- If the EADS server cannot keep up with the processing because both data operations and restoration processing must be performed, the EADS server might place data operations on hold to prevent a memory shortage.

Tip

If you are using disk caches or two-way caches, restore the EADS server by using one of the following methods:

Restoration method	Restoration procedure	Processing	Criteria
Using cache files for restoration	Use the `ezstart -r` or `ezserver -r` command to restore the EADS servers that have stopped without deleting the EADS servers' cache files.	Imports data from the cache files and corrects the data by comparing with the data for an active EADS server.	If the frequency of data update and deletion processing on the cache is low and the cache files contain a large amount of valid data, the time required for restoration processing might be reduced by using cache files, as compared with when cache files are not used.
Not using cache files for restoration	Delete the cache data files for the corresponding cache by executing the `deleteecf -l` command with the EADS servers that have stopped specified. Then execute the `ezstart -r` or `ezserver -r` command to restore the EADS servers. If this restoration processing fails, perform this procedure again.	Acquires all data from an active EADS server.	If the frequency of data update and deletion processing on the cache is high and the cache files contain a large amount of invalid data, the time required for restoration processing might be reduced by not using cache files, as compared with when cache files are used.

If a space shortage occurs in cache data files during data restoration processing, compaction is performed internally. If the space shortage is resolved by this compaction processing, the restoration processing resumes.

If the space shortage cannot be resolved, increase the value of the `eads.cache.disk.filenum` parameter in the cache properties according to the procedure described in *11.4.1 How to change the properties*.

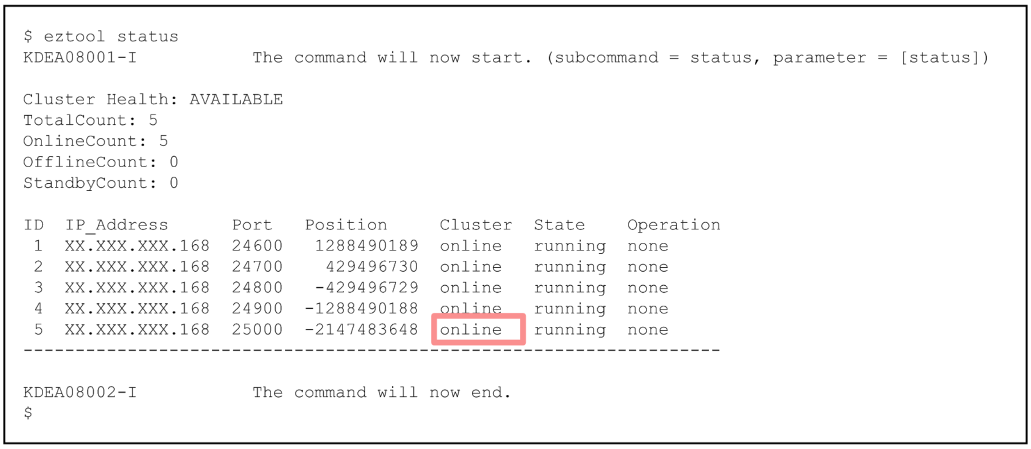
If restoration processing fails for any of the reasons listed below, delete the cache files from the cache that contains the corrupted files, execute the `ezstart -r` or `ezserver -r` command, and then restore the EADS servers that have stopped:

- Cache files have become corrupted.
- A Java heap overflow occurred.
- Internal compaction processing failed.

(6) Verify that the restarted EADS servers are participating in the cluster

Execute the `eztool status` command to verify that the restarted EADS servers have been restored in the cluster.

Command execution example



```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      Port   Position     Cluster  State    Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   running  none
 2  XX.XXX.XXX.168  24700    429496730  online   running  none
 3  XX.XXX.XXX.168  24800   -429496729  online   running  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   running  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   running  none
--------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

If an EADS server is participating in the cluster, `online` is displayed in the `Cluster` column.

If there is any other EADS server that is isolated or stopped, repeat the procedure starting from *12.2.1(1) Verify which EADS servers are isolated or stopped*.

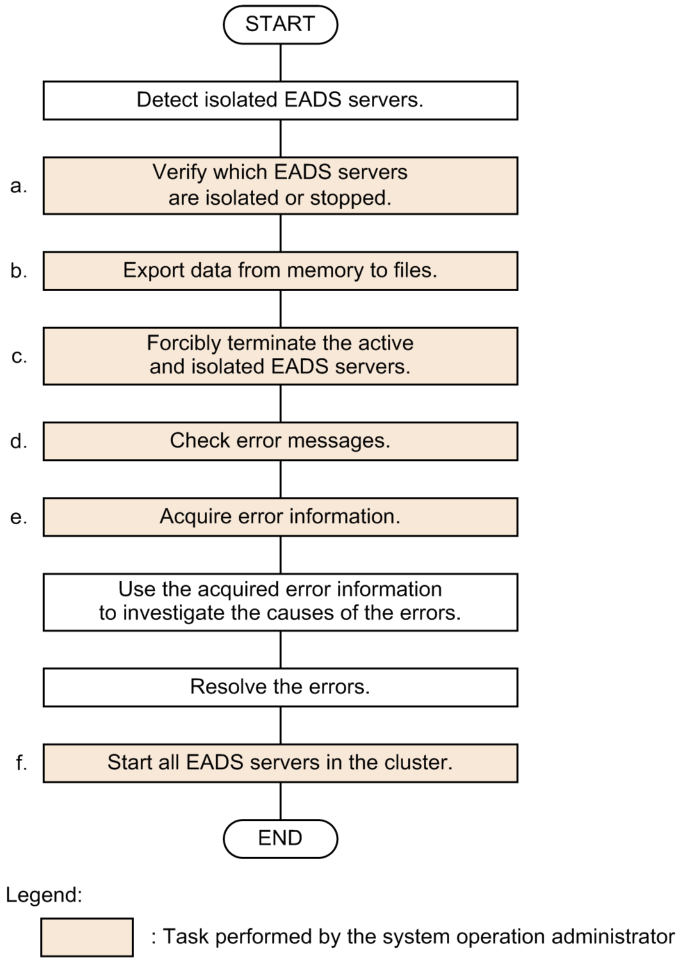
12.2.2 If the cluster is unavailable (NOT_AVAILABLE) or is partially available (PARTIALLY_AVAILABLE)

This subsection explains the restoration procedure when the cluster is unavailable (`NOT_AVAILABLE`) or is partially available (`PARTIALLY_AVAILABLE`).

(1) When using only memory caches

The following figure shows the general restoration procedure when memory caches are used and the cluster is unavailable (`NOT_AVAILABLE`) or is partially available (`PARTIALLY_AVAILABLE`).

Figure 12–2: Restoration procedure when the cluster is unavailable (NOT_AVAILABLE) or is partially available (PARTIALLY_AVAILABLE) (using only memory caches)


START
Detect isolated EADS servers.
a.
Verify which EADS servers are isolated or stopped.
b.
Export data from memory to files.
c.
Forcibly terminate the active and isolated EADS servers.
d.
Check error messages.
e.
Acquire error information.
Use the acquired error information to investigate the causes of the errors.
Resolve the errors.
f.
Start all EADS servers in the cluster.
END
Legend:
▭ : Task performed by the system operation administrator

Each of the system operation administrator's tasks is explained in more detail below.

(a) Verify which EADS servers are isolated or stopped

Verify which EADS servers are isolated or stopped.

For details about this procedure, see *12.2.1(1) Verify which EADS servers are isolated or stopped*.

(b) Export data from memory to files

Execute the `eztool export -s` command to export data from each individual memory. You must execute this command for each EADS server.

```
eztool export -s
```

Important note

Note that there might not be consensus on the request received immediately before the range became unavailable. This means that the exported data might not be consistent.

(c) Forcibly terminate the active and isolated EADS servers

Execute the eztool forcestop command to forcibly terminate the active and isolated EADS servers.

If all EADS servers have already stopped, there is no need to execute this command.

Execute the eztool forcestop command on the active and isolated EADS servers.

```
eztool forcestop
```

(d) Check error messages

Check the error messages output to the message logs of the EADS servers that you identified in (a) above.

(e) Acquire error information

Acquire error information on all EADS servers.

For details about this procedure, see *12.2.1(4) Acquire error information*.

(f) Start all EADS servers in the cluster (import data from files)

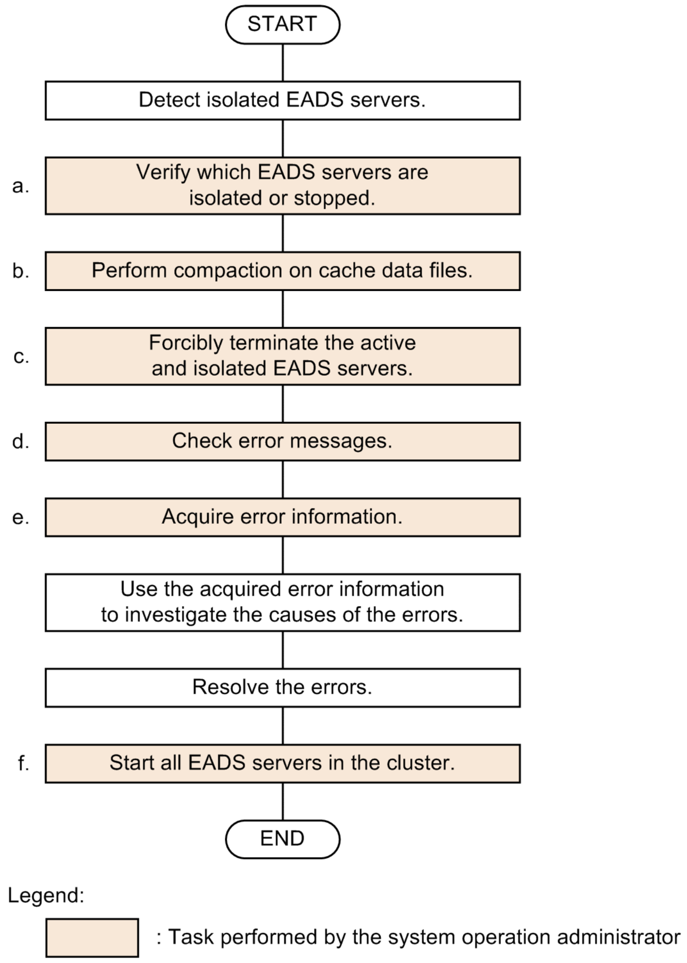
After resolving the errors, start all EADS servers in the cluster and re-import to memory the data that was exported to files in subsection (b).

For details about this procedure, see *10.3 Starting the EADS servers (and creating caches by importing data from files)*.

(2) When using disk caches

The following figure shows the general restoration procedure when disk caches are used and the cluster is unavailable (NOT_AVAILABLE) or is partially available (PARTIALLY_AVAILABLE).

Figure 12–3: Restoration procedure when the cluster is unavailable (NOT_AVAILABLE) or is partially available (PARTIALLY_AVAILABLE) (using disk caches)


START
Detect isolated EADS servers.
a.
Verify which EADS servers are isolated or stopped.
b.
Perform compaction on cache data files.
c.
Forcibly terminate the active and isolated EADS servers.
d.
Check error messages.
e.
Acquire error information.
Use the acquired error information to investigate the causes of the errors.
Resolve the errors.
f.
Start all EADS servers in the cluster.
END
Legend:
: Task performed by the system operation administrator

Each of the system operation administrator's tasks is explained in more detail below.

(a) Verify which EADS servers are isolated or stopped

Verify which EADS servers are isolated or stopped.

For details about this procedure, see *12.2.1(1) Verify which EADS servers are isolated or stopped*.

(b) Perform compaction on cache data files

Execute the `compaction` command to perform compaction on the cache data files. You must execute this command for each EADS server.

```
eztool compaction
```

(c) Forcibly terminate the active and isolated EADS servers

Execute the `eztool forcestop` command to forcibly terminate the active and isolated EADS servers.

If all EADS servers have already stopped, there is no need to execute this command.

Execute the `eztool forcestop` command on the active and isolated EADS servers.

```
eztool forcestop
```

(d) Check error messages

Check the error messages output to the message logs of the EADS servers that you identified in (a) above.

(e) Acquire error information

Acquire error information for all EADS servers.

For details about this procedure, see *12.2.1(4) Acquire error information*.

(f) Start all EADS servers in the cluster (resume disk caches)

After resolving the errors, start all EADS servers in the cluster and resume disk caches.

For details about this procedure, see *10.3.2 How to start the EADS servers (resuming caches on disk)*.

Reference note

If caches cannot be resumed because of corrupted cache files, determine the number of EADS servers whose cache files have become corrupted.

- When the number of EADS servers whose cache files have become corrupted is fewer than the number of redundant copies of data plus the original
  Execute the `deleteecf -l` command on the EADS servers whose files have become corrupted to delete all cache files from the caches that contain the corrupted files. Then re-execute the `eztool resume` command.
- When the number of EADS servers whose cache files are corrupted is equal to or greater than the number of redundant copies of data plus the original
  The EADS servers cannot be restored.

12.2.3 If a poor response was reported

If a poor response was reported from an administrator of a business application program, or if application program processing timed out due to a poor response, acquire error information to investigate the cause on the EADS servers and EADS clients.

(1) Error information to acquire on the EADS servers

Acquire the following information on all EADS servers:

- Statistics
- Thread dumps
- All files under the directory specified in the `eads.logger.dir` parameter in the server properties
- All property files under *management-directory*/`conf`

For details about how to acquire error information, see *12.3 Acquiring error information*.

(2) Error information to acquire on the EADS clients

Acquire the following information on all EADS clients:

- All files under the directory specified in the `eads.client.logger.dir` parameter in the client properties
- Client property file

(3) Items to check with the administrator of business applications

Check the following with the administrator of the business application program:

- Is there any regularity in the time period in which the response has become poor?
- Has a poor response occurred repeatedly on the same EADS client?
- Is there any regularity in the sorts of operations that result in a poor response?

12.3 Acquiring error information

This section explains the error information needed for root cause investigation and how to obtain it.

12.3.1 Error information needed for root cause investigation

To investigate the cause of the problem, you need the error information listed below. In the event of a failure, the system operation administrator must obtain this error information.

You use the `eztool snapshot` command to collect logs and property files in a single batch operation.

- Message logs
  The message log files contain message logs for checking operations and monitoring errors.
- Statistics
  The statistics file contains statistics, including those used for re-evaluating tuning, measuring performance, and re-estimating resources.
- Thread dumps
  Thread dumps contain information about the threads running in Java processes.
  Obtain thread dumps as needed.
- Core dumps
  If an EADS server is shut down, a core dump is output immediately under the management directory. Its file name is core.[*PID*].[#]
  Make note of the available disk space because no limit is set for the size of core dumps output by EADS servers.
  #
  [*PID*]: EADS server's process ID

Also obtain the following property files in the event of a failure:

- All property files under *management-directory*/conf

12.3.2 Obtaining statistics

For details about how to obtain statistics, see *11.13 Obtaining statistics*.

12.3.3 Obtaining thread dumps

Use the `eztool threaddump` command to obtain thread dumps.

Command execution example

```
eztool threaddump
```

The thread dumps are output under the directory specified in the `eads.logger.dir` parameter in the server properties.

If the `eads.logger.dir` parameter is not specified in the server properties, the thread dumps are output under *management-directory*`/logs`.

13 Investigating the Causes of Failures

This chapter explains how to investigate the causes of failures (how to determine the source of a failure).

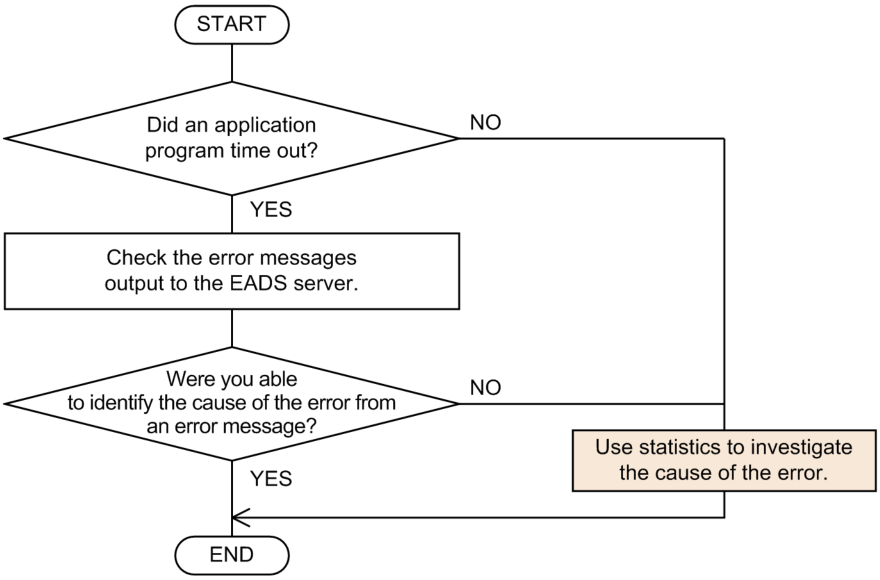
13.1 Investigating the cause of a poor response

This section explains how to investigate the cause of a poor response.

13.1.1 General investigation procedure

The following figure shows the general procedure for investigating the cause of a poor response, and an application program timeout due to a poor response.

Figure 13–1: General procedure for investigating the cause of a poor response and an application program timeout due to a poor response


START
Did an application program time out?
NO
YES
Check the error messages output to the EADS server.
Were you able to identify the cause of the error from an error message?
NO
YES
Use statistics to investigate the cause of the error.
END

Note: How to perform the investigation shown in the shaded rectangle is explained in a later section.

Examples of events

- A poor response was reported.
- Application program processing timed out due to a poor response.

Possible causes

- Accesses are concentrated on a specific EADS server.
- The size of the data is not suitable for storage.
- A non-EADS process is performing heavy processing.
- Requests are concentrated or have increased.

Angles of investigation using statistics

- Is the cause the EADS server, a user function, or the network?

  Identify the cause of the poor response from the average user function internal processing time and the average user program processing time in a user function.

Error information needed for the investigation

- All EADS servers' statistics

For details about how to obtain statistics, see *11.13 Obtaining statistics*.

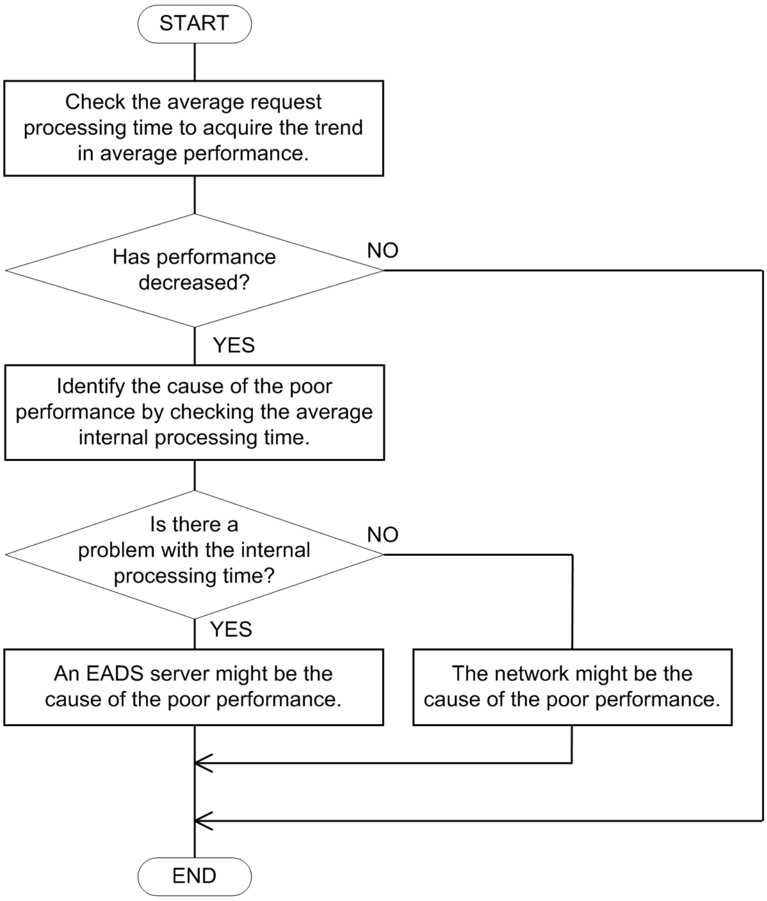
13.1.2 Investigating the cause

This subsection explains how to check the performance by using statistics.

(1) Check application program performance

The figure below shows how to use statistics to check application program performance. The detailed procedure is provided below.

Figure 13–2: General procedure for checking application program performance


START
Check the average request processing time to acquire the trend in average performance.
Has performance decreased?
NO
YES
Identify the cause of the poor performance by checking the average internal processing time.
Is there a problem with the internal processing time?
NO
YES
An EADS server might be the cause of the poor performance.
The network might be the cause of the poor performance.
END

(a) Check the trend in average performance

Check the trend in average performance from the following statistics items:

Statistics type	Item	Column name
Statistics (`eads_stats.csv`)	Average request processing time	`RequestProcessingAverageTime`

(b) Identify the cause of poor performance

Identify the possible causes of poor performance by checking the following statistics item:

Statistics type	Item	Column name
Statistics (`eads_stats.csv`)	Average internal processing time	`InternalProcessingAverageTime`

(c) Analyze the cause of poor performance

Analyze the possible causes of poor performance identified above.

■ If the EADS server is the cause

If the EADS server might be the cause, check the following:

- Determine if accesses to the EADS server are concentrated by checking the following statistics item:

Statistics type	Item	Column name
Statistics (eads_stats.csv)	Number of requests accepted (put, create, update, replace, get, remove)	RequestCount

- Check the Java logs for any garbage collection.
- Check the OS information (such as the CPU usage and memory usage)

■ If the network is the cause

If the network might be the cause, check the following:

- Determine the amount of communication by checking the following statistics items:

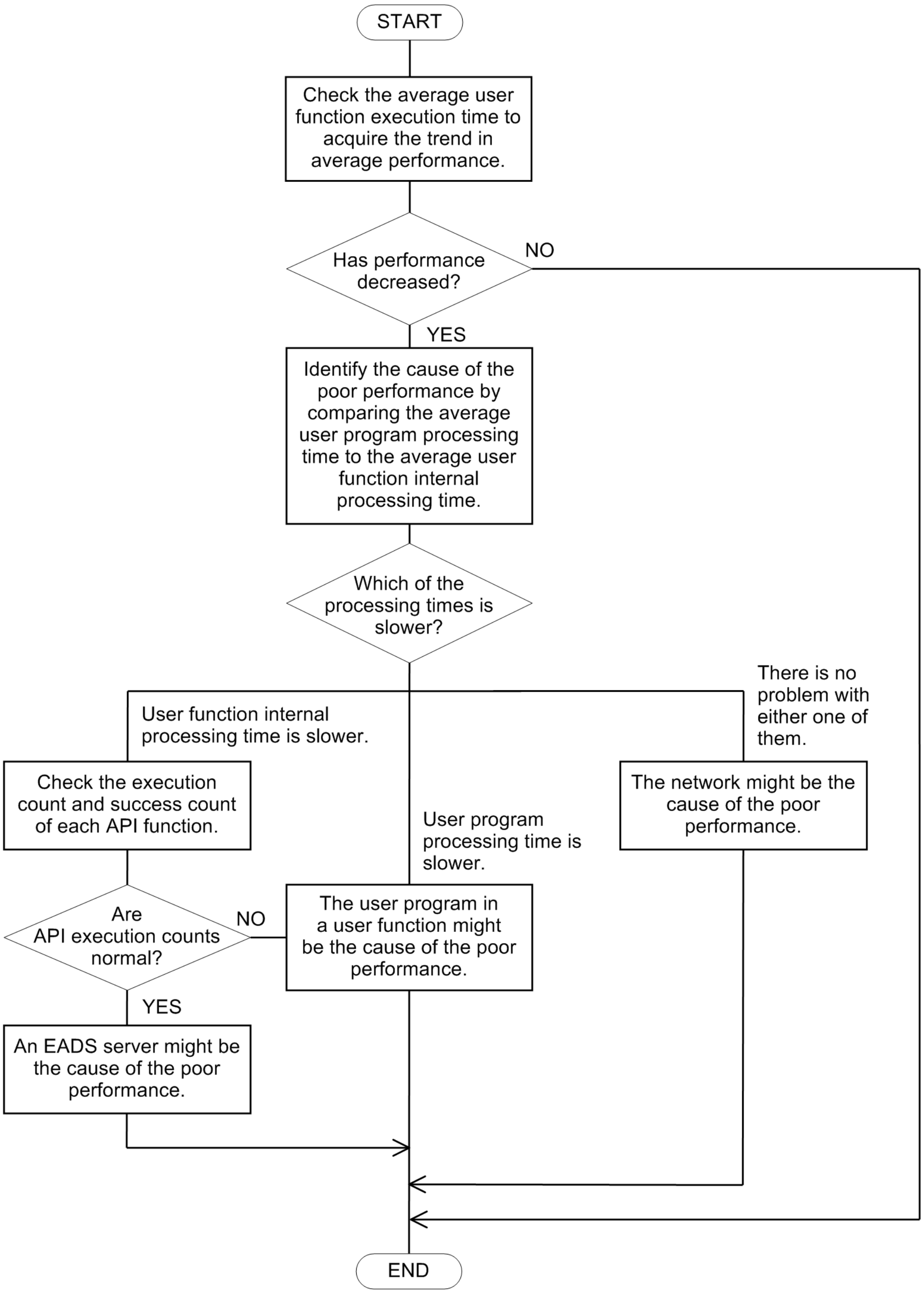
Statistics type	Item	Column name
Statistics (eads_stats.csv)	Maximum size of receive data between EADS client and EADS servers	CSReadMaxSize
	Total size of receive data between EADS client and EADS servers	CSReadTotalSize
	Maximum size of send data between EADS client and EADS servers	CSWriteMaxSize
	Total size of send data between EADS client and EADS servers	CSWriteTotalSize
	Maximum size of receive data between EADS servers	SSReadMaxSize
	Total size of receive data between EADS servers	SSReadTotalSize
	Maximum size of send data between EADS servers	SSWriteMaxSize
	Total size of send data between EADS servers	SSWriteTotalSize

- Check the network status.

(2) Checking the performance of user functions

The following figure shows the general procedure for checking the performance of user functions by using statistics. The detailed procedure is provided below.

Figure 13–3: General procedure for checking the performance of user functions


START
Check the average user function execution time to acquire the trend in average performance.
Has performance decreased?
NO
YES
Identify the cause of the poor performance by comparing the average user program processing time to the average user function internal processing time.
Which of the processing times is slower?
User function internal processing time is slower.
There is no problem with either one of them.
Check the execution count and success count of each API function.
User program processing time is slower.
The network might be the cause of the poor performance.
Are API execution counts normal?
NO
The user program in a user function might be the cause of the poor performance.
YES
An EADS server might be the cause of the poor performance.
END

(a) Check the trend in average performance

Check the trend in average performance from the following statistics item:

Statistics type	Item	Column name
User function statistics (eads_function_stats.csv)	Average user function execution time	FunctionExecuteAverageTime

(b) Identify the cause of poor performance

Identify the possible causes of poor performance by comparing the following statistics items:

Statistics type	Item	Column name
User function statistics (`eads_function_stats.csv`)	Average user function internal processing time	`FunctionInternalProcessingAverageTime`
	Average user program processing time in the user function	`UserProgramProcessingAverageTime`

(c) Analyze the cause of poor performance

Analyze the possible causes of poor performance identified above.

■ If the EADS server is the cause

If the EADS server might be the cause, check the following:

- Determine if accesses to the EADS server are concentrated by checking the following statistics item:

Statistics type	Item	Column name
User function statistics (`eads_function_stats.csv`)	Number of times the user function was executed	`FunctionExecuteCount`

- Check the Java logs for any garbage collection.
- Check the OS information (such as the CPU usage and memory usage)

■ If the user program in a user function is the cause

If the user program in a user function might be the cause, check each user function's statistics to determine which user function is the cause.

■ If the network is the cause

If the network might be the cause, check the following:

- Determine the amount of communication by checking the following statistics items:

Statistics type	Item	Column name
Statistics (`eads_stats.csv`)	Maximum size of receive data between EADS client and EADS servers	`CSReadMaxSize`
	Total size of receive data between EADS client and EADS servers	`CSReadTotalSize`
	Maximum size of send data between EADS client and EADS servers	`CSWriteMaxSize`
	Total size of send data between EADS client and EADS servers	`CSWriteTotalSize`
	Maximum size of receive data between EADS servers	`SSReadMaxSize`
	Total size of receive data between EADS servers	`SSReadTotalSize`
	Maximum size of send data between EADS servers	`SSWriteMaxSize`

Statistics type	Item	Column name
	Total size of send data between EADS servers	SSWriteTotalSize

- Check the network status.

14 Command Reference

This chapter explains the syntax of EADS commands.

14.1 Command storage location

The commands used in EADS are stored in the following directory:

management-directory`/bin`

Reference note

If you want to simplify the execution of commands, create a symbolic link to the management directory so that you can execute commands by using the link.

14.2 EADS commands

The following table lists and describes the EADS commands.

Table 14–1: List of EADS commands

No.	Usage scenario		Command name	Execution target
1	Startup	Starting an EADS server	`ezstart`	EADS server
2		Starting an EADS server in the foreground	`ezserver`	EADS server
3	Running	Running a cluster or an EADS server	`eztool`	Cluster or EADS server

Reference note

In this manual, a command option specified using only one character is shown with a single hyphen (`-`) and a command option specified using two or more characters is shown with two consecutive hyphens (`--`). The number of hyphens actually specified can be either one or two, regardless of the number of characters used. For example, the `-h` or `--help` option can be specified as `-h`, `--h`, `-help`, or `--help`.

14.2.1 ezstart (starts an EADS server)

(1) Description

This command starts an EADS server in the background.

(2) Rules

- This command can be executed when the EADS server is stopped.
- To execute this command, log in to the host on which you plan to start the EADS server. To start all EADS servers in the cluster, you must execute this command for each EADS server.
- The message logs output during command execution are not output to the console. They are output to `eads_start.log` under *management-directory*`/logs`.

(3) Format

```
ezstart [-h]
        [-r|-ai EADS-server-ID|-ap EADS-server-location-(hash-value)]
        [-rd port-number suspend-method]
```

(4) Options and arguments

(a) -h or --help

Specify this option to display the command's usage.

When this option is specified, any other options that are specified are ignored.

(b) -r or --recovery

Specify this option to restore a stopped EADS server into the cluster and set its cluster participation status to `online`.

(c) -ai EADS-server-ID or --add_id EADS-server-ID

Specify this option if you want to add a new EADS server to a specific range in the existing cluster configuration. Specify in this option the EADS server ID of the EADS server that currently manages the range into which the new EADS server is to be added.

An EADS server ID is an integer in the range from 1 to 96.

The new EADS server will be placed at the midpoint between the two EADS servers (hash values) adjacent to where the new EADS server will be located on the consistent hashing circumference. The smallest integer in the range from 1 to 96 that has not been assigned to any existing EADS server is assigned as the new EADS server's EADS server ID.

If none of the EADS servers in the cluster has the specified EADS server ID, an error results. An error also results if the cluster contains an isolated or stopped (`-----------`) EADS server when the command with this option specified is executed.

(d) -ap EADS-server-location-(hash-value) or --add_position EADS-server-location-(hash-value)

Specify this option if you want to add a new EADS server at a specific location (hash value) in the existing cluster configuration. Specify in this option the location (hash value) at which the new EADS server is to be added.

For the location (hash value), you can specify an integer in the range from `-2147483648` to `2147483647`.

The smallest integer in the range from 1 to 96 that has not been assigned to any existing EADS server is assigned as the new EADS server's EADS server ID.

If the specified location (hash value) is already occupied by an existing EADS server, an error results. An error also results if the cluster contains an isolated or stopped (`-----------`) EADS server when the command with this option specified is executed.

(e) -rd port-number suspend-method or --remotedebug port-number suspend-method

This option is used to remotely debug user functions by using a debugger application. Do not use this option in a production environment.

Use this option if you want to start the EADS server with the following java options specified:

```
-agentlib:jdwp=transport=dt_socket,server=y,suspend=suspend-
method,address=port-number
```

For the port number (used by the EADS server to connect to the debugger application), you can specify an integer in the range from `1024` to `65535`.

If the EADS server is to be started after it has been connected with the debugger application, specify `y` for *suspend-method*. If the EADS server is to be started without waiting to be connected with the debugger application, specify `n` for *suspend-method*.

(5) Return code

0: Normal termination

Other than 0: Error

(6) Notes

- To execute this command, you must first specify /bin and /usr/bin in the PATH environment variable.
- If you execute multiple instances of this command concurrently, eads_start.log is overwritten by the last process executed.
- If a number of the EADS servers are started in quick succession in a cluster that consists of many EADS servers, the processing will require some time to complete, and startup might time out. In such a case, start the EADS servers sequentially with enough of an interval between startups of the EADS servers. Alternatively, adjust the eads.admin.boot.timeout parameter value in the server properties.
- When you specify the -ai or --add_id option or the -ap or --add_position option, no cluster property file will be needed for the EADS server to be added. If the EADS server to be added already has a cluster property file, that cluster property file will not be imported. No error results if the EADS server to be added has a cluster property file with invalid contents.

14.2.2 ezserver (starts an EADS server in the foreground)

(1) Description

This command starts an EADS server in the foreground.

(2) Rules

- This command can be executed when the EADS server is stopped.
- To execute this command, log in to the host on which you plan to start the EADS server. To start all EADS servers in the cluster, you must execute this command for each EADS server.
- During command execution, the message logs are output to the console and to eads_server_message[*n*].log ([*n*] indicates the sequence number of the file) under the directory specified in the eads.logger.dir parameter in the server properties.

(3) Format

```
ezserver [-h]
         [-r|-ai EADS-server-ID|-ap EADS-server-location-(hash-value)]
         [-rd port-number suspend-method]
```

(4) Options and arguments

(a) -h or --help

Specify this option to display the command's usage.

When this option is specified, any other options that are specified are ignored.

(b) -r or --recovery

Specify this option to restore a stopped EADS server into the cluster and set its cluster participation status to `online`.

(c) -ai EADS-server-ID or --add_id EADS-server-ID

Specify this option if you want to add a new EADS server to a specific range in the existing cluster configuration. Specify in this option the EADS server ID of the EADS server that currently manages the range into which the new EADS server is to be added.

An EADS server ID is an integer in the range from 1 to 96.

The new EADS server will be placed at the midpoint between the two EADS servers (hash values) adjacent to where the new EADS server will be located on the consistent hashing circumference. The smallest integer in the range from 1 to 96 that has not been assigned to any existing EADS servers is assigned as the new EADS server's EADS server ID.

If none of the EADS servers in the cluster has the specified EADS server ID, an error results. An error also results if the cluster contains an isolated or stopped (`-----------`) EADS server when the command with this option specified is executed.

(d) -ap EADS-server-location-(hash-value) or --add_position EADS-server-location-(hash-value)

Specify this option if you want to add a new EADS server at a specific location (hash value) in the existing cluster configuration. Specify in this option the location (hash value) at which the new EADS server is to be added.

For the location (hash value), you can specify an integer in the range from `-2147483648` to `2147483647`.

The smallest integer in the range from 1 to 96 that has not been assigned to any existing EADS server is assigned as the new EADS server's EADS server ID.

If the specified location (hash value) is already occupied by an existing EADS server, an error results. An error also results if the cluster contains an isolated or stopped (`-----------`) EADS server when the command with this option specified is executed.

(e) -rd port-number suspend-method or --remotedebug port-number suspend-method

This option is used to remotely debug user functions by using a debugger application. Do not use this option in a production environment.

Use this option to start the EADS server with the following java options specified:

```
-agentlib:jdwp=transport=dt_socket,server=y,suspend=suspend-
method,address=port-number
```

For the port number (used by the EADS server to connect to the debugger application), you can specify an integer in the range from `1024` to `65535`.

If the EADS server is to be started after it has been connected with the debugger application, specify `y` for *suspend-method*. If the EADS server is to be started without waiting to be connected with the debugger application, specify `n` for *suspend-method*.

(5) Return code

`0`: Normal termination

Other than `0`: Error

(6) Notes

- To execute this command, you must first specify `/bin` and `/usr/bin` in the `PATH` environment variable.
- If a number of the EADS servers are started in quick succession in a cluster that consists of many EADS servers, the processing will require some time to complete, and startup might time out. In such a case, start the EADS servers sequentially with enough of an interval between startups of the EADS servers. Alternatively, adjust the `eads.admin.boot.timeout` parameter value in the server properties.
- If you execute this command for a running EADS server, the `KDEA08408-E` message is output and the processing is canceled. However, if two attempts to start the EADS server are made at almost the same time, the message might not be output. In that case, wait a while, and then try to start the EADS server again.
- When you specify the `-ai` or `--add_id` option or the `-ap` or `--add_position` option, no cluster property file will be needed for the EADS server to be added. If the EADS server to be added already has a cluster property file, that cluster property file will not be imported. No error results if the EADS server to be added has a cluster property file with invalid contents.

14.2.3 eztool (runs the cluster)

(1) Description

This command runs the cluster.

(2) Rules

- Execute this command on any EADS server in the cluster.
- Message logs are output during command execution to the console and to `eads_command_message[`*n*`].log` (`[`*n*`]`: file sequence number) under the directory specified in the `eads.command.logger.dir` parameter in the command properties.

(3) Format

```
eztool [-h]
       [-t command-timeout-value]
       [--messageoff]
       subcommand options-and-arguments
```

(4) Options and arguments

(a) -h or --help

Specify this option to display the command's usage.

When this option is specified, any other options that are specified are ignored.

If no subcommand is specified, an overview of all subcommands is displayed. If a subcommand is specified, the details about the specified subcommand are displayed.

(b) -t command-timeout-value or --timeout command-timeout-value

Specify this option if you want to set the command's timeout value (in seconds).

The permitted value is an integer from 0 through 2,147,483,647.

If zero is specified, no timeout occurs.

This option is not supported in the following commands:

- `eztool threaddump`
- `eztool snapshot`
- `eztool forcestop`

(c) --messageoff

Specify this option if messages are not to be output to the standard output.

If the command results in a syntax error, this option is ignored.

This option cannot be specified for the following commands:

- `eztool threaddump`
- `eztool snapshot`
- `eztool forcestop`

(d) subcommand options-and-arguments

For details about the subcommands that can be specified in this command, see *14.3 Subcommands of the eztool command*.

For details about the options and arguments that can be specified in the subcommands, see the explanation of each subcommand.

> **Important note**
>
> If you want to specify a character string beginning with a hyphen (`-`) in this subcommand's argument, specify two consecutive hyphens (`--`). Two consecutive hyphens alone cannot be specified as an argument.

(5) Return code

See the description of each subcommand.

(6) Notes

To execute this command, you must first specify `/bin` and `/usr/bin` in the `PATH` environment variable.

14.3 Subcommands of the eztool command

The following table lists and describes the subcommands that can be specified in the `eztool` command.

Table 14–2: List of subcommands that can be specified in the eztool command

No.	Usage scenario		Subcommand name	Execution target
1	Closing and opening	Closing the cluster	`close`	Cluster
2		Opening the cluster	`open`	Cluster
3	Checking status	Checking the status of the cluster	`status`	Cluster#1
4		Displaying a list of the most recent parameters	`listconf`	Cluster#1
5		Displaying a list of caches	`listcache`	Cluster
6		Displaying a list of store data files	`listesd`	Cluster
7		Displaying a list of group names	`listgroup`	Cluster#1
8		Displaying a list of keys	`listkey`	Cluster#1, #2
9		Displaying data storage locations	`getposition`	Cluster
10		Checking the usage status of ranges and caches	`storeusage`	Cluster
11	Locking	Unlocking	`unlock`	Cluster
12	Memory and disk management	Creating a cache	`createcache`	Cluster
13		Deleting a cache	`deletecache`	Cluster
14	Data migration	Exporting data	`export`	Cluster#1
15		Importing data	`import`	Cluster
16		Deleting store data files	`deleteesd`	Cluster
17	Data manipulation	Storing specified data	`put`	EADS server
18		Acquiring specified data	`get`	EADS server
19		Deleting specified data	`remove`	EADS server
20		Deleting all data in a specified range	`removeall`	Cluster#1, #2
21	User function	Displaying which user functions are executable	`listfunc`	Cluster
22		Executing user functions	`execfunc`	Cluster#1, #2, #3
23	Persistence	Displaying a list of information about persistent data	`listecf`	Cluster#1
24		Resuming caches	`resume`	Cluster
25		Importing cache files	`importecf`	Cluster
26		Deleting cache files	`deleteecf`	Cluster
27		Performing compaction on cache data files	`compaction`	EADS server#4
28	Troubleshooting	Outputting a thread dump	`threaddump`	EADS server

No.	Usage scenario		Subcommand name	Execution target
29		Collecting logs, settings, hardware information, and network information	`snapshot`	EADS server
30	Termination	Terminating all EADS servers in the cluster	`stop`	Cluster
31		Forcibly terminating an EADS server	`forcestop`	EADS server
32		Isolating an EADS server	`isolate`	EADS server

#1

If the `-s` or `--single` option is specified, only the EADS server on which the command is executed is the target.

#2

If you specify the `-g` or `--group` option, the target EADS server is determined from the hash value of the specified group name.

#3

If the `-k` or `--key` option is specified, the EADS server storing the specified key is the target EADS server.

#4

If the `--cache` option is specified, the cache used to execute the command is the target.

If the `--range` option is specified, the range used to execute the command is the target.

14.3.1 Locking between commands

The following table shows whether commands can be executed simultaneously.

Table 14–3: Whether commands can be executed simultaneously

Command type	Whether commands can be executed simultaneously			
	Updating	Referencing	Data manipulation	Restoration and scale-out
Updating • `eztool close` • `eztool open` • `eztool createcache` • `eztool deletecache` • `eztool export` • `eztool import` • `eztool deleteesd` • `eztool resume` • `eztool importecf` • `eztool deleteecf` • `eztool snapshot -sd` or `eztool snapshot --safedump` • `eztool stop` • `eztool isolate`	N	Y	Y	N
Referencing • `eztool status`	Y	Y	Y	Y

Command type	Whether commands can be executed simultaneously			
	Updating	Referencing	Data manipulation	Restoration and scale-out
• eztool listconf • eztool listcache • eztool listesd • eztool listecf • eztool getposition • eztool storeusage • eztool listfunc				
Data manipulation • eztool listgroup • eztool listkey • eztool put • eztool get • eztool remove • eztool removeall • eztool execfunc	Y	Y	Y	Y
Restoration and scale-out • ezstart -r or ezstart --recovery • ezserver -r or ezserver --recovery • ezstart -ai or ezstart --add_id • ezstart -ap or ezstart --add_position • ezserver -ai or ezserver --add_id • ezserver -ap or ezserver --add_position	N	Y	Y	N

Legend:

Y: Can be executed simultaneously

N: Cannot be executed simultaneously

Notes:

The `eztool unlock` command can be executed only when a lock can be released.

The following commands can always be executed simultaneously:

- `eztool compaction`
- `eztool threaddump`
- `eztool snapshot` (except when the `-sd` or `--safedump` option is specified)
- `eztool forcestop`

When an updating command is executed, a lock is obtained from the EADS server. This prevents another updating command from being executed simultaneously by another host.

Normally, a lock is released automatically when the command's processing terminates. However, if unlocking fails for some reason (such as when an error occurs and the command terminates before the lock is released), execute the `eztool unlock` command to unlock.

14.3.2 close (closes the cluster)

(1) Description

This subcommand closes the cluster.

(2) Rules

- This subcommand can be executed only when the status of the cluster is one of the following:
  - Cluster available (`AVAILABLE`)
  - Cluster partially available (`PARTIALLY_AVAILABLE`)
- The target of this subcommand is the EADS servers whose cluster participation status is `online`. This subcommand cannot be executed if the cluster contains any EADS server whose cluster participation status is `standby`. You can determine the cluster participation status with the `eztool status` command.
- This subcommand can be executed when the target EADS servers are in the following status:
  - Running
  - Closed

(3) Format

```
eztool close
```

(4) Return code

The following table lists the return codes that this subcommand returns.

Table 14–4: Return codes returned by the eztool close command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	0	0	Command execution was successful.
2	101	101	Initialization of the command failed.
3	110		Connection establishment failed.
4	111		The command failed due to a communication timeout.
5	120		The command failed due to a syntax error.
6	130		The command failed because it could not be executed.

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
7	131		The command failed because another command was executing.
8	150		The command failed during execution.
9	200		The command failed due to a timeout.

14.3.3 open (opens the cluster)

(1) Description

This subcommand opens the cluster.

(2) Rules

- This subcommand can be executed only when the status of the cluster is one of the following:
    - Cluster available (`AVAILABLE`)
    - Cluster partially available (`PARTIALLY_AVAILABLE`)
- The target of this subcommand is the EADS servers whose cluster participation status is `online`. This subcommand cannot be executed if the cluster contains any EADS server whose cluster participation status is `standby`. You can determine the cluster participation status with the `eztool status` command.
- This subcommand can be executed when the target EADS servers are in the following status:
    - Initialized
    - Running
    - Closed

(3) Format

```
eztool open
```

(4) Return code

The following table lists the return codes that this subcommand returns.

Table 14–5: Return codes returned by the eztool open command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	0	0	Command execution was successful.

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
2	`101`	`101`	Initialization of the command failed.
3	`110`		Connection establishment failed.
4	`111`		The command failed due to a communication timeout.
5	`120`		The command failed due to a syntax error.
6	`130`		The command failed because it could not be executed.
7	`131`		The command failed because another command was executing.
8	`150`		The command failed during execution.
9	`200`		The command failed due to a timeout.

14.3.4 status (checks the status of the cluster)

(1) Description

This subcommand checks the status of the cluster.

(2) Rules

- This subcommand can be executed regardless of the cluster's status.
- This subcommand can be executed when the EADS servers are in the following status:
  - Initializing
  - Initialized
  - Running
  - Closing
  - Closed
  - Isolated
  - Stopping

(3) Format

```
eztool status [-v]
              [-s]
              [-c display-item-name==status]
              [--format format-name]
              [--columns column-name[,column-name]...]
              [--filter filter-condition]
              [--match matching-condition]
```

(4) Options and arguments

(a) -v or --verbose

Specify this option if you want to display the details of the command execution results.

(b) -s or --single

Specify this option to check the status of only the EADS server on which the command is executed, not the status of the entire cluster.

(c) -c display-item-name==status or --count display-item-name==status

We do not recommend that you specify this option. This option might be deleted without notice.

Specify this option to verify that all EADS servers participating in the cluster have the same status.

The subcommand sets as the return code the number of EADS servers whose status exactly matches the specified status. Note that items for which a hyphen (-) is displayed because the information could not be acquired are not counted.

This option cannot be specified together with any of the following options:

- `-s` option
- `--format` option
- `--columns` option
- `--filter` option
- `--match` option

■ Display item names

The following table lists and describes the display item names that can be specified:

Display item name	Description	Whether displayed	
		-v or --verbose option	
		Omitted	Specified
`ID`	EADS server ID	Y	Y
`IP_Address`	IP address of EADS server	Y	Y
`ServerName`	EADS server name (management directory name)	N	Y
`Port`[#1]	EADS server's port number used for communication with EADS clients	Y	Y
`ServerPort`[#2]	Port number used for creating redundant copies of data among the EADS servers	N	Y
`ManagePort`[#2]	Port number used for checking the communication port that will be used by the command	N	Y
`Position`	EADS server position	Y	Y
`Cluster`	Cluster participation status	Y	Y
`State`	EADS server status	Y	Y
`Operation`	Name of the subcommand or option of the command that is currently executing	Y	Y

Display item name	Description	Whether displayed	
		-v or --verbose option	
		Omitted	Specified
`Lock`	Lock status of the EADS server	N	Y
`KeyCount`	Total number of keys in the EADS server	N	Y
`UsedCache`#2	Memory usage rate in the area for storing the value part of the memory caches or two-way caches (explicit heap) on the EADS server (*memory usage amount* divided by *total memory capacity*)	N	Y
`UsedMemoryRatio`#3	Memory usage rate in the area for storing the value part of the memory caches or two-way caches (explicit heap) on the EADS server	N	Y
`UsedMemory`#3	Memory usage amount in the area for storing the value part of the memory caches or two-way caches (explicit heap) on the EADS server (megabytes)	N	Y
`MaxMemory`#3	Total memory capacity of the area for storing the value part of the memory caches or two-way caches on the EADS server (megabytes)	N	Y
`Version`	Version information	N	Y

Legend:

Y: Displayed

N: Not displayed

#1

If `0300` is specified in the `eads.command.compat` parameter in the command properties, the item name is `ClientPort`.

#2

This item can be specified only when `0300` is specified in the `eads.command.compat` parameter in the command properties.

#3

This item cannot be specified if `0300` is specified in the `eads.command.compat` parameter in the command properties.

■ Status

Specify a character string that can be displayed for the specified displayed item name.

The permitted characters are ASCII codes `0x20` through `0x7E`.

You can specify a single-byte space by enclosing it in double quotation marks (`"`). Any single byte spaces before or after a display item name in the table or before or after the status are ignored.

Because a hyphen (`-`) is displayed for an item whose information cannot be obtained, specifying only a hyphen (`-`) will result in an error. If `0300` is specified in the `eads.command.compat` parameter in the command properties, an asterisk (`*`) is displayed for an item whose information cannot be obtained. In this case, specifying only an asterisk (`*`) results in an error.

(d) --format format-name

For details about this option, see *14.4.2 How to specify the display format*.

(e) --columns column-name[,column-name]...

For details about this option, see *14.4.3 How to specify column filters*.

(f) --filter filter-condition

For details about this option, see *14.4.4 How to specify row filters*.

(g) --match matching-condition

For details about this option, see *14.4.5 How to specify a condition match*.

(5) Output example

The following shows output examples of the `eztool status` command's execution results.

For details about the components of the displayed information, see *14.4.1 Components of the displayed information*.

■ If option `-v` or `--verbose` is omitted

```
$ eztool status
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address       Port   Position     Cluster  State    Operation
 1  XX.XXX.XXX.168  24600   1288490189  online   running  none
 2  XX.XXX.XXX.168  24700    429496730  online   running  none
 3  XX.XXX.XXX.168  24800   -429496729  online   running  none
 4  XX.XXX.XXX.168  24900  -1288490188  online   running  none
 5  XX.XXX.XXX.168  25000  -2147483648  online   running  none
--------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

■ If option `-v` or `--verbose` is specified

```
$ eztool status -v
KDEA08001-I           The command will now start. (subcommand = status, parameter = [status, -v])

Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address      ServerName  Port   Position     Cluster  State    Operation  Lock    KeyCount  UsedMemoryRatio  UsedMemory  MaxMemory  Version
 1  XX.XXX.XXX.168  server01    24600   1288490189  online   running  none       unlock       582               13           3         23  04-00-00
 2  XX.XXX.XXX.168  server02    24700    429496730  online   running  none       unlock       618               13           3         23  04-00-00
 3  XX.XXX.XXX.168  server03    24800   -429496729  online   running  none       unlock       636               13           3         23  04-00-00
 4  XX.XXX.XXX.168  server04    24900  -1288490188  online   running  none       unlock       591               13           3         23  04-00-00
 5  XX.XXX.XXX.168  server05    25000  -2147483648  online   running  none       unlock       573               13           3         23  04-00-00
-----------------------------------------------------------------------------------------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

The following tables list and describe the summary and content information that are displayed.

Table 14–6: Summary information displayed by the eztool status command

No.	Summary name	Description	Whether displayed	
			-v or --verbose option	
			Omitted	Specified
1	`Cluster Health`	Cluster status	Y	Y

No.	Summary name	Description	Whether displayed	
			-v or --verbose option	
			Omitted	Specified
		One of the following is displayed: • `AVAILABLE` The cluster is running normally. • `PARTIALLY_AVAILABLE` The cluster is partially running. The cluster might not be accessible depending on the key. • `NOT_AVAILABLE` The cluster is not running.		
2	`TotalCount`	Number of EADS servers This item is not displayed if the `-s` or `--single` option is specified.	Y	Y
3	`OnlineCount`	Number of EADS servers that are participating in the cluster This item is not displayed if the `-s` or `--single` option is specified. This item is not counted if the cluster status is `NOT_AVAILABLE`.	Y	Y
4	`OfflineCount`	Number of EADS servers that are not participating in the cluster This item is not displayed if the `-s` or `--single` option is specified. This item is not counted if the cluster status is `NOT_AVAILABLE`.	Y	Y
5	`StandbyCount`	Number of EADS servers that are designated to participate in the cluster This item is not displayed if the `-s` or `--single` option is specified. This item is not counted if the cluster status is `NOT_AVAILABLE`.	Y	Y

Legend:

Y: Displayed

Table 14–7: Content information displayed by the eztool status command

No.	Column name	Description	Whether displayed	
			-v or --verbose option	
			Omitted	Specified
1	`ID`	EADS server ID. If an EADS server to be added to the cluster has not participated in the cluster, a hyphen (-) is displayed as its EADS server ID.	Y	Y
2	`IP_Address`	IP address of EADS server	Y	Y
3	`ServerName`	EADS server name (management directory name)[#1, #2]	N	Y

No.	Column name	Description	Whether displayed	
			-v or --verbose option	
			Omitted	Specified
4	`Port`[#3]	EADS server's port number used for communication with EADS clients	Y	Y
5	`ServerPort`[#4]	Port number used for creating redundant copies of data among the EADS servers[#2]	N	Y
6	`ManagePort`[#4]	Port number used for checking the communication port that will be used by the command[#2]	N	Y
7	`Position`	EADS server position The positions (hash values) are displayed in descending order. If an EADS server to be added to the cluster has not participated in the cluster, a hyphen (`-`) is displayed as its position.	Y	Y
8	`Cluster`	Cluster participation status.[#5] One of the following is displayed: • `online` Participating in the cluster. • `offline` Not participating in the cluster. • `standby` Designated to participate in the cluster.	Y	Y
9	`State`	EADS server status.[#1, #2]	Y	Y
10	`Operation`	Name of the subcommand or option of the command that is currently executing[#1, #2] One of the following is displayed: • `close` • `open` • `unlock` • `createcache` • `deletecache` • `export` • `import` • `deleteesd` • `resume` • `importecf` • `deleteecf` • `compaction` • `stop` • `isolate` • `recovery` • `add` For `export`, `import`, `importecf`, and `stop`, the progress of the processing is also displayed. Example: `export(88%)` If the command is not executing, `none` is displayed.	Y	Y
11	`Lock`	Lock status of the EADS server.[#5]	N	Y

No.	Column name	Description	Whether displayed	
			-v or --verbose option	
			Omitted	Specified
		One of the following is displayed: • `lock` Locked. • `unlock` Not locked.		
12	`KeyCount`	Total number of keys in the EADS server.#1, #2 This value includes the number of keys copied for data redundancy purposes.	N	Y
13	`UsedCache`#4	Memory usage rate in the area for storing the value part of the memory caches or two-way caches (explicit heap) on the EADS server (*memory usage amount* divided *total memory capacity*)#1, #2 • The digits after the decimal point are truncated. • The memory usage amount includes the values that were copied for data redundancy purposes.	N	Y
14	`UsedMemoryRatio`#6	Memory usage rate in the area for storing the value part of the memory caches or two-way caches (explicit heap) on the EADS server#1, #2 • Because the memory usage rate equals (*memory usage amount* ÷ *total memory capacity*) × 100 (%), the digits after the decimal point are truncated. • The memory usage amount includes the values that were copied for data redundancy purposes.	N	Y
15	`UsedMemory`#6	Memory usage amount in the area for storing the value part of the memory caches or two-way caches (explicit heap) on the EADS server#1, #2 (megabytes) • The digits after the decimal point are truncated. • The memory usage amount includes the values that were copied for data redundancy purposes.	N	Y
16	`MaxMemory`#6	Total memory capacity of the area for storing the value part of the memory caches or two-way caches on the EADS server#1, #2 (megabytes) • The digits after the decimal point are truncated. • The memory usage amount includes the values that were copied for data redundancy purposes.	N	Y
17	`Version`	Version information#1, #2 For 03-00, `03-00-00` is displayed.	N	Y

Legend:

Y: Displayed

N: Not displayed

#1

In the case of a connection failure, communication error, or communication timeout, a hyphen (`-`) is displayed. If `0300` is specified in the `eads.command.compat` parameter in the command properties, an asterisk (`*`) is displayed.

#2

If the EADS server's cluster participation status is `standby`, a hyphen (`-`) is displayed. If `0300` is specified in the `eads.command.compat` parameter in the command properties, an asterisk (`*`) is displayed.

#3

If `0300` is specified in the `eads.command.compat` parameter in the command properties, `ClientPort` is displayed as the column name.

#4

The column is displayed only when `0300` is specified in the `eads.command.compat` parameter in the command properties.

#5

If the cluster status is `NOT_AVAILABLE`, a hyphen (`-`) is displayed. If `0300` is specified in the `eads.command.compat` parameter in the command properties, an asterisk (`*`) is displayed.

#6

The column is not displayed if `0300` is specified in the `eads.command.compat` parameter in the command properties.

(6) Return code

The return code depends on whether any of the following options was specified:

- `-s` or `--single` option
- `-c` or `--count` option
- `--match` option

If none of the `-s`, `-c`, and `--match` options was specified, the return codes listed in the following table are returned.

Table 14–8: Return codes returned by the eztool status command (when none of the -s, -c, and --match options was specified)

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`110`		Connection establishment failed.
4	`111`		The command failed due to a communication timeout.
5	`120`		The command failed due to a syntax error.
6	`150`		The command failed during execution.
7	`200`		The command failed due to a timeout.

When the `-s` or `--single` option is specified:

The subcommand sets the EADS server's status as the return code.

If the --match option was also specified and the command's execution was successful, the results of the --match option take precedence.

Table 14–9: Return codes returned by the eztool status command (when the -s or --single option was specified)

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	0	0	The EADS server is running (running).
2	1	1	The EADS server is being initialized (initializing).
3	2	2	The EADS server has been initialized (initialized).
4	3	3	The EADS server is closing (closing).
5	4	4	The EADS server is closed (closed).
6	5	5	The EADS server is stopping (stopping).
7	10	10	The EADS server is isolated (isolated).
8	101	101	Initialization of the command failed.
9	110	110	Connection establishment failed.
10	111	111	The command failed due to a communication timeout.
11	120	101	The command failed due to a syntax error.
12	150		The command failed during execution.
13	200		The command failed due to a timeout.

When the -c or --count option is specified:

The subcommand sets as the return code the number of EADS servers for which the specified item is in the specified status. If the command's execution failed, the return code is the same as when the -c or --count option was not specified.

When the --match option is specified:

If the condition was satisfied, the subcommand returns 0; otherwise, the subcommand returns 1. If the command's execution failed, the return code is the same as when the --match option was not specified.

(7) Notes

- If the EADS server with the smallest EADS server ID is not running, no other EADS servers can participate in the cluster. If this command is executed on an EADS server that is not participating in the cluster, information about only the EADS server on which the command was executed is displayed even when several other EADS servers are running.
- If this command is executed on an EADS server for which restoration processing is underway, information about other EADS servers is not displayed until the EADS server has been restored.
- If this subcommand is executed on an EADS server that is engaged in scale-out processing, information about the other EADS servers is not displayed until this EADS server participates in the cluster.

- If there are only isolated EADS servers, outdated information might be displayed because the cluster information has not been updated.
- If the command times out during output processing, the output results might not be complete. The command might time out even if output processing is complete. If this happens, increase the command's timeout value, and then re-execute the command.

14.3.5 listconf (displays a list of most recent parameters)

(1) Description

This subcommand displays a list of parameters that begin with `eads.` and are specified in the EADS server's system properties. For any parameter that has been set for version 03-60 or earlier, the new parameter name is displayed.

(2) Rules

- This subcommand can be executed on an EADS server that is in any of the following statuses, regardless of the cluster's status:
  - Initializing
  - Initialized
  - Running
  - Closing
  - Closed
  - Isolated
  - Stopping

(3) Format

```
eztool listconf [-v]
                [-s]
                [--file property-file-type]
                [--non_default]
                [--format format-name]
                [--columns column-name[,column-name]...]
                [--filter filter-condition]
                [--match matching-condition]
```

(4) Options and arguments

(a) -v or --verbose

Specify this option if you want to display the details of the command's execution results.

(b) -s or --single

Specify this option to check the parameters in the property file for only the EADS server on which the command is executed, not for the entire cluster.

(c) --file property-file-type

Specify this option to check only those parameters that are specified in a specific property file.

You can specify any of the following types of property files:

- `server`: Server property file
- `cluster`: Cluster property file
- `shared`: Shared property file

If this option is omitted, the parameters that are specified in all of the following property files are displayed:

- Server property file
- Cluster property file
- Shared property file

(d) --non_default

This option displays for the EADS servers only the parameters whose settings are not the default values. Parameters for which there is no default value are not subject to this option.

(e) --format format-name

For details about this option, see *14.4.2 How to specify the display format*.

(f) --columns column-name[,column-name]...

For details about this option, see *14.4.3 How to specify column filters*.

(g) --filter filter-condition

For details about this option, see *14.4.4 How to specify row filters*.

(h) --match matching-condition

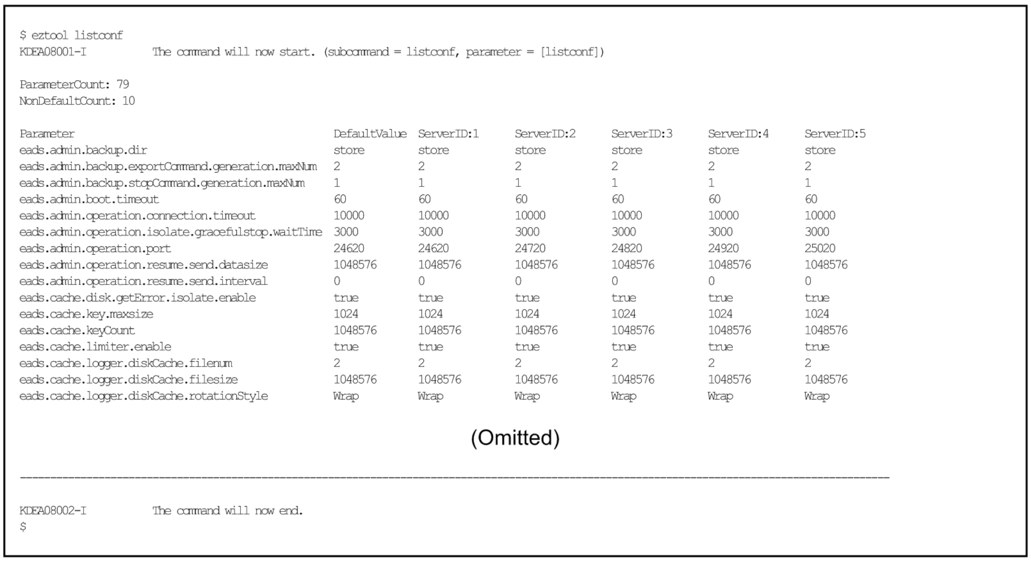
For details about this option, see *14.4.5 How to specify a condition match*.

(5) Output example

The following shows output examples of the `eztool listconf` command's execution results.

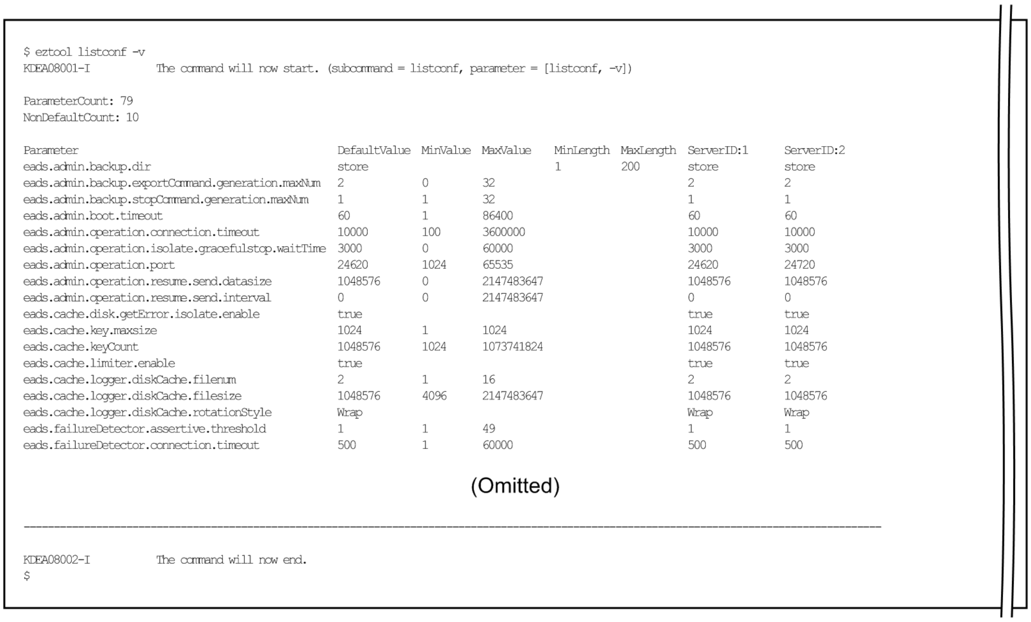
For details about the components of the displayed information, see *14.4.1 Components of the displayed information*.

■ If option `-v` or `--verbose` is omitted



```
$ eztool listconf
KDEA08001-I         The command will now start. (subcommand = listconf, parameter = [listconf])

ParameterCount: 79
NonDefaultCount: 10

Parameter                                        DefaultValue ServerID:1    ServerID:2    ServerID:3    ServerID:4    ServerID:5
eads.admin.backup.dir                            store        store         store         store         store         store
eads.admin.backup.exportCommand.generation.maxNum 2           2             2             2             2             2
eads.admin.backup.stopCommand.generation.maxNum  1            1             1             1             1             1
eads.admin.boot.timeout                          60           60            60            60            60            60
eads.admin.operation.connection.timeout          10000        10000         10000         10000         10000         10000
eads.admin.operation.isolate.gracefulstop.waitTime 3000       3000          3000          3000          3000          3000
eads.admin.operation.port                        24620        24620         24720         24820         24920         25020
eads.admin.operation.resume.send.datasize        1048576      1048576       1048576       1048576       1048576       1048576
eads.admin.operation.resume.send.interval        0            0             0             0             0             0
eads.cache.disk.getError.isolate.enable          true         true          true          true          true          true
eads.cache.key.maxsize                           1024         1024          1024          1024          1024          1024
eads.cache.keyCount                              1048576      1048576       1048576       1048576       1048576       1048576
eads.cache.limiter.enable                        true         true          true          true          true          true
eads.cache.logger.diskCache.filenum              2            2             2             2             2             2
eads.cache.logger.diskCache.filesize             1048576      1048576       1048576       1048576       1048576       1048576
eads.cache.logger.diskCache.rotationStyle        Wrap         Wrap          Wrap          Wrap          Wrap          Wrap

                                          (Omitted)

----------------------------------------------------------------------------------------------------------------------------------

KDEA08002-I         The command will now end.
$
```

■ If option `-v` or `--verbose` is specified



```
$ eztool listconf -v
KDEA08001-I         The command will now start. (subcommand = listconf, parameter = [listconf, -v])

ParameterCount: 79
NonDefaultCount: 10

Parameter                                          DefaultValue MinValue MaxValue    MinLength MaxLength ServerID:1    ServerID:2
eads.admin.backup.dir                              store                             1         200       store         store
eads.admin.backup.exportCommand.generation.maxNum  2            0        32                              2             2
eads.admin.backup.stopCommand.generation.maxNum    1            1        32                              1             1
eads.admin.boot.timeout                            60           1        86400                           60            60
eads.admin.operation.connection.timeout            10000        100      3600000                         10000         10000
eads.admin.operation.isolate.gracefulstop.waitTime 3000         0        60000                           3000          3000
eads.admin.operation.port                          24620        1024     65535                           24620         24720
eads.admin.operation.resume.send.datasize          1048576      0        2147483647                      1048576       1048576
eads.admin.operation.resume.send.interval          0            0        2147483647                      0             0
eads.cache.disk.getError.isolate.enable            true                                                  true          true
eads.cache.key.maxsize                             1024         1        1024                            1024          1024
eads.cache.keyCount                                1048576      1024     1073741824                      1048576       1048576
eads.cache.limiter.enable                          true                                                  true          true
eads.cache.logger.diskCache.filenum                2            1        16                              2             2
eads.cache.logger.diskCache.filesize               1048576      4096     2147483647                      1048576       1048576
eads.cache.logger.diskCache.rotationStyle          Wrap                                                  Wrap          Wrap
eads.failureDetector.assertive.threshold           1            1        49                              1             1
eads.failureDetector.connection.timeout            500          1        60000                           500           500

                                          (Omitted)

----------------------------------------------------------------------------------------------------------------------------------

KDEA08002-I         The command will now end.
$
```

The following tables list and describe the summary and content information that are displayed.

Table 14–10: Summary information displayed by the eztool listconf command

No.	Summary name	Description	Whether displayed	
			-v or --verbose option	
			Omitted	Specified
1	`ParameterCount`	Number of parameters.	Y	Y
2	`NonDefaultCount`	Number of EADS server parameters whose settings are not the default value	Y	Y

Legend:

Y: Displayed

Table 14–11: Content information displayed by the eztool listconf command

No.	Column name	Description	Whether displayed	
			-v or --verbose option	
			Omitted	Specified
1	`Parameter`	Parameter name. Displayed in natural order.	Y	Y
2	`DefaultValue`	Parameter's default value. Nothing is displayed for a parameter that does not have a default value. If the default value varies from one EADS server to another, `UNMATCH` is displayed.	Y	Y
3	`MinValue`	Minimum value of the parameter (numeric value). Nothing is displayed for a parameter in which a nonnumeric value can be specified. If the minimum value varies from one EADS server to another, `UNMATCH` is displayed.	N	Y
4	`MaxValue`	Maximum value of the parameter (numeric value). Nothing is displayed for a parameter in which a nonnumeric value can be specified. If the maximum value varies from one EADS server to another, `UNMATCH` is displayed.	N	Y
5	`MinLength`	Minimum length for a character string in the parameter. Nothing is displayed for a parameter that does not have a minimum length. If the minimum length varies from one EADS server to another, `UNMATCH` is displayed.	N	Y
6	`MaxLength`	Maximum length for a character string in the parameter. Nothing is displayed for a parameter that does not have a maximum length. If the maximum length varies from one EADS server to another, `UNMATCH` is displayed.	N	Y
7	`ServerID:`*EADS-server-ID*	Parameter's setting on each EADS server. Information is displayed for all EADS servers for which information was acquired, in ascending order of the EADS server IDs. If the `-s` or `--single` option is specified, information about only the EADS server on which the command was executed is displayed. If an EADS server ID is undetermined because an EADS server to be added to the cluster has not participated in the cluster, a hyphen (`-`) is displayed for *EADS-server-ID*.	Y	Y

Legend:

Y: Displayed

N: Not displayed

(6) Return code

The following table lists the return codes that this subcommand returns.

Table 14–12: Return codes returned by the eztool listconf command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`110`		Connection establishment failed.
4	`111`		The command failed due to a communication timeout.
5	`120`		The command failed due to a syntax error.
6	`130`		The command failed because it could not be executed.
7	`150`		The command failed during execution.
8	`200`		The command failed due to a timeout.

When the `--match` option is specified:

If the condition was satisfied, the subcommand returns `0`; otherwise, the subcommand returns `1`. If the command's execution failed, the return code is the same as when the `--match` option is not specified.

(7) Notes

- If the command times out during output processing, the output results might not be complete. The command might time out even if output processing is complete. If this happens, increase the command's timeout value, and then re-execute the command.
- The values set in system properties are the verified values, not the specified values themselves. When parameter values are used as functions, different values might be used for some parameters.

14.3.6 listcache (displays a list of caches)

(1) Description

This subcommand displays a list of caches.

(2) Rules

- This subcommand can be executed only when the status of the cluster is one of the following:
  - Cluster available (`AVAILABLE`)
  - Cluster partially available (`PARTIALLY_AVAILABLE`)
- The target of this subcommand is the EADS servers whose cluster participation status is `online`. This subcommand cannot be executed if the cluster contains any EADS server whose cluster participation status is `standby`. You can determine the cluster participation status with the `eztool status` command.
- This subcommand can be executed when the target EADS servers are in the following status:

- Initialized
- Running
- Closing
- Closed

(3) Format

```
eztool listcache [-v]
                 [--format format-name]
                 [--columns column-name[,column-name]...]
                 [--filter filter-condition]
                 [--match matching-condition]
```

(4) Options and arguments

(a) -v or --verbose

Specify this option if you want to display the details of the command's execution results.

(b) --format format-name

For details about this option, see *14.4.2 How to specify the display format*.

(c) --columns column-name[,column-name]...

For details about this option, see *14.4.3 How to specify column filters*.

(d) --filter filter-condition

For details about this option, see *14.4.4 How to specify row filters*.

(e) --match matching-condition

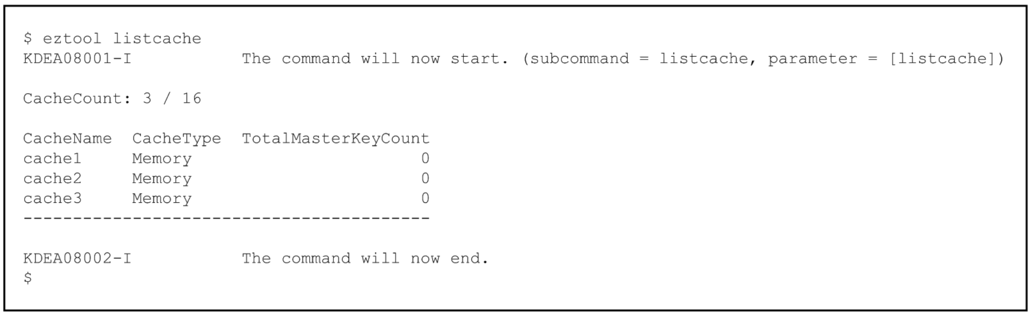
For details about this option, see *14.4.5 How to specify a condition match*.

(5) Output example

The following shows an output example of the `eztool listcache` command's execution results.

For details about the components of the displayed information, see *14.4.1 Components of the displayed information*.

■ If option `-v` or `--verbose` is omitted



```
$ eztool listcache
KDEA08001-I           The command will now start. (subcommand = listcache, parameter = [listcache])

CacheCount: 3 / 16

CacheName  CacheType  TotalMasterKeyCount
cache1     Memory                       0
cache2     Memory                       0
cache3     Memory                       0
-----------------------------------------

KDEA08002-I           The command will now end.
$
```

■ If option `-v` or `--verbose` is specified

```
$ eztool listcache -v
KDEA08001-I           The command will now start. (subcommand = listcache, parameter =
[listcache, -v])

CacheCount: 3 / 16

CacheName  CacheType  TotalMasterKeyCount  Server                 MasterKeyCount
cache1     Memory                       0  XX.XXX.XXX.168:24600                0
                                           XX.XXX.XXX.168:24700                0
                                           XX.XXX.XXX.168:24800                0
                                           XX.XXX.XXX.168:24900                0
                                           XX.XXX.XXX.168:25000                0
cache2     Memory                       0  XX.XXX.XXX.168:24600                0
                                           XX.XXX.XXX.168:24700                0
                                           XX.XXX.XXX.168:24800                0
                                           XX.XXX.XXX.168:24900                0
                                           XX.XXX.XXX.168:25000                0
cache3     Memory                       0  XX.XXX.XXX.168:24600                0
                                           XX.XXX.XXX.168:24700                0
                                           XX.XXX.XXX.168:24800                0
                                           XX.XXX.XXX.168:24900                0
                                           XX.XXX.XXX.168:25000                0
--------------------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

The following tables list and describe the summary and content information that are displayed.

Table 14–13: Summary information displayed by the eztool listcache command

No.	Summary name	Description	Whether displayed	
			-v or --verbose option	
			Omitted	Specified
1	`CacheCount`	Current and maximum numbers of caches	Y	Y

Legend:

Y: Displayed

Table 14–14: Content information displayed by the eztool listcache command

No.	Column name	Description	Whether displayed	
			-v or --verbose option	
			Omitted	Specified
1	`CacheName`	Cache name#	Y	Y
2	`CacheType`	Cache type# One of the following is displayed: • `Memory` Memory cache • `Disk` Disk cache • `2Way` Two-way cache	Y	Y
3	`TotalMasterKeyCount`	Total number of keys in each cache.# This value does not include the number of keys copied for data redundancy purposes.	Y	Y

No.	Column name	Description	Whether displayed	
			-v or --verbose option	
			Omitted	Specified
4	`Server`	EADS server's IP address and port number used to communicate with the EADS clients. This information is displayed in the following format: *IP address*:*port number* Nothing is displayed for an isolated EADS server or if the EADS server has not started.	N	Y
5	`MasterKeyCount`	Number of keys in each cache on each EADS server. This value does not include the number of keys copied for data redundancy purposes. Nothing is displayed for an isolated EADS server or if the EADS server has not started.	N	Y

Legend:

Y: Displayed

N: Not displayed

Notes:

The execution results are displayed according to the following priorities:

1. Displayed in natural order of the `CacheName` values.
2. If the `-v` or `--verbose` option is specified, the rows for the same `CacheName` value are displayed in natural order of the `Server` values.

#

If the execution results contain multiple cells with the same value consecutively, only the first such cell is displayed and the other cells are omitted.

(6) Return code

The following table lists the return codes that this subcommand returns.

Table 14–15: Return codes returned by the eztool listcache command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`110`		Connection establishment failed.
4	`111`		The command failed due to a communication timeout.
5	`120`		The command failed due to a syntax error.

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
6	`130`		The command failed because it could not be executed.
7	`150`		The command failed during execution.
8	`200`		The command failed due to a timeout.

When the `--match` option is specified:

If the condition was satisfied, the subcommand returns `0`; otherwise, the subcommand returns `1`. If the command's execution failed, the return code is the same as when the `--match` option is not specified.

(7) Notes

- If the command times out during output processing, the output results might not be complete. The command might time out even if output processing is complete. If this happens, increase the command's timeout value, and then re-execute the command.
- If this subcommand is executed while scale-out processing is underway, a smaller number of keys might be displayed temporarily.

14.3.7 listesd (displays a list of store data files)

(1) Description

This subcommand displays a list of store data files in the cluster.

(2) Rules

- This subcommand can be executed only when the status of the cluster is one of the following:
  - Cluster available (`AVAILABLE`)
  - Cluster partially available (`PARTIALLY_AVAILABLE`)
- The target of this subcommand is the EADS servers whose cluster participation status is `online`. This subcommand cannot be executed if the cluster contains any EADS server whose cluster participation status is `standby`. You can determine the cluster participation status with the `eztool status` command.
- This subcommand can be executed when the target EADS servers are in the following status:
  - Initialized
  - Running
  - Closing
  - Closed

(3) Format

```
eztool listesd [-v]
               [-d path-name-of-store-data-file-storage-location]
               [--format format-name]
               [--columns column-name[,column-name]...]
               [--filter filter-condition]
               [--match matching-condition]
```

(4) Options and arguments

(a) -v or --verbose

Specify this option if you want to display the details of the command execution results.

(b) -d path-name-of-store-data-file-storage-location or --directory path-name-of-store-data-file-storage-location

This option specifies the path name of the store data file storage location.

Specify this option if you want to display a list of store data files that are located under a specified directory.

The path name cannot be a directory that contains an asterisk (`*`), double quotation mark (`"`), question mark (`?`), vertical bar (`|`), less-than sign (<), or greater-than sign (>).

If a relative path is specified as the path of the store data file storage location, the specified path is treated as being relative to the management directory.

(c) --format format-name

For details about this option, see *14.4.2 How to specify the display format*.

(d) --columns column-name[,column-name]...

For details about this option, see *14.4.3 How to specify column filters*.

(e) --filter filter-condition

For details about this option, see *14.4.4 How to specify row filters*.

(f) --match matching-condition

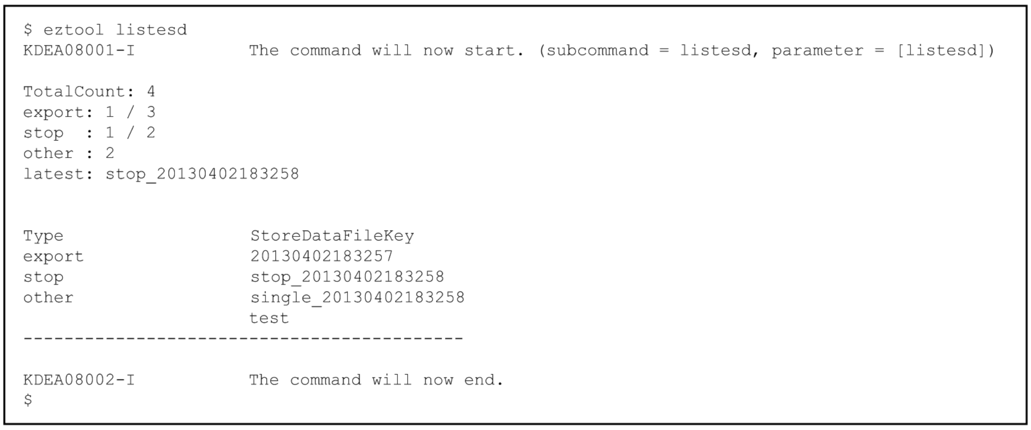
For details about this option, see *14.4.5 How to specify a condition match*.

(5) Output example

The following shows output examples of the `eztool listesd` command's execution results.

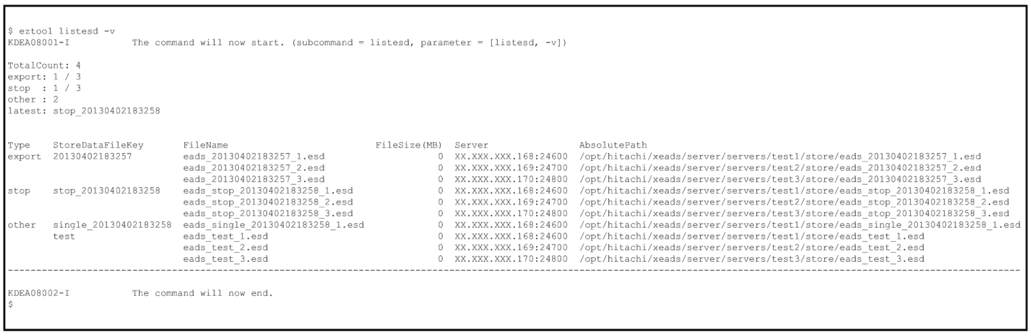
For details about the components of the displayed information, see *14.4.1 Components of the displayed information*.

■ If option `-v` or `--verbose` is omitted



```
$ eztool listesd
KDEA08001-I           The command will now start. (subcommand = listesd, parameter = [listesd])

TotalCount: 4
export: 1 / 3
stop  : 1 / 2
other : 2
latest: stop_20130402183258


Type                  StoreDataFileKey
export                20130402183257
stop                  stop_20130402183258
other                 single_20130402183258
                      test
-------------------------------------------

KDEA08002-I           The command will now end.
$
```

■ If option `-v` or `--verbose` is specified



```
$ eztool listesd -v
KDEA08001-I           The command will now start. (subcommand = listesd, parameter = [listesd, -v])

TotalCount: 4
export: 1 / 3
stop  : 1 / 3
other : 2
latest: stop_20130402183258


Type    StoreDataFileKey       FileName                          FileSize(MB)  Server                AbsolutePath
export  20130402183257         eads_20130402183257_1.esd                    0  XX.XXX.XXX.168:24600  /opt/hitachi/xeads/server/servers/test1/store/eads_20130402183257_1.esd
                               eads_20130402183257_2.esd                    0  XX.XXX.XXX.169:24700  /opt/hitachi/xeads/server/servers/test2/store/eads_20130402183257_2.esd
                               eads_20130402183257_3.esd                    0  XX.XXX.XXX.170:24800  /opt/hitachi/xeads/server/servers/test3/store/eads_20130402183257_3.esd
stop    stop_20130402183258    eads_stop_20130402183258_1.esd               0  XX.XXX.XXX.168:24600  /opt/hitachi/xeads/server/servers/test1/store/eads_stop_20130402183258_1.esd
                               eads_stop_20130402183258_2.esd               0  XX.XXX.XXX.169:24700  /opt/hitachi/xeads/server/servers/test2/store/eads_stop_20130402183258_2.esd
                               eads_stop_20130402183258_3.esd               0  XX.XXX.XXX.170:24800  /opt/hitachi/xeads/server/servers/test3/store/eads_stop_20130402183258_3.esd
other   single_20130402183258  eads_single_20130402183258_1.esd             0  XX.XXX.XXX.168:24600  /opt/hitachi/xeads/server/servers/test1/store/eads_single_20130402183258_1.esd
        test                   eads_test_1.esd                              0  XX.XXX.XXX.168:24600  /opt/hitachi/xeads/server/servers/test1/store/eads_test_1.esd
                               eads_test_2.esd                              0  XX.XXX.XXX.169:24700  /opt/hitachi/xeads/server/servers/test2/store/eads_test_2.esd
                               eads_test_3.esd                              0  XX.XXX.XXX.170:24800  /opt/hitachi/xeads/server/servers/test3/store/eads_test_3.esd
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

The following tables list and describe the summary and content information that are displayed.

Table 14–16: Summary information displayed by the eztool listesd command

No.	Summary name	Description	Whether displayed	
			-v or --verbose option	
			Omitted	Specified
1	`TotalCount`	Total number of store data file keys	Y	Y
2	`export`	Number of store data file keys and the upper limit on the number of generations of the store data files that are output when the `eztool export` command (with the argument omitted) is executed This information is displayed in the following format: *Number of store data file keys / upper limit on the number of generations* This information is not displayed if 0 is specified as the upper limit on the number of generations.	Y	Y
3	`stop`	Number of store data file keys and the upper limit on the number of generations of the store data files that are output when the `eztool stop` command is executed This information is displayed in the following format: *Number of store data file keys / upper limit on the number of generations* This information is not displayed if 0 is specified as the upper limit on the number of generations.	Y	Y

No.	Summary name	Description	Whether displayed	
			-v or --verbose option	
			Omitted	Specified
4	`other`	Number of store data file keys of store data files that are output when the `eztool export` command (with an argument and the `-s` or `--single` option specified) is executed	Y	Y
5	`latest`	Store data file keys of the store data files that are to be imported when the `eztool import` command (with the argument omitted) is executed If there are no applicable store data file keys, `none` is displayed.	Y	Y

Legend:

Y: Displayed

Table 14–17: Content information displayed by the eztool listesd command

No.	Column name	Description	Whether displayed	
			-v or --verbose option	
			Omitted	Specified
1	`Type`	Types of store data files# One of the following is displayed: • `export` Store data files that are output when the `eztool export` command (with the argument omitted) is executed • `stop` Store data files that are output when the `eztool stop` command is executed • `other` Store data files that are output when the `eztool export` command (with an argument and the `-s` or `--single` option specified) is executed	Y	Y
2	`StoreDataFileKey`	Store data file keys# The store data file keys are displayed in ascending order of the ASCII code.	Y	Y
3	`FileName`	Store data file name	N	Y
4	`FileSize(MB)`	Store data file size (megabytes) The digits after the decimal point are truncated.	N	Y
5	`Server`	IP address of the EADS server containing the store data files and the port number of the EADS server that is used to communicate with the EADS client This information is displayed in the following format: *IP address*`:`*port number*	N	Y
6	`AbsolutePath`	Store data file name (absolute path)	N	Y

Legend:

Y: Displayed

N: Not displayed

Notes:

The execution results are displayed according to the following priorities:

1. The value of `Type` is displayed in the order of `export`, `stop`, and `other`.
2. The rows for the same `Type` value are displayed in natural order of the `StoreDataFileKey` values.
3. If the `-v` or `--verbose` option is specified, the rows for the same `Type` and `StoreDataFileKey` values are displayed in natural order of the `Server` values.

#

If the execution results contain multiple cells with the same value consecutively, only the first such cell is displayed and the other cells are omitted.

(6) Return code

The following table lists the return codes that this subcommand returns.

Table 14–18: Return codes returned by the eztool listesd command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`110`		Connection establishment failed.
4	`111`		The command failed due to a communication timeout.
5	`120`		The command failed due to a syntax error.
6	`130`		The command failed because it could not be executed.
7	`150`		The command failed during execution.
8	`200`		The command failed due to a timeout.

When the `--match` option is specified:

If the condition was satisfied, the subcommand returns `0`; otherwise, the subcommand returns `1`. If the command's execution failed, the return code is the same as when the `--match` option is not specified.

(7) Notes

If the command times out during output processing, the output results might not be complete. The command might time out even if output processing is complete. If this happens, increase the command's timeout value, and then re-execute the command.

14.3.8 listgroup (displays a list of group names)

(1) Description

This subcommand displays a list of the group hierarchy names of the groups in the highest hierarchy that are stored in a cache.

(2) Rules

- You can execute this subcommand when the EADS server is running.
- This subcommand executes in descending order of the EADS server positions (`Position`).
- This subcommand can be executed if the number of groups at the highest hierarchy in the target range does not exceed 1,000, based on a check performed prior to execution of the subcommand.

(3) Format

```
eztool listgroup [-s] [-v] [-f maximum-number-of-groups-during-forced-
execution]
                 cache-name
                 [--format format-name]
                 [--columns column-name[,column-name]...]
                 [--filter filter-condition]
                 [--match matching-condition]
```

(4) Options and arguments

(a) -s or --single

Specify this option to display the group hierarchy names at the highest hierarchy for only the EADS server that stores the specified groups and on which this subcommand is executed.

(b) -v or --verbose

Specify this option if you want to display the details of the command's execution results.

(c) -f maximum-number-of-groups-during-forced-execution or --force maximum-number-of-groups-during-forced-execution

Specify this option when the number of groups in the target range is greater than 1,000 and you want to increase the maximum number of groups that can be processed by the subcommand, and then forcibly execute the subcommand.

Note that when this subcommand is executed with the maximum value increased, its processing might not be completed successfully and a large amount of EADS server resources might be used.

You can specify an integer in the range from `1001` to `10000` for the maximum number of groups.

Important note

Consider specifying this option when the number of groups in the target range is not too much greater than 1,000.

If the total number of groups in the highest hierarchy stored in any one of the EADS servers does not exceed 1,000, we recommend that instead of specifying this option, you execute the `eztool listgroup` command sequentially on each EADS server with the `-s` or `--single` option specified.

(d) cache-name

Specify the name of the cache for which group names are to be displayed.

The following characters are permitted for a cache name:

- If cache property files were not used to create caches
  A maximum of 32 single-byte characters in ASCII codes `0x20` to `0x7E`
- If cache property files were used to create caches
  A maximum of 32 single-byte alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`)

(e) --format format-name

For details about this option, see *14.4.2 How to specify the display format*.

(f) --columns column-name[,column-name]...

For details about this option, see *14.4.3 How to specify column filters*.

(g) --filter filter-condition

For details about this option, see *14.4.4 How to specify row filters*.

(h) --match matching-condition

For details about this option, see *14.4.5 How to specify a condition match*.

(5) Output example

The following shows output examples of the `eztool listgroup` command's execution results.

For details about the components of the displayed information, see *14.4.1 Components of the displayed information*.

■ If option `-v` or `--verbose` is omitted

```
$ eztool listgroup cache1
KDEA08001-I           The command will now start. (subcommand = listgroup, parameter =
[listgroup, cache1])

GroupCount: 20

Server                Position     ID  GroupName
XX.XXX.XXX.168:24600   1288490189   1  group02
                                       group04
XX.XXX.XXX.168:24700    429496730   2  group01
                                       group09
                                       group11
                                       group17
                                       group18
                                       group19
XX.XXX.XXX.168:24800   -429496729   3  group05
                                       group10
                                       group12
                                       group13
                                       group15
XX.XXX.XXX.168:24900  -1288490188   4  group06
                                       group07
                                       group08
                                       group14
XX.XXX.XXX.168:25000  -2147483648   5  group03
                                       group16
                                       group20
------------------------------------------------

KDEA08002-I           The command will now end.
$
```

■ If option `-v` or `--verbose` is specified

```
$ eztool listgroup -v cache1
KDEA08001-I           The command will now start. (subcommand = listgroup, parameter =
[listgroup, -v, cache1])

GroupCount: 20

Server                Position     ID  GroupName  HashValue    KeyCount
XX.XXX.XXX.168:24600   1288490189   1  group02     2037266030       100
                                       group04     1388111144       100
XX.XXX.XXX.168:24700    429496730   2  group09     1269679203       100
                                       group11     1229725181       100
                                       group17     1172163628       100
                                       group19     1006380133       100
                                       group18      703078836       100
                                       group01      648506385       100
XX.XXX.XXX.168:24800   -429496729   3  group10      260938713       100
                                       group15      232103928       100
                                       group12      -56466136       100
                                       group05      -73411842       100
                                       group13     -163284496       100
XX.XXX.XXX.168:24900  -1288490188   4  group14     -503630070       100
                                       group07     -699511542       100
                                       group08     -992621419       100
                                       group06    -1203276328       100
XX.XXX.XXX.168:25000  -2147483648   5  group16    -1445454454       100
                                       group20    -1917239959       100
                                       group03    -1965919439       100
-----------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

The following tables list and describe the summary and content information that are displayed.

Table 14–19: Summary information displayed by the eztool listgroup command

No.	Summary name	Description	Whether displayed	
			-v or --verbose option	
			Omitted	Specified
1	`GroupCount`	Number of groups in the highest hierarchy that are stored on EADS servers	Y	Y

Legend:

Y: Displayed

Table 14–20: Content information displayed by the eztool listgroup command

No.	Column name	Description	Whether displayed	
			-v or --verbose option	
			Omitted	Specified
1	`Server`	EADS server's IP address and port number used to communicate with the EADS clients.# This information is displayed in the following format: *IP address*:*port number*	Y	Y
2	`Position`	Location (hash value) of an EADS server#	Y	Y
3	`ID`	EADS server ID#	Y	Y
4	`GroupName`	Group hierarchy name in the highest hierarchy	Y	Y
5	`HashValue`	Location (hash value) of a group in the highest hierarchy	N	Y
6	`KeyCount`	Number of keys belonging to the groups in the highest hierarchy	N	Y

Legend:

Y: Displayed

N: Not displayed

Notes:

The execution results are displayed according to the following priorities:

1. The results are displayed in descending order of the `Position` values.
2. The rows with the same `Position` value are displayed in natural order of the `GroupName` values. If the `-v` or `--verbose` option is specified, the rows with the same `Position` value are displayed in descending order of the `HashValue` values.

#

If the execution results contain multiple cells with the same value consecutively, only the first such cell is displayed and the other cells are omitted.

(6) Return code

The following table lists the return codes that this subcommand returns.

Table 14–21: Return codes returned by the eztool listgroup command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`120`		The command failed due to a syntax error.

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
4	`150`		The command failed during execution.
5	`200`		The command failed due to a timeout.

When the `--match` option is specified:

If the condition was satisfied, the subcommand returns `0`; otherwise, the subcommand returns `1`. If the command's execution failed, the return code is the same as when the `--match` option is not specified.

(7) Notes

- This subcommand enables you to check a list of group hierarchy names in the highest hierarchy without having to create an application program. However, if the number of groups in the highest hierarchy in the target range exceeds 1,000, you need to create an application program.
- If group hierarchy names in the highest hierarchy are added while this subcommand is executing, the following occurs, depending on the area where the group hierarchy names are stored:
    - If the group hierarchy names are stored in an area for which group names have already been acquired
      The added group hierarchy names are not displayed in the list.
    - If the group hierarchy names are stored in an area for which group names have not been acquired
      The added group hierarchy names are displayed in the list.

  Note that the number of group hierarchy names in the highest hierarchy is checked before the list of group names is actually acquired. If a large number of keys that include new group names are added after that check, the group names that exceed the maximum value are included in the display.
- If the command times out during output processing, the output results might not be complete. The command might time out even if output processing is complete. If this happens, increase the command's timeout value, and then re-execute the command.
- If the target range contains a large number of group names, the amount of resources required for processing increases.
- The number of keys that do not belong to a group cannot be checked.
- If EADS servers are added to or restored in the cluster while this subcommand is executing, the added or restored EADS servers are not included as processing targets. As a result, some group names might not be displayed. If this occurs, re-execute the subcommand.

14.3.9 listkey (displays a list of keys)

(1) Description

This subcommand displays a list of keys that are stored in a cache.

(2) Rules

- You can execute this subcommand when the EADS server is running.
- This subcommand executes in descending order of the EADS server positions (`Position`).

- This subcommand can be executed if the number of keys in the target range does not exceed 1,000, based on a check performed prior to execution of the subcommand.

(3) Format

```
eztool listkey [-g group-name|-s] [-f maximum-number-of-keys-during-forced-
execution]
               cache-name
               [--format format-name]
               [--columns column-name[,column-name]...]
               [--filter filter-condition]
               [--match matching-condition]
```

(4) Options and arguments

(a) -g group-name or --group group-name

Specify this option to display only those keys that belong to the specified group.

For details about the data that can be specified as group names, see *15.2.2(2) Data that can be specified as group names*.

(b) -s or --single

Specify this option if you want to display only those keys stored on the EADS server on which this subcommand is executed.

(c) -f maximum-number-of-keys-during-forced-execution or --force maximum-number-of-keys-during-forced-execution

Specify this option when the number of keys in the target range is greater than 1,000 and you want to increase the maximum number of keys that can be processed by the subcommand, and then forcibly execute the subcommand.

Note that when this subcommand is executed with the maximum value increased, its processing might not be completed successfully and a large amount of EADS server resources might be used.

You can specify an integer in the range from `1001` to `10000` for the maximum number of keys.

> **Important note**
>
> Consider specifying this option when the number of keys in the target range is not too much greater than 1,000.
>
> - If the total number of keys stored in any one of the EADS servers does not exceed 1,000, we recommend that instead of executing this operation, you execute the `eztool listkey` command sequentially on each EADS server with the `-s` or `--single` option specified.
> - If the total number of keys that belong to any one of the groups in the highest hierarchy does not exceed 1,000, execute the `eztool listkey` command with the `-g` or `--group` option specified sequentially for each group with the group hierarchy name in the highest hierarchy obtained by using the `eztool listgroup` command (note that keys that do not belong to groups cannot be checked by this method).

(d) cache-name

Specify the name of the cache for which keys are to be displayed.

The following characters are permitted for a cache name:

- If cache property files were not used to create caches
  A maximum of 32 single-byte characters in ASCII codes 0x20 to 0x7E
- If cache property files were used to create caches
  A maximum of 32 single-byte alphanumeric characters (0 to 9, A to Z, and a to z)

(e) --format format-name

For details about this option, see *14.4.2 How to specify the display format*.

(f) --columns column-name[,column-name]...

For details about this option, see *14.4.3 How to specify column filters*.

(g) --filter filter-condition

For details about this option, see *14.4.4 How to specify row filters*.

(h) --match matching-condition

For details about this option, see *14.4.5 How to specify a condition match*.

(5) Output example

The following shows output examples of the eztool listkey command's execution results.

For details about the components of the displayed information, see *14.4.1 Components of the displayed information*.

```
$ eztool listkey cache1
KDEA08001-I           The command will now start. (subcommand = listkey, parameter = [listkey,
cache1])

KeyCount: 25

Server                Position     ID  Key
XX.XXX.XXX.168:24600   1288490189   1  [1]groupA:element01
                                       [1]groupA:element02
                                       [1]groupA:element03
                                       [1]groupA:element04
                                       [1]groupA:element05
XX.XXX.XXX.168:24700    429496730   2  [2]groupB:element01
                                       [2]groupB:element02
                                       [2]groupB:element03
                                       [2]groupB:element04
                                       [2]groupB:element05
XX.XXX.XXX.168:24800   -429496729   3  [3]groupC:element01
                                       [3]groupC:element02
                                       [3]groupC:element03
                                       [3]groupC:element04
                                       [3]groupC:element05
XX.XXX.XXX.168:24900  -1288490188   4  [4]groupD:element01
                                       [4]groupD:element02
                                       [4]groupD:element03
                                       [4]groupD:element04
                                       [4]groupD:element05
XX.XXX.XXX.168:25000  -2147483648   5  [5]groupE:element01
                                       [5]groupE:element02
                                       [5]groupE:element03
                                       [5]groupE:element04
                                       [5]groupE:element05
----------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

The following tables list and describe the summary and content information that are displayed.

Table 14–22: Summary information displayed by the eztool listkey command

No.	Summary name	Description
1	`KeyCount`	Number of keys in the specified range

Table 14–23: Content information displayed by the eztool listkey command

No.	Column name	Description
1	`Server`	EADS server's IP address and port number used to communicate with the EADS clients.# This information is displayed in the following format: *IP address* **:** *port number*
2	`Position`	Location (hash value) of an EADS server#
3	`ID`	EADS server ID#
4	`Key`	Keys in the specified range

Notes:

The execution results are displayed according to the following priorities:

1. The results are displayed in descending order of the `Position` values.
2. The rows with the same `Position` value are displayed in natural order of the `Key` values.

#

If the execution results contain multiple cells with the same value consecutively, only the first such cell is displayed and the other cells are omitted.

(6) Return code

The following table lists the return codes that this subcommand returns.

Table 14–24: Return codes returned by the eztool listkey command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`120`		The command failed due to a syntax error.
4	`150`		The command failed during execution.
5	`200`		The command failed due to a timeout.

When the `--match` option is specified:

If the condition was satisfied, the subcommand returns `0`; otherwise, the subcommand returns `1`. If the command's execution failed, the return code is the same as when the `--match` option is not specified.

(7) Notes

- This subcommand enables you to check a list of keys without having to create an application program. However, if the number of keys in the target range exceeds 1,000, you need to create an application program.
- If keys are added while this subcommand is executing, the following occurs, depending on the area where the added keys are stored:
    - If the added keys are stored in an area for which a list has already been acquired
      The added keys are not displayed in the list.
    - If the added keys are stored in an area for which a list of keys has not been acquired
      The added keys are displayed in the list.

  Note that the number of keys is checked before the list of keys is actually acquired. If a large number of keys are added after that check, the keys that exceed the maximum value are included in the display.
- If the command times out during output processing, the output results might not be complete. The command might time out even if output processing is complete. If this happens, increase the command's timeout value, and then re-execute the command.
- If a large amount of data is stored, the amount of resources required for processing increases.
- If EADS servers are added to or restored in the cluster while this subcommand is executing, the added or restored EADS servers are not included as processing targets. As a result, some keys might not be displayed. If this occurs, re-execute the subcommand.

14.3.10 getposition (displays data storage locations)

(1) Description

This subcommand displays the EADS servers that store a specified key or group.

(2) Rules

- This subcommand can be executed only when the status of the cluster is one of the following:
    - Cluster available (`AVAILABLE`)
    - Cluster partially available (`PARTIALLY_AVAILABLE`)
- This subcommand can be executed when the EADS servers are in the following status:
    - Initializing
    - Initialized
    - Running
    - Closing
    - Closed
    - Isolated
    - Stopping
- If the `-l` or `--local` option is specified, this subcommand can be executed regardless of the cluster or EADS server status.

- If an EADS server to be added to the cluster has not yet participated in the cluster, this subcommand cannot be executed on that EADS server.

(3) Format

```
eztool getposition key-or-group-name [-l]
                   [--format format-name]
                   [--columns column-name[,column-name]...]
                   [--filter filter-condition]
                   [--match matching-condition]
```

(4) Options and arguments

(a) key-or-group-name

Specify the key name or group name associated with the data whose storage EADS servers are to be displayed.

For details about the data that can be specified, see *15.2.2(1) Data types that can be specified as keys* or *15.2.2(2) Data that can be specified as group names*.

(b) -l or --local

Specify this option if you want to know whether the EADS server stores the specified key or group in the cluster configuration set up in the cluster properties.

When this option is specified, the command imports the contents of the cluster property file for the EADS server on which the command was executed.

An error occurs in the following cases:

- There is no cluster property file on the EADS server on which the command is executed.
- The `eads.node.`*EADS-server-ID*`.address` and `eads.node.`*EADS-server-ID*`.port` parameters are not defined in the cluster properties.

Important note

If you want to know from the cluster information currently in use which EADS servers store the specified key or group, execute the subcommand without specifying this option.

If this option is omitted, the subcommand does not display information about the EADS servers that are shut down. Check the data copy status by comparing the number of displayed EADS servers with the value of `ReplicationCount` (data multiplicity).

(c) --format format-name

For details about this option, see *14.4.2 How to specify the display format*.

(d) --columns column-name[,column-name]...

For details about this option, see *14.4.3 How to specify column filters*.

(e) --filter filter-condition

For details about this option, see *14.4.4 How to specify row filters*.

(f) --match matching-condition

For details about this option, see *14.4.5 How to specify a condition match*.

(5) Output example

The following shows an output example of the `eztool getposition` command's execution results.

For details about the components of the displayed information, see *14.4.1 Components of the displayed information*.

```
$ eztool getposition group:element
KDEA08001-I          The command will now start. (subcommand = getposition, parameter = [getposition,
group:element])

ReplicationCount: 3 / 3
HashValue: -1039521297

No.  IP_Address     ClientPort  Position
  1  XX.XXX.XXX.171      24900  -1288490188
  2  XX.XXX.XXX.172      25000  -2147483648
  3  XX.XXX.XXX.168      24600   1288490189
-------------------------------------------

KDEA08002-I          The command will now end.
$
```

The following tables list and describe the summary and content information that are displayed.

Table 14–25: Summary information displayed by the eztool getposition command

No.	Summary name	Description
1	`ReplicationCount`	Data multiplicity
2	`HashValue`	Hash value of the specified key or group name

Table 14–26: Content information displayed by the eztool getposition command

No.	Column name	Description
1	`No.`	Data priority# Displayed in ascending order.
2	`IP_Address`	IP address of EADS server
3	`ClientPort`	EADS server's port number used for communication with the EADS clients
4	`Position`	Location of EADS server (hash value)

#

If redundant copies of data have been created, this information is displayed in order, beginning with the EADS server on which the original data is stored, followed by the EADS servers to which the data has been copied.

In this example, `No.1` indicates the EADS server on which the original data is stored and the subsequent numbers starting with `No.2` indicate the EADS servers to which data has been copied.

(6) Return code

The following table lists the return codes that this subcommand returns.

Table 14–27: Return codes returned by the eztool getposition command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`110`		Connection establishment failed.
4	`111`		The command failed due to a communication timeout.
5	`120`		The command failed due to a syntax error.
6	`130`		The command failed because it could not be executed.
7	`150`		The command failed during execution.
8	`200`		The command failed due to a timeout.

When the `--match` option is specified:

If the condition was satisfied, the subcommand returns `0`; otherwise, the subcommand returns `1`. If the command's execution failed, the return code is the same as when the `--match` option is not specified.

(7) Notes

- If the command times out during output processing, the output results might not be complete. The command might time out even if output processing is complete. If this happens, increase the command's timeout value, and then re-execute the command.
- This subcommand results in an error if all EADS servers are isolated.

14.3.11 storeusage (checks the usage status of ranges and caches)

(1) Description

This subcommand displays the number of keys stored in and the amount of memory usage for each range. The subcommand also displays the upper limits of the total data restrictions that have been set for the ranges and caches.

(2) Rules

- This subcommand can be executed only when the status of the cluster is one of the following:
  - Cluster available (`AVAILABLE`)
  - Cluster partially available (`PARTIALLY_AVAILABLE`)
- The target of this subcommand is the EADS servers whose cluster participation status is `online`. This subcommand cannot be executed if the cluster contains any EADS server whose cluster participation status is `standby`. You can determine the cluster participation status with the `eztool status` command.
- This subcommand can be executed when the target EADS servers are in the following status:

- Initializing
- Initialized
- Running
- Closing
- Closed

(3) Format

```
eztool storeusage [--replica|--cache cache-name]
                  [--format format-name]
                  [--columns column-name[,column-name]...]
                  [--filter filter-condition]
                  [--match matching-condition]
```

(4) Options and arguments

(a) --replica

Specify this option to also include the number of keys copied for data redundancy purposes.

If this option is omitted, the number of keys copied for data redundancy purposes is not included.

(b) --cache cache-name

Specify this option if you want to display the number of keys and memory usage amount for each range in a specific cache. For a disk cache or two-way cache, the disk usage amount and its upper limit value are displayed.

The following characters are permitted for a cache name:

- If cache property files were not used to create caches
  A maximum of 32 single-byte characters in ASCII codes `0x20` to `0x7E`
- If cache property files were used to create caches
  A maximum of 32 single-byte alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`)

If this option is omitted, the sum of the numbers of keys in all caches that are obtained for each range is displayed as the total number of keys.

(c) --format format-name

For details about this option, see *14.4.2 How to specify the display format*.

(d) --columns column-name[,column-name]...

For details about this option, see *14.4.3 How to specify column filters*.

(e) --filter filter-condition

For details about this option, see *14.4.4 How to specify row filters*.

(f) --match matching-condition

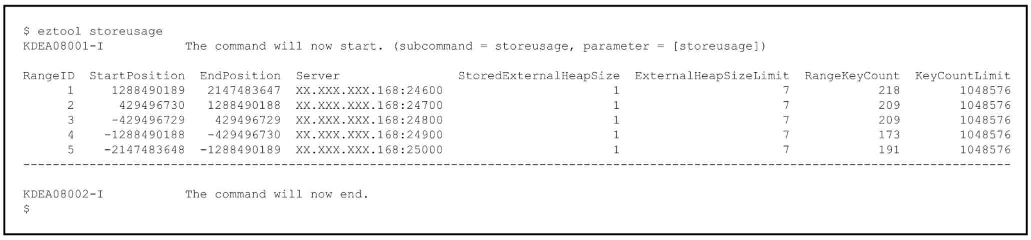
For details about this option, see *14.4.5 How to specify a condition match*.

(5) Output example

The following shows output examples of the `eztool storeusage` command's execution results.

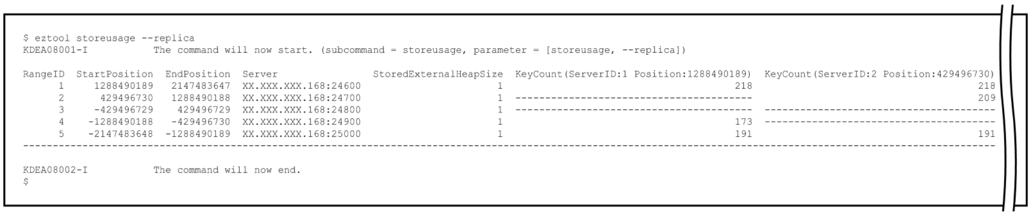
For details about the components of the displayed information, see *14.4.1 Components of the displayed information*.

■ When the `--replica` option and the `--cache` option are both omitted



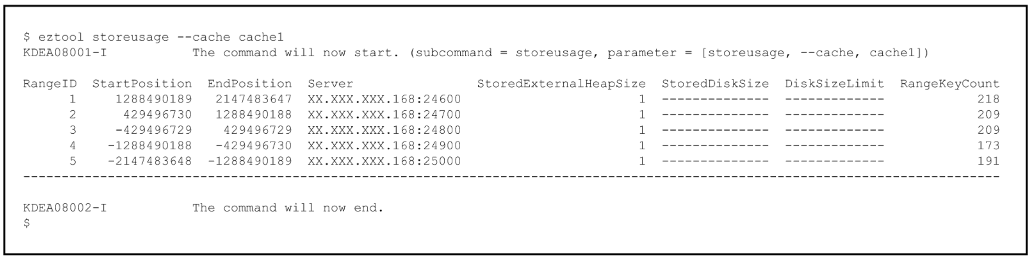
```
$ eztool storeusage
KDEA08001-I          The command will now start. (subcommand = storeusage, parameter = [storeusage])

RangeID  StartPosition  EndPosition  Server                StoredExternalHeapSize  ExternalHeapSizeLimit  RangeKeyCount  KeyCountLimit
      1     1288490189   2147483647  XX.XXX.XXX.168:24600                       1                      7            218        1048576
      2      429496730   1288490188  XX.XXX.XXX.168:24700                       1                      7            209        1048576
      3     -429496729    429496729  XX.XXX.XXX.168:24800                       1                      7            209        1048576
      4    -1288490188   -429496730  XX.XXX.XXX.168:24900                       1                      7            173        1048576
      5    -2147483648  -1288490189  XX.XXX.XXX.168:25000                       1                      7            191        1048576
----------------------------------------------------------------------------------------------------------------------------------

KDEA08002-I          The command will now end.
$
```

■ When the `--replica` option is specified



```
$ eztool storeusage --replica
KDEA08001-I          The command will now start. (subcommand = storeusage, parameter = [storeusage, --replica])

RangeID  StartPosition  EndPosition  Server                StoredExternalHeapSize  KeyCount(ServerID:1 Position:1288490189)  KeyCount(ServerID:2 Position:429496730)
      1     1288490189   2147483647  XX.XXX.XXX.168:24600                       1                                       218                                      218
      2      429496730   1288490188  XX.XXX.XXX.168:24700                       1  ----------------------------------------                                      209
      3     -429496729    429496729  XX.XXX.XXX.168:24800                       1  ----------------------------------------  ---------------------------------------
      4    -1288490188   -429496730  XX.XXX.XXX.168:24900                       1                                       173  ---------------------------------------
      5    -2147483648  -1288490189  XX.XXX.XXX.168:25000                       1                                       191                                      191
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------

KDEA08002-I          The command will now end.
$
```

■ When the `--cache` option is specified



```
$ eztool storeusage --cache cache1
KDEA08001-I          The command will now start. (subcommand = storeusage, parameter = [storeusage, --cache, cache1])

RangeID  StartPosition  EndPosition  Server                StoredExternalHeapSize  StoredDiskSize  DiskSizeLimit  RangeKeyCount
      1     1288490189   2147483647  XX.XXX.XXX.168:24600                       1  --------------  -------------            218
      2      429496730   1288490188  XX.XXX.XXX.168:24700                       1  --------------  -------------            209
      3     -429496729    429496729  XX.XXX.XXX.168:24800                       1  --------------  -------------            209
      4    -1288490188   -429496730  XX.XXX.XXX.168:24900                       1  --------------  -------------            173
      5    -2147483648  -1288490189  XX.XXX.XXX.168:25000                       1  --------------  -------------            191
---------------------------------------------------------------------------------------------------------------------------------

KDEA08002-I          The command will now end.
$
```

The following table lists and describes the content information that is displayed.

Table 14–28: Content information displayed by the eztool storeusage command

No.	Column name	Description	Whether displayed		
			Specification of --replica and --cache options		
			Both omitted	--replica option specified	---cache option specified
1	`RangeID`	Range ID	Y	Y	Y
2	`StartPosition`	Start position (hash value) of each range#1 The positions (hash values) are displayed in descending order.	Y	Y	Y
3	`EndPosition`	End position (hash value) of each range#1	Y	Y	Y

No.	Column name	Description	Whether displayed		
			Specification of --replica and --cache options		
			Both omitted	--replica option specified	---cache option specified
4	`Server`	IP address of the data storage EADS server in each range and the EADS server's port number used to communicate with the EADS clients.[#2] This information is displayed in the following format: *IP address*:*port number*	Y	Y	Y
5	`StoredExternalHeapSize`	Amount of total memory usage in memory caches and two-way caches[#3] (megabytes) • The digits after the decimal point are truncated. • The memory usage amount for a disk cache is treated as 0.	Y	Y	Y
6	`ExternalHeapSizeLimit`	Maximum memory usage amount[#3,#4] (megabytes) For a disk cache, `0` is displayed.	Y	N	N
7	`StoredDiskSize`	Disk usage amount of disk caches or two-way caches[#3,#5] (megabytes) The digits after the decimal point are truncated.	N	N	Y
8	`DiskSizeLimit`	Maximum disk usage amount of disk caches or two-way caches[#3,#4,#5] (megabytes)	N	N	Y
9	`RangeKeyCount`	Total number of keys in each range[#3] This value does not include the number of keys copied for data redundancy purposes.	Y	N	Y
10	`KeyCountLimit`	Upper limit of the total number of keys in each range[#3,#4] This does not include the number of keys copied for data redundancy purposes.	Y	N	N
11	`KeyCount(ServerID:x Position:y)`[#6]	Total number of keys in each range for each EADS server[#3] • This includes the number of keys copied for data redundancy purposes. • This does not include the number of keys on isolated or inactive EADS servers. • If redundant copies of data are created, this information is displayed in descending order of the locations (hash values) of the EADS servers. • Because this value is acquired when the command is executed, the value might not match between the EADS server on which the original data is stored and the EADS servers to which the data has been copied. • Nothing is displayed for an EADS server that is neither the EADS server on which the original data is stored nor an EADS server to which the data has been copied.	N	Y	N

Legend:

Y: Displayed

N: Not displayed

#1

Each range is as follows:

- If the StartPosition value is less than the EndPosition value
  Range from the StartPosition value to the EndPosition value
- If the StartPosition value is greater than the EndPosition value
  Range that combines the following ranges:
  - From the StartPosition value to 2,147,483,647
  - from -2,147,483,648 to the EndPosition value

#2

A hyphen (-) is displayed if the EADS server on which the original data in the corresponding range is stored and the EADS servers to which the data in the corresponding range is to be copied are all isolated or in process down status.

#3

A hyphen (-) is displayed if information for the corresponding range cannot be acquired.

#4

A hyphen (-) is displayed if the total data restriction function is disabled.

#5

A hyphen (-) is displayed for a memory cache.

#6

For *x*, the EADS server's server ID is displayed. For *y*, the EADS server's location (hash value) is displayed.

(6) Return code

The following table lists the return codes that this subcommand returns.

Table 14–29: Return codes returned by the eztool storeusage command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	0	0	Command execution was successful.
2	101	101	Initialization of the command failed.
3	110		Connection establishment failed.
4	111		The command failed due to a communication timeout.
5	120		The command failed due to a syntax error.
6	130		The command failed because it could not be executed.
7	150		The command failed during execution.
8	200		The command failed due to a timeout.

When the --match option is specified:

If the condition was satisfied, the subcommand returns 0; otherwise, the subcommand returns 1. If the command's execution failed, the return code is the same as when the --match option is not specified.

(7) Notes

- If the command times out during output processing, the output results might not be complete. The command might time out even if output processing is complete. If this happens, increase the command's timeout value, and then re-execute the command.
- This subcommand might fail during scale-out processing because the cluster configuration is changed.

14.3.12 unlock (unlock)

(1) Description

This subcommand unlocks a command.

For details, see *14.3.1 Locking between commands*.

(2) Rules

- This subcommand can be executed only when the status of the cluster is one of the following:
  - Cluster available (AVAILABLE)
  - Cluster partially available (PARTIALLY_AVAILABLE)
- The target of this subcommand is the EADS servers whose cluster participation status is online. However, the subcommand cannot be executed in the following cases:
  - An operation is underway.
  - The cluster contains any EADS server whose cluster participation status is standby.

  You can determine the cluster participation status with the eztool status command.
- This subcommand can be executed when the target EADS servers are in the following status:
  - Initialized
  - Running
  - Closed

(3) Format

```
eztool unlock
```

(4) Return code

The following table lists the return codes that this subcommand returns.

Table 14–30: Return codes returned by the eztool unlock command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`110`		Connection establishment failed.
4	`111`		The command failed due to a communication timeout.
5	`120`		The command failed due to a syntax error.
6	`130`		The command failed because it could not be executed.
7	`131`		The command failed because another command was executing.
8	`150`		The command failed during execution.
9	`200`		The command failed due to a timeout.

(5) Notes

This subcommand results in an error if all EADS servers are isolated.

14.3.13 createcache (creates a cache)

(1) Description

This subcommand creates a cache.

You can create a maximum of 16 caches, including memory caches, disk caches, and two-way caches.

(2) Rules

- You can execute this subcommand when the cluster's status is `AVAILABLE`.
- The target of this subcommand is the EADS servers whose cluster participation status is `online`. This subcommand cannot be executed if the cluster contains any EADS server whose cluster participation status is `standby`. You can determine the cluster participation status with the `eztool status` command.
- This subcommand can be executed when the target EADS servers are in the following status:
  - Initialized
  - Closed
- To prevent a full garbage collection (`FullGC`) from occurring while operations are underway, each EADS server performs `FullGC` when this subcommand terminates.

- While this subcommand has obtained a lock from an EADS server, the EADS server is not isolated. However, if processing is shut down or the EADS server terminates while the subcommand has the lock, the lock is released, and then the EADS server is isolated.

(3) Format

```
eztool createcache cache-name
```

(4) Options and arguments

(a) cache-name

Specify a name for the cache you are creating.

The following characters are permitted for a cache name:

- If cache property files are not used
  A maximum of 32 single-byte characters in ASCII codes `0x20` to `0x7E`
- If cache property files are used
  A maximum of 32 single-byte alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`)

(5) Return code

The following table lists the return codes that this subcommand returns.

Table 14–31: Return codes returned by the eztool createcache command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	0	0	Command execution was successful.
2	101	101	Initialization of the command failed.
3	110		Connection establishment failed.
4	111		The command failed due to a communication timeout.
5	120		The command failed due to a syntax error.
6	130		The command failed because it could not be executed.
7	131		The command failed because another command was executing.
8	150		The command failed during execution.
9	200		The command failed due to a timeout.

(6) Notes

- If no cache property file has been created, a memory cache is created.

- If the cache type is memory cache or two-way cache, an error results if either of the following conditions, as applicable, is satisfied:
  - Condition when the total data restriction function is enabled

    The value obtained by dividing the size of the area that stores the value part of the `eads.java.external.heapsize` parameter in the shared properties by the number of redundant copies of data plus the original is less than 1 megabyte.
  - Condition when the total data restriction function is disabled

    The size of the area that stores the value part of the `eads.java.external.heapsize` parameter in the shared properties is zero.
- When a cache is created, the number of threads created for the cache is 2 × (data multiplicity × 2 - 1). Note that, as the number of threads increases, the amount of memory used also increases.
- If cache creation processing fails, use one of the following methods to create the cache again:
  - Execute the `eztool listcache` command to determine if the cache exists. If it does exist, use the `eztool deletecache` command to delete the cache, and then re-create the cache.
  - If the cache type is disk cache or two-way cache, execute the `eztool listecf` command to determine if the cache files exist. If they do exist, use the `eztool deleteecf` command to delete the cache files, and then re-create the cache.
  - Check the parameter values specified in the cache property file. If the parameter values are invalid, correct them, and then re-create the cache.
  - Check the `eads.java.external.heapsize` parameter value in the shared properties. If the parameter value is invalid, correct it, and then re-create the cache.

14.3.14 deletecache (deletes a cache)

(1) Description

This subcommand deletes a cache.

When executed, this subcommand deletes a specified cache and the data contained in it.

(2) Rules

- You can execute this subcommand when the cluster's status is `AVAILABLE`.
- The target of this subcommand is the EADS servers whose cluster participation status is `online`. This subcommand cannot be executed if the cluster contains any EADS server whose cluster participation status is `standby`. You can determine the cluster participation status with the `eztool status` command.
- This subcommand can be executed when the target EADS servers are in the following status:
  - Initialized
  - Closed
- To prevent a full garbage collection (`FullGC`) from occurring while operations are underway, each EADS server performs `FullGC` when this subcommand terminates.
- While this subcommand has obtained a lock from the EADS server, the EADS server is not isolated. However, if a process is shut down or the EADS server terminates while the subcommand has the lock, the lock is released, and then the EADS server is isolated.

(3) Format

```
eztool deletecache cache-name
                   [--with_deleteecf]
```

(4) Options and arguments

(a) cache-name

Specify the name of cache you want to delete.

The following characters are permitted for a cache name:

- If cache property files were not used to create caches
  A maximum of 32 single-byte characters in ASCII codes `0x20` to `0x7E`
- If cache property files were used to create caches
  A maximum of 32 single-byte alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`)

(b) --with_deleteecf

If you are deleting a disk cache or two-way cache, specify this option to also delete the cache files.

(5) Return code

The following table lists the return codes that this subcommand returns.

Table 14–32: Return codes returned by the eztool deletecache command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	0	0	Command execution was successful.
2	101	101	Initialization of the command failed.
3	110		Connection establishment failed.
4	111		The command failed due to a communication timeout.
5	120		The command failed due to a syntax error.
6	130		The command failed because it could not be executed.
7	131		The command failed because another command was executing.
8	150		The command failed during execution.
9	200		The command failed due to a timeout.

14.3.15 export (exports data)

(1) Description

This subcommand exports data from memory caches to store data files.

If there is no memory cache, this subcommand does not execute.

(2) Rules

- You can execute this subcommand when the cluster's status is `AVAILABLE`.
- The target of this subcommand is the EADS servers whose cluster participation status is `online`. This subcommand cannot be executed if the cluster contains any EADS server whose cluster participation status is `standby`. You can determine the cluster participation status with the `eztool status` command.
- You can execute this subcommand when the target EADS servers are closed.
- When the `-s` or `--single` option is specified, the subcommand can be executed on an EADS server that is in any of the following statuses, regardless of the cluster's status:
    - Initialized
    - Running
    - Closing
    - Closed
    - Isolated
- To prevent a full garbage collection (`FullGC`) from occurring while operations are underway, each EADS server performs `FullGC` when this subcommand terminates. If the `-s` or `--single` option is specified, `FullGC` is not performed.
- While this subcommand has obtained a lock from the EADS server, the EADS server is not isolated. However, if a process is shut down or the EADS server terminates while the subcommand has the lock, the lock is released, and then the EADS server is isolated.

(3) Format

```
eztool export [-s]
              [-d path-name-of-store-data-file-output-destination]
              [store-data-file-key]
```

(4) Options and arguments

(a) -s or --single

Specify this option to export only the data retained by the EADS server that executes the command, not the data that is retained in the entire cluster.

For example, specify this option to back up data from an isolated EADS server.

A lock is not obtained from the EADS server when this option is specified. Operation is not guaranteed if caches are deleted, if data is deleted from caches, or if store data files are deleted during data export processing. Note also that the operation is not displayed by the `eztool status` command as being underway.

(b) -d path-name-of-store-data-file-output-destination or --directory path-name-of-store-data-file-output-destination

This option specifies the path name of the store data file output destination.

Specify this option if you want to export a store data file to a specified directory.

The path name cannot be a directory that contains an asterisk (`*`), double quotation mark (`"`), question mark (`?`), vertical bar (`|`), less-than sign (<), or greater-than sign (>).

If you specify a relative path for the path name of the store data file output destination, the path is relative to the management directory of each EADS server.

(c) store-data-file-key

This option specifies the store data file key of a store data file.

A store data file key is expressed as a maximum of 32 single-byte characters. The permitted characters are alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`), underscores (`_`), and hyphens (`-`).

The prefix `eads_` or `eads_single_` and the suffix `_`*EADS-server-ID*`.`*extension* (where the extension is `.esd`) are added automatically to the store data file key.

If the store data file key is omitted, the command execution date and time become the store data file key, as shown in the following table:

-s or --single option	Store data file name	Generation management
Omitted	`eads_`*YYYYMMDDhhmmss*`_`*EADS-server-ID*`.esd`	Enabled
Specified	`eads_single_`*YYYYMMDDhhmmss*`_`*EADS-server-ID*`.esd`	Disabled

Legend:

YYYYMMDDhhmmss: Command execution date and time

YYYY: year, *MM*: month, *DD*: day, *hh*: hour (00 through 23), *mm*: minute, *ss*: second

(5) Return code

The following table lists the return codes that this subcommand returns.

Table 14–33: Return codes returned by the eztool export command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`110`		Connection establishment failed.
4	`111`		The command failed due to a communication timeout.

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
5	`120`		The command failed due to a syntax error.
6	`130`		The command failed because it could not be executed.
7	`131`		The command failed because another command was executing.
8	`150`		The command failed during execution.
9	`200`		The command failed due to a timeout.

(6) Notes

If a store data file key is specified in either of the formats shown below, the store data file might become subject to generation management. If you do not want to manage store data file generations, specify the store data file key in a different format.

- Format assumed when the store data file key is omitted

  `eads_`*YYYYMMDDhhmmss*`_`*EADS-server-ID*`.esd`

- Format of the store data file name that is output when the `eztool stop` command is executed

  `eads_stop_`*YYYYMMDDhhmmss*`_`*EADS-server-ID*`.esd`

Legend:

YYYYMMDDhhmmss: Command execution date and time

YYYY: year, *MM*: month, *DD*: day, *hh*: hour (00 through 23), *mm*: minute, *ss*: second

Note that if the store data file key is specified in the format of the store data file name that is output when the `eztool stop` command is executed, the store data file might be deleted. For details, see *7.6.2 Specifying the number of store data file generations*.

14.3.16 import (imports data)

(1) Description

This subcommand imports data from the store data files to which data has been exported from memory caches.

The subcommand relocates data by importing (`put`) the data from store data files.

(2) Rules

- You can execute this subcommand when the cluster's status is `AVAILABLE`.
- The target of this subcommand is the EADS servers whose cluster participation status is `online`. This subcommand cannot be executed if the cluster contains any EADS server whose cluster participation status is `standby`. You can determine the cluster participation status with the `eztool status` command.

- You can execute this subcommand when the target EADS servers are in `initialized` status.
- To prevent a full garbage collection (`FullGC`) from occurring while operations are underway, each EADS server performs `FullGC` when this subcommand terminates.
- While this subcommand has obtained a lock from the EADS server, the EADS server is not isolated. However, if a process is shut down or the EADS server terminates while the subcommand has the lock, the lock is released, and then the EADS server is isolated.

(3) Format

```
eztool import [-d path-name-of-store-data-file-storage-location]
              [--convertid EADS-server-ID-conversion-rule]
              [store-data-file-key]
```

(4) Options and arguments

(a) -d path-name-of-store-data-file-storage-location or --directory path-name-of-store-data-file-storage-location

This option specifies the path name of the store data file storage location.

Specify this option to import only the store data files located in the specified directory.

The path name cannot be a directory that contains an asterisk (`*`), double quotation mark (`"`), question mark (`?`), vertical bar (`|`), less-than sign (<), or greater-than sign (>).

If you specify a relative path for the path name of the store data file storage location, the path is relative to the management directory of each EADS server.

(b) --convertid EADS-server-ID-conversion-rule

Specify this option if you have grouped keys by specifying the EADS server ID of the storage location (EADS server ID specified groups are used) and if you want to convert the specified EADS server ID to another EADS server ID and import data.

Specify the EADS server ID conversion rule in the following format:

```
source-EADS-server-ID>target-EADS-server-ID
```

You can specify for the source EADS server ID and the target EADS server ID an integer in the range from `1` to `96` (a two digit integer beginning with zero, such as `01` and `02`, cannot be specified).

The target EADS server ID must differ from the source EADS server ID.

If you specify multiple EADS server ID conversion rules, delimit them with the comma. When multiple EADS server ID conversion rules are specified, the order in which the rules are specified has no effect on the priority. Note that the same source EADS server ID cannot be specified more than once.

The following characters and character strings are ignored:

- Comma at the beginning or at the end
  Example: `--convertid ,1>2,`

- Null character string delimited by commas or a character string consisting of only spaces that is delimited by commas
  Example: `--convertid 1>2,, ,`

(c) store-data-file-key

This option specifies the store data file key of a store data file.

Specify the store data file key of a store data file that was output by the `eztool export` or `eztool stop` command.

If the store data file key is omitted, the store data file with the most recent store data file key displayed in `latest` by the `eztool listesd` command is imported automatically.

A store data file with the following name will be imported:

Command used to output the store data file	Store data file name
`eztool export`	`eads_`*YYYYMMDDhhmmss*`_`*EADS-server-ID*`.esd`
`eztool stop`	`eads_stop_`*YYYYMMDDhhmmss*`_`*EADS-server-ID*`.esd`

Legend:

YYYYMMDDhhmmss: Command execution date and time
YYYY: year, *MM*: month, *DD*: day, *hh*: hour (00 through 23), *mm*: minute, *ss*: second

If an EADS server or the store data file storage directory contains multiple store data files, this subcommand automatically imports the store data file with the most recent command execution date and time.

If there are multiple store data file with the same command execution date and time (with only the EADS server ID being different), the subcommand imports all the applicable store data files.

If a store data file output by the `eztool export` command has the same command execution date and time as a store data file output by the `eztool stop` command, the subcommand imports the store data file output by the `eztool export` command.

(5) Return code

The following table lists the return codes that this subcommand returns.

Table 14–34: Return codes returned by the eztool import command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`110`		Connection establishment failed.
4	`111`		The command failed due to a communication timeout.
5	`120`		The command failed due to a syntax error.

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
6	130		The command failed because it could not be executed.
7	131		The command failed because another command was executing.
8	150		The command failed during execution.
9	200		The command failed due to a timeout.

(6) Notes

- The subcommand does not import a corrupted or invalid store data file.
- If data is imported from a store data file that was created by an EADS server whose version is earlier than 03-00, valid operation is not guaranteed.
- Processing of the subcommand might require a considerable amount of time depending on the number of data items and the amount of data because data items are added again individually.
- When data imported (`put`) from a store data file and data already in the cache have the same key name, the subcommand checks the key update dates and times and overwrites the data already in the cache only when the data's update date and time in the store data file are the more recent.
- When data with keys including EADS server ID specified groups is added and there is no range for a specified EADS server ID, the subcommand does not add that data. If this happens, a one-time warning message is issued.
- The total data restriction function is disabled while this subcommand is executing.

14.3.17 deleteesd (deletes store data files)

(1) Description

This subcommand deletes specified store data files from the cluster.

(2) Rules

- This subcommand can be executed only when the status of the cluster is one of the following:
    - Cluster available (`AVAILABLE`)
    - Cluster partially available (`PARTIALLY_AVAILABLE`)
- The target of this subcommand is the EADS servers whose cluster participation status is `online`. This subcommand cannot be executed if the cluster contains any EADS server whose cluster participation status is `standby`. You can determine the cluster participation status with the `eztool status` command.
- This subcommand can be executed when the target EADS servers are in the following status:
    - Initialized
    - Running
    - Closed

(3) Format

```
eztool deleteesd [-d path-name-of-store-data-file-storage-location]
                 store-data-file-key
```

(4) Options and arguments

(a) -d path-name-of-store-data-file-storage-location or --directory path-name-of-store-data-file-storage-location

This option specifies the path name of the store data file storage location.

Specify this option to delete only the store data files located in the specified directory.

The path name cannot be a directory that contains an asterisk (`*`), double quotation mark (`"`), question mark (`?`), vertical bar (`|`), less-than sign (<), or greater-than sign (>).

If you specify a relative path for the path name of the store data file storage location, the path is relative to the management directory of each EADS server.

(b) store-data-file-key

This option specifies the store data file key of the store data file (or files) to be deleted from the cluster.

Specify the store data file key of a store data file that was output by the `eztool export` or `eztool stop` command.

A store data file key is expressed as a maximum of 32 single-byte characters. The permitted characters are alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`), underscores (_), and hyphens (`-`).

(5) Return code

The following table lists the return codes that this subcommand returns.

Table 14–35: Return codes returned by the eztool deleteesd command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	0	0	Command execution was successful.
2	101	101	Initialization of the command failed.
3	110		Connection establishment failed.
4	111		The command failed due to a communication timeout.
5	120		The command failed due to a syntax error.
6	130		The command failed because it could not be executed.
7	131		The command failed because another command was executing.

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
8	`150`		The command failed during execution.
9	`200`		The command failed due to a timeout.

(6) Notes

All store data files with the specified store data file key are deleted.

For example, if this subcommand is executed with `20111101130203` specified as the store data file key, the subcommand deletes all files named `eads_20111101130203_[`*EADS-server-ID*`].esd` from each EADS server.

14.3.18 put (stores specified data)

(1) Description

This subcommand associates a specified value with a key, and then stores it.

This subcommand is used to test whether the configured execution environment is operational.

(2) Rules

You can execute this subcommand when the EADS server is running.

(3) Format

```
eztool put cache-name key value
```

(4) Options and arguments

(a) cache-name

Specify the name of the cache that stores the value.

The following characters are permitted for a cache name:

- If cache property files were not used to create caches
  A maximum of 32 single-byte characters in ASCII codes `0x20` to `0x7E`
- If cache property files were used to create caches
  A maximum of 32 single-byte alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`)

(b) key

Specify the key associated with the value that you are storing.

For details about the data that can be specified as keys, see *15.2.2(1) Data types that can be specified as keys*.

(c) value

Specify the value you want to store.

For *value*, you can specify a character string (`java.lang.String`) consisting of a maximum of 1,024 single-byte characters.

(5) Return code

The following table lists the return codes that this subcommand returns.

Table 14–36: Return codes returned by the eztool put command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`120`		The command failed due to a syntax error.
4	`150`		The command failed during execution.
5	`200`		The command failed due to a timeout.

(6) Notes

You can use this subcommand to manipulate data processed by an API function; conversely, you can use an API function to manipulate data processed by this subcommand.

Note that the data types and sizes permitted for *key* and *value* are not completely compatible with those permitted by the API functions. For example, this subcommand cannot store an object that is not a character string (`java.lang.String`) or a value that consists of more than 1,024 single-byte characters.

14.3.19 get (acquires specified data)

(1) Description

This subcommand acquires specified value.

This subcommand is used to test whether the configured execution environment is operational.

(2) Rules

You can execute this subcommand when the EADS server is running.

(3) Format

```
eztool get cache-name key
           [--format format-name]
```

```
            [--columns column-name[,column-name]...]
            [--filter filter-condition]
            [--match matching-condition]
```

(4) Options and arguments

(a) cache-name

Specify the name of the cache containing the value you want to acquire.

The following characters are permitted for a cache name:

- If cache property files were not used to create caches
  A maximum of 32 single-byte characters in ASCII codes `0x20` to `0x7E`
- If cache property files were used to create caches
  A maximum of 32 single-byte alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`)

(b) key

Specify the key associated with the value you want to acquire.

For details about the data that can be specified as keys, see *15.2.2(1) Data types that can be specified as keys*.

(c) --format format-name

For details about this option, see *14.4.2 How to specify the display format*.

(d) --columns column-name[,column-name]...

For details about this option, see *14.4.3 How to specify column filters*.

(e) --filter filter-condition

For details about this option, see *14.4.4 How to specify row filters*.

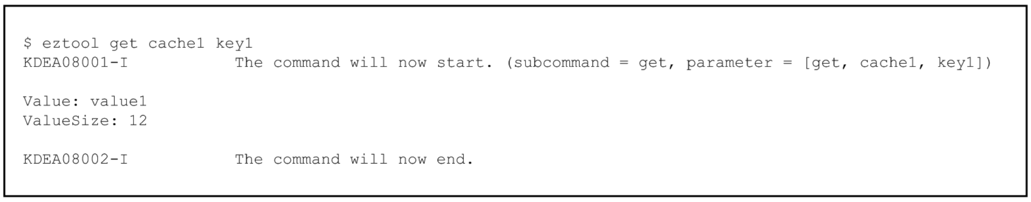
(f) --match matching-condition

For details about this option, see *14.4.5 How to specify a condition match*.

(5) Output example

The following shows an output example of the `eztool get` command's execution results.

For details about the components of the displayed information, see *14.4.1 Components of the displayed information*.



```
$ eztool get cache1 key1
KDEA08001-I           The command will now start. (subcommand = get, parameter = [get, cache1, key1])

Value: value1
ValueSize: 12

KDEA08002-I           The command will now end.
```

The following table lists and describes the summary display information.

Table 14–37: Summary information displayed by the eztool get command

No.	Summary name	Description
1	`Value`	Acquired value
2	`ValueSize`	Size of acquired value (in bytes)

(6) Return code

The following table lists the return codes that this subcommand returns.

Table 14–38: Return codes returned by the eztool get command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`120`		The command failed due to a syntax error.
4	`150`		The command failed during execution.
5	`200`		The command failed due to a timeout.

When the `--match` option is specified:

If the condition was satisfied, the subcommand returns `0`; otherwise, the subcommand returns `1`. If the command's execution failed, the return code is the same as when the `--match` option is not specified.

(7) Notes

- You can use this subcommand to manipulate data processed by an API function; conversely, you can use an API function to manipulate data processed by this subcommand.

  Note that the data types and sizes permitted for *key* and *value* are not completely compatible with those permitted by the API functions.
- The `--format`, `--columns`, and `--filter` options are ignored, if specified. These options are provided for future extension of functions.
- If the command times out during output processing, the output results might not be complete. The command might time out even if output processing is complete. If this happens, increase the command's timeout value, and then re-execute the command.

14.3.20 remove (deletes specified data)

(1) Description

This subcommand deletes a specified key and the value associated with the key.

This subcommand is used to test whether the configured execution environment is operational.

(2) Rules

You can execute this subcommand when the EADS server is running.

(3) Format

```
eztool remove cache-name key
```

(4) Options and arguments

(a) cache-name

Specify the name of the cache containing the value you want to delete.

The following characters are permitted for a cache names:

- If cache property files were not used to create caches
  A maximum of 32 single-byte characters in ASCII codes `0x20` to `0x7E`
- If cache property files were used to create caches
  A maximum of 32 single-byte alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`)

(b) key

Specify the key associated with the value you want to delete.

For details about the data that can be specified as keys, see *15.2.2(1) Data types that can be specified as keys*.

(5) Return code

The following table lists the return codes that this subcommand returns.

Table 14–39: Return codes returned by the eztool remove command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`120`		The command failed due to a syntax error.
4	`150`		The command failed during execution.
5	`200`		The command failed due to a timeout.

(6) Notes

You can use this subcommand to manipulate data processed by an API function; conversely, you can use an API function to manipulate data processed by this subcommand.

Note that the data types and sizes permitted for *key* and *value* are not completely compatible with those permitted by the API functions.

14.3.21 removeall (deletes all data)

(1) Description

This subcommand deletes a specified range of keys and all the values associated with those keys.

(2) Rules

- You can execute this subcommand when the EADS server is running.
- This subcommand executes in descending order of the EADS server positions (`Position`).

(3) Format

```
eztool removeall [-g group-name|-s] cache-name
```

(4) Options and arguments

(a) -g group-name or --group group-name

Specify this option to delete the values that belong to a specific group.

For details about the data that can be specified as group names, see *15.2.2(2) Data that can be specified as group names*.

(b) -s or --single

Specify this option to delete the values whose master copy (original) is located on the EADS server on which this subcommand is executed.

(c) cache-name

This option specifies the name of the cache that contain the values to be deleted.

The following characters are permitted for a cache name:

- If cache property files were not used to create caches
  A maximum of 32 single-byte characters in ASCII codes `0x20` to `0x7E`
- If cache property files were used to create caches
  A maximum of 32 single-byte alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`)

(5) Return code

Table 14–40: Return codes returned by the eztool removeall command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	0	0	Command execution was successful.
2	101	101	Initialization of the command failed.
3	120		The command failed due to a syntax error.
4	150		The command failed during execution.
5	200		The command failed due to a timeout.

(6) Notes

- Executing this subcommand enables you to delete all values in a specified range without having to create an application program. However, if you cannot obtain the intended results by using this subcommand's options and arguments, you will have to create an application program, such as in the following cases:
    - Specifying multiple keys to delete the values associated with each key
    - Implementing processing to be performed if the subcommand fails to delete some of the data
- If data is added while this subcommand is executing, the following occurs, depending on the area where the added data is stored:
    - If the added data is stored in the area from which data has already been deleted

      The added data is not deleted.
    - If the added data is stored in the area from which data has not been deleted

      The added data is deleted.
- If EADS servers are added to or restored in the cluster while this subcommand is executing, the added or restored EADS servers are excluded as processing targets. As a result, some of the data might not be deleted. If this occurs, re-execute the subcommand.

14.3.22 listfunc (displays which user functions are executable)

(1) Description

This subcommand displays which user functions are executable.

(2) Rules

- This subcommand can be executed only when the status of the cluster is one of the following:
    - Cluster available (AVAILABLE)
    - Cluster partially available (PARTIALLY_AVAILABLE)

- The target of this subcommand is the EADS servers whose cluster participation status is `online`. This subcommand cannot be executed if the cluster contains any EADS server whose cluster participation status is `standby`. You can determine the cluster participation status with the `eztool status` command.
- This subcommand can be executed when the target EADS servers are in the following status:
  - Initialized
  - Running
  - Closing
  - Closed

(3) Format

```
eztool listfunc [-v] [user-function-name]
                [--format format-name]
                [--columns column-name[,column-name]...]
                [--filter filter-condition]
                [--match matching-condition]
```

(4) Options and arguments

(a) -v or --verbose

Specify this option if you want to display the details of the command execution results.

(b) user-function-name

Specify the name of a user function whose executability you want to display.

Specify this option if you want to display information about only the specified user function.

A user function name can consist of alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`), underscores (_), periods (`.`), and dollar signs (`$`).

There is no limit to the number of characters.

(c) --format format-name

For details about this option, see *14.4.2 How to specify the display format*.

(d) --columns column-name[,column-name]...

For details about this option, see *14.4.3 How to specify column filters*.

(e) --filter filter-condition

For details about this option, see *14.4.4 How to specify row filters*.

(f) --match matching-condition

For details about this option, see *14.4.5 How to specify a condition match*.

(5) Output example

The following shows output examples of the `eztool listfunc` command's execution results.

For details about the components of the displayed information, see *14.4.1 Components of the displayed information*.

(a) If option -v or --verbose is omitted

```
$ eztool listfunc
KDEA08001-I           The command will now start. (subcommand = listfunc, parameter = [listfunc])

FunctionCount: 3

FunctionName  Enable  Disable
FunctionA          3        0
FunctionB          2        1
FunctionC          0        1
-----------------------------
KDEA08002-I           The command will now end.
$
```

The following tables list and describe the summary and content information that are displayed when the `-v` or `--verbose` option is omitted.

Table 14–41: Summary information displayed by the eztool listfunc command (-v or --verbose option omitted)

No.	Summary name	Description
1	`FunctionCount`	Total number of user functions This information is not displayed when a user function name is specified.

Table 14–42: Content information displayed by the eztool listfunc command (-v or --verbose option omitted)

No.	Column name	Description
1	`FunctionName`	User function name When a user function name is specified, only the specified user function name is displayed. Displayed in natural order.
2	`Enable`	Number of EADS servers that can execute the user function
3	`Disable`	Number of EADS servers that cannot execute the user function

(b) If option -v or --verbose is specified

```
$ eztool listfunc -v
KDEA08001-I           The command will now start. (subcommand = listfunc, parameter = [listfunc, -v])

FunctionCount: 3

Server                FunctionName  Status
-------------------------------------------
XX.XXX.XXX.168:24600  FunctionA     enable
XX.XXX.XXX.168:24600  FunctionB     enable
XX.XXX.XXX.168:24600  FunctionC     none

XX.XXX.XXX.169:24700  FunctionA     enable
XX.XXX.XXX.169:24700  FunctionB     enable
XX.XXX.XXX.169:24700  FunctionC     none

XX.XXX.XXX.170:24800  FunctionA     enable
XX.XXX.XXX.170:24800  FunctionB     disable
XX.XXX.XXX.170:24800  FunctionC     disable
-------------------------------------------
KDEA08002-I           The command will now end.
$
```

The following tables list and describe the summary and content information that are displayed when the `-v` or `--verbose` option is specified.

Table 14–43: Summary information displayed by the eztool listfunc command (-v or --verbose option specified)

No.	Summary name	Description
1	`FunctionCount`	Total number of user functions This information is not displayed when a user function name is specified.

Table 14–44: Content information displayed by the eztool listfunc command (-v or --verbose option specified)

No.	Column name	Description
1	`Server`	EADS server's IP address and the port number used to communicate with the EADS client. This information is displayed in the following format: *IP-address*`:`*port-number*
2	`FunctionName`	User function name When a user function name is specified, only the specified user function name is displayed.
3	`Status`	Whether the user function can be executed One of the following is displayed: • `enable` Can be executed. • `disable` Cannot be executed. If the specified user function does not exist, `none` is displayed.

Notes:

The execution results are displayed according to the following priorities:

1. Displayed in natural order of the `Server` values.
2. Rows with the same `Server` value are displayed in natural order of the `FunctionName` values.

(6) Return code

The following table lists the return codes that this subcommand returns.

Table 14–45: Return codes returned by the eztool listfunc command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`110`		Connection establishment failed.
4	`111`		The command failed due to a communication timeout.
5	`120`		The command failed due to a syntax error.
6	`130`		The command failed because it could not be executed.
7	`150`		The command failed during execution.
8	`200`		The command failed due to a timeout.

When the `--match` option is specified:

If the condition was satisfied, the subcommand returns `0`; otherwise, the subcommand returns `1`. If the command's execution failed, the return code is the same as when the `--match` option is not specified.

(7) Notes

If the command times out during output processing, the output results might not be complete. The command might time out even if output processing is complete. If this happens, increase the command's timeout value, and then re-execute the command.

14.3.23 execfunc (executes user functions)

(1) Description

This subcommand executes a specified user function.

You can use this subcommand to execute user functions without having to configure an EADS client for executing user functions.

(2) Rules

- You can execute this subcommand when the EADS server is running.
- The subcommand executes a specified user function on the EADS servers in descending order of the EADS server locations (`Position`). The subcommand continues processing even if execution of the user function fails on one of the EADS servers.

(3) Format

```
eztool execfunc [-k key|-g group-name|-s]
                cache-name user-function-name [user-function-arguments]
                [--format format-name]
                [--columns column-name[,column-name]...]
                [--filter filter-condition]
                [--match matching-condition]
```

(4) Options and arguments

(a) -k key or --key

Specify this option if you want to execute the user function on the EADS servers where a specified key is stored.

For details about the data that can be specified as keys, see *15.2.2(1) Data types that can be specified as keys*.

(b) -g group-name or --group group-name

Specify this option if you want to execute the user function on the EADS servers where a specified group is stored.

For details about the data that can be specified as group names, see *15.2.2(2) Data that can be specified as group names*.

(c) -s or --single

Specify this option if you want to execute the user function only on the EADS server where the command is executed.

(d) cache-name

Specify the name of the cache for the user function to be executed.

The following characters are permitted for a cache name:

- If cache property files were not used to create caches
  A maximum of 32 single-byte characters in ASCII codes 0x20 to 0x7E
- If cache property files were used to create caches
  A maximum of 32 single-byte alphanumeric characters (0 to 9, A to Z, and a to z)

(e) user-function-name

Specify the name of the user function to be executed.

A user function name can consist of alphanumeric characters (0 to 9, A to Z, a to z), underscores (_), periods (.), and dollar signs ($).

There is no limit to the number of characters.

(f) user-function-arguments

Specify this option if you want to pass specific arguments to the user function.

If this option is omitted, null is passed as the arguments.

You can specify for user function arguments a maximum of 1,024 of the single-byte characters `0x20` through `0x7E` in ASCII codes.

(g) --format format-name

For details about this option, see *14.4.2 How to specify the display format*.

(h) --columns column-name[,column-name]...

For details about this option, see *14.4.3 How to specify column filters*.

(i) --filter filter-condition

For details about this option, see *14.4.4 How to specify row filters*.

(j) --match matching-condition

For details about this option, see *14.4.5 How to specify a condition match*.

(5) Output example

The following shows output examples of the `eztool execfunc` command's execution results.

For details about the components of the displayed information, see *14.4.1 Components of the displayed information*.

```
$ eztool execfunc cache1 samplefunc.ExportCsvFunction export_test.csv
KDEA08001-I           The command will now start. (subcommand = execfunc, parameter = [execfunc, cache1,
samplefunc.ExportCsvFunction, export_test.csv])

Server                Result
XX.XXX.XXX.168:24600  Success
XX.XXX.XXX.168:24600  Success
XX.XXX.XXX.168:24600  Success
XX.XXX.XXX.168:24600  Success
XX.XXX.XXX.168:24600  Success
-----------------------------

KDEA08002-I           The command will now end.
```

The following table lists and describes the content information that is displayed.

Table 14–46: Content information displayed by the eztool execfunc command

No.	Column name	Description
1	`Server`	IP address of the EADS server that executed the user function and the EADS server's port number used to communicate with the EADS clients. This information is displayed in the following format: *IP address*`:`*port number* Displayed in natural order.
2	`Result`	If successful: `toString()` value in the execution result object of the user function If the value is `null`, character string `null` is output. If failed: Information about the exception that occurred

(6) Return code

The following table lists the return codes that this subcommand returns.

Table 14–47: Return codes returned by the eztool execfunc command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`120`		The command failed due to a syntax error.
4	`150`		The command failed during execution. If command's execution fails, the corresponding return code is set. If an exception occurs while the user function is running, the command is treated as being successful. For details about an exception, see the displayed `Result`.
5	`200`		The command failed due to a timeout.

When the `--match` option is specified:

If the condition was satisfied, the subcommand returns `0`; otherwise, the subcommand returns `1`. If the command's execution failed, the return code is the same as when the `--match` option is not specified.

(7) Notes

- Unlike for user functions that are executed normally on an EADS client, the user function arguments and acquired data of this subcommand are always of the character string data type (`java.lang.String`). If you will be using this command, take this into account when you create user functions.

  If large-sized character strings are used, garbage collection might occur on the EADS server, adversely affecting performance.
- If the command times out during output processing, the output results might not be complete. The command might time out even if output processing is complete. If this happens, increase the command's timeout value, and then re-execute the command.

14.3.24 listecf (displays a list of information about persistent data)

This subsection is applicable when you are using disk caches or two-way caches.

(1) Description

This subcommand displays a list of information about the persistent data in the cluster.

(2) Rules

- This subcommand can be executed only when the status of the cluster is one of the following:
  - Cluster available (`AVAILABLE`)
  - Cluster partially available (`PARTIALLY_AVAILABLE`)

- The target of this subcommand is the EADS servers whose cluster participation status is `online` or `offline`. This subcommand cannot be executed if the cluster contains any EADS server whose cluster participation status is `standby`. You can determine the cluster participation status with the `eztool status` command.
- This subcommand can be executed when the target EADS servers are in the following status:
  - Initialized
  - Running
  - Closing
  - Closed
  - Isolated
- When the `-s` or `--single` option is specified, the subcommand can be executed on an EADS server that is in any of the following statuses, regardless of the cluster's status:
  - Initialized
  - Running
  - Closing
  - Closed
  - Isolated

(3) Format

```
eztool listecf [-v [--divide number-of-segments]] [-s] [--threshold
threshold-value]
                [cache-name]
                [--format format-name]
                [--columns column-name[,column-name]...]
                [--filter filter-condition]
                [--match matching-condition]
```

(4) Options and arguments

(a) -v or --verbose

Specify this option if you want to display the details of the command's execution results.

(b) --divide number-of-segments

When the `-v` or `--verbose` option is specified, the distribution of the numbers of files in each compaction effect range is displayed in the command execution results. Use this option to specify a desired number of ranges in which to distribute the file counts.

You can specify an integer from `1` to `10` for the number of ranges.

If this option is omitted, the value of the `eads.command.compaction.effect.division` parameter in the command properties is assumed as the number of segments (default is `5`).

(c) -s or --single

Specify this option if you want to display the execution results for only the EADS server on which the command is executed.

(d) --threshold threshold-value

Specify this option if you want to display the number of cache data files whose compaction effects are at least a specified threshold (%).

You can specify for the threshold value an integer from `1` to `100`.

If this option is omitted, the value of the `eads.command.compaction.effect.threshold` parameter in the command properties is used as the threshold (default is `50`).

(e) cache-name

Specify this option if you want to display information about only the specified cache.

You can specify for the cache name a maximum of 32 single-byte alphanumeric characters (`0` to `9`, `A` to `Z`, `a` to `z`).

(f) --format format-name

For details about this option, see *14.4.2 How to specify the display format*.

(g) --columns column-name[,column-name]...

For details about this option, see *14.4.3 How to specify column filters*.

(h) --filter filter-condition

For details about this option, see *14.4.4 How to specify row filters*.

(i) --match matching-condition

For details about this option, see *14.4.5 How to specify a condition match*.

(5) Output example

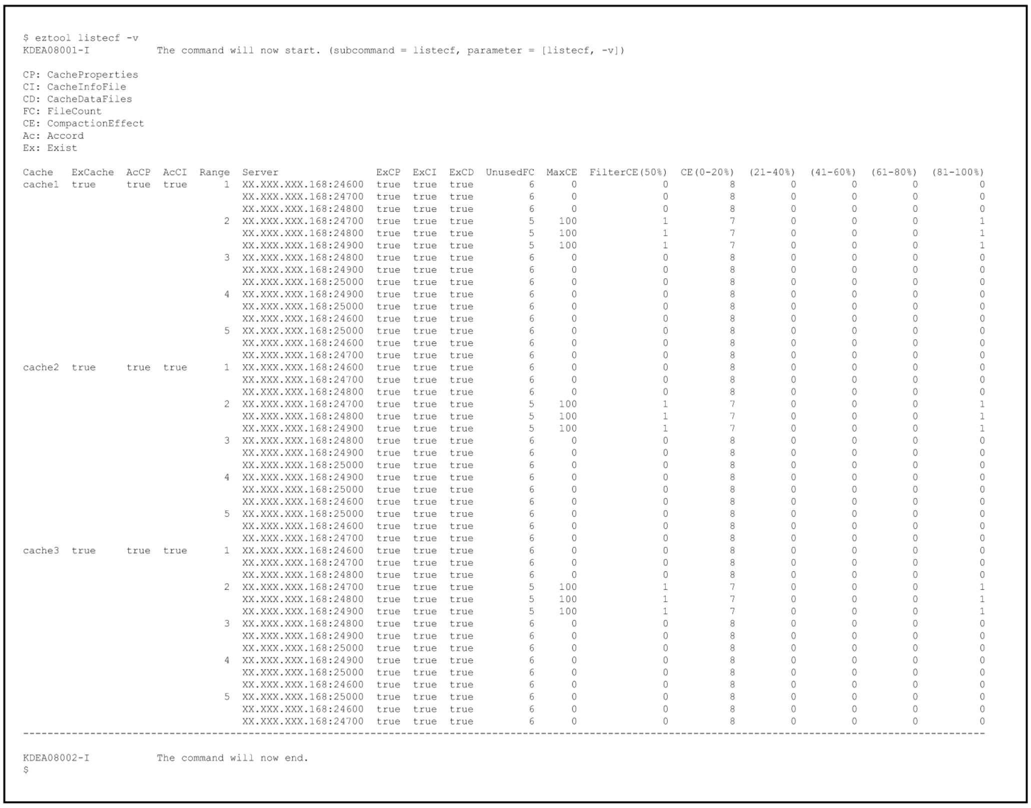
The following shows output examples of the `eztool listecf` command's execution results.

For details about the components of the displayed information, see *14.4.1 Components of the displayed information*.

■ If option `-v` or `--verbose` is omitted

```
$ eztool listecf
KDEA08001-I           The command will now start. (subcommand = listecf,
parameter = [listecf])

FC: FileCount
CE: CompactionEffect
Ex: Exist

Cache   ExCache  Range  Server                UnusedFC  MaxCE  FilterCE(50%)
cache1  true         1  XX.XXX.XXX.168:24600         6      0              0
                        XX.XXX.XXX.168:24700         6      0              0
                        XX.XXX.XXX.168:24800         6      0              0
                     2  XX.XXX.XXX.168:24700         6      0              0
                        XX.XXX.XXX.168:24800         6      0              0
                        XX.XXX.XXX.168:24900         6      0              0
                     3  XX.XXX.XXX.168:24800         6      0              0
                        XX.XXX.XXX.168:24900         6      0              0
                        XX.XXX.XXX.168:25000         6      0              0
                     4  XX.XXX.XXX.168:24900         6      0              0
                        XX.XXX.XXX.168:25000         6      0              0
                        XX.XXX.XXX.168:24600         6      0              0
                     5  XX.XXX.XXX.168:25000         6      0              0
                        XX.XXX.XXX.168:24600         6      0              0
                        XX.XXX.XXX.168:24700         6      0              0
cache2  true         1  XX.XXX.XXX.168:24600         6      0              0
                        XX.XXX.XXX.168:24700         6      0              0
                        XX.XXX.XXX.168:24800         6      0              0
                     2  XX.XXX.XXX.168:24700         6      0              0
                        XX.XXX.XXX.168:24800         6      0              0
                        XX.XXX.XXX.168:24900         6      0              0
                     3  XX.XXX.XXX.168:24800         6      0              0
                        XX.XXX.XXX.168:24900         6      0              0
                        XX.XXX.XXX.168:25000         6      0              0
                     4  XX.XXX.XXX.168:24900         6      0              0
                        XX.XXX.XXX.168:25000         6      0              0
                        XX.XXX.XXX.168:24600         6      0              0
                     5  XX.XXX.XXX.168:25000         6      0              0
                        XX.XXX.XXX.168:24600         6      0              0
                        XX.XXX.XXX.168:24700         6      0              0
cache3  true         1  XX.XXX.XXX.168:24600         6      0              0
                        XX.XXX.XXX.168:24700         6      0              0
                        XX.XXX.XXX.168:24800         6      0              0
                     2  XX.XXX.XXX.168:24700         6      0              0
                        XX.XXX.XXX.168:24800         6      0              0
                        XX.XXX.XXX.168:24900         6      0              0
                     3  XX.XXX.XXX.168:24800         6      0              0
                        XX.XXX.XXX.168:24900         6      0              0
                        XX.XXX.XXX.168:25000         6      0              0
                     4  XX.XXX.XXX.168:24900         6      0              0
                        XX.XXX.XXX.168:25000         6      0              0
                        XX.XXX.XXX.168:24600         6      0              0
                     5  XX.XXX.XXX.168:25000         6      0              0
                        XX.XXX.XXX.168:24600         6      0              0
                        XX.XXX.XXX.168:24700         6      0              0
----------------------------------------------------------------------------

KDEA08002-I           The command will now end.
$
```

■ If option `-v` or `--verbose` is specified



```
$ eztool listecf -v
KDEA08001-I          The command will now start. (subcommand = listecf, parameter = [listecf, -v])

CP: CacheProperties
CI: CacheInfoFile
CD: CacheDataFiles
FC: FileCount
CE: CompactionEffect
Ac: Accord
Ex: Exist

Cache   ExCache  AcCP  AcCI  Range  Server                ExCP  ExCI  ExCD  UnusedFC  MaxCE  FilterCE(50%)  CE(0-20%)  (21-40%)  (41-60%)  (61-80%)  (81-100%)
cache1  true     true  true      1  XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24700  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24800  true  true  true         6      0              0          8         0         0         0          0
                                 2  XX.XXX.XXX.168:24700  true  true  true         5    100              1          7         0         0         0          1
                                    XX.XXX.XXX.168:24800  true  true  true         5    100              1          7         0         0         0          1
                                    XX.XXX.XXX.168:24900  true  true  true         5    100              1          7         0         0         0          1
                                 3  XX.XXX.XXX.168:24800  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24900  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                 4  XX.XXX.XXX.168:24900  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                 5  XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24700  true  true  true         6      0              0          8         0         0         0          0
cache2  true     true  true      1  XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24700  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24800  true  true  true         6      0              0          8         0         0         0          0
                                 2  XX.XXX.XXX.168:24700  true  true  true         5    100              1          7         0         0         0          1
                                    XX.XXX.XXX.168:24800  true  true  true         5    100              1          7         0         0         0          1
                                    XX.XXX.XXX.168:24900  true  true  true         5    100              1          7         0         0         0          1
                                 3  XX.XXX.XXX.168:24800  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24900  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                 4  XX.XXX.XXX.168:24900  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                 5  XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24700  true  true  true         6      0              0          8         0         0         0          0
cache3  true     true  true      1  XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24700  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24800  true  true  true         6      0              0          8         0         0         0          0
                                 2  XX.XXX.XXX.168:24700  true  true  true         5    100              1          7         0         0         0          1
                                    XX.XXX.XXX.168:24800  true  true  true         5    100              1          7         0         0         0          1
                                    XX.XXX.XXX.168:24900  true  true  true         5    100              1          7         0         0         0          1
                                 3  XX.XXX.XXX.168:24800  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24900  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                 4  XX.XXX.XXX.168:24900  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                 5  XX.XXX.XXX.168:25000  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24600  true  true  true         6      0              0          8         0         0         0          0
                                    XX.XXX.XXX.168:24700  true  true  true         6      0              0          8         0         0         0          0
--------------------------------------------------------------------------------------------------------------------------------------------------------------

KDEA08002-I          The command will now end.
$
```

The following tables list and describe the summary and content information that are displayed.

Table 14–48: Summary information displayed by the eztool listecf command

No.	Summary name	Description	Whether displayed		
			-v or --verbose option		Specification of -s or --single option
			Omitted	Specified	
1	`CP: CacheProperties`	`CP` is the abbreviation for a cache property file (`CacheProperties`).	N	Y	N
2	`CI: CacheInfoFile`	`CI` is the abbreviation for a cache information file (`CacheInfoFile`).	N	Y	N
3	`CD: CacheDataFiles`	`CD` is the abbreviation for cache data files (`CacheDataFiles`).	N	Y	N
4	`FC: FileCount`	`FC` is the abbreviation for the number of files (`FileCount`).	Y	Y	Y
5	`CE: CompactionEffect`	`CE` is the abbreviation for compaction effect (`CompactionEffect`).	Y	Y	Y
6	`Ac: Accord`	`Ac` is the abbreviation for match (`Accord`).	N	Y	N
7	`Ex: Exist`	`Ex` is the abbreviation for exist (`Exist`).	Y	Y	N

Legend:

Y: Displayed

N: Not displayed

Table 14–49: Content information displayed by the eztool listecf command

No.	Column name	Description	Whether displayed		
			-v or --verbose option		Specification of -s or --single option
			Omitted	Specified	
1	`Cache`	Cache name.[#1] Names of disk caches and two-way caches are extracted from the following cache information and then displayed: • Caches existing on the EADS servers • Caches specified in the cache property files	Y	Y	Y
2	`ExCache`	Whether the cache exists on the EADS server.[#1] One of the following is displayed: • `true` The cache exists on the EADS server. • `false` The cache does not exist on the EADS server.	Y	Y	N
3	`AcCP`	Whether the cache property files match among the EADS servers.[#1] This item displays whether the cache property files exist and whether they match among all EADS servers. One of the following is displayed: • `true` Match. • `false` Do not match.	N	Y	N
4	`AcCI`	Whether the cache information files match among the EADS servers.[#1] This item displays whether the cache information files match, ignoring EADS servers that do not have cache information files. One of the following is displayed: • `true` Match. • `false` Do not match.	N	Y	N
5	`Range`	Range ID.[#1] Number (integer `1` to `96`) used to identify a range in the cache. The range ID matches the server ID of the EADS server on which the data is stored.	Y	Y	Y
6	`Server`	IP address of the EADS server that is included in `Range` and the EADS server's port number used to communicate with the EADS clients. This information is displayed in the following format: *IP address*`:`*port number*	Y	Y	N
7	`ExCP`	Whether the cache property file exists. One of the following is displayed: • `true` Exists.	N	Y	N

No.	Column name	Description	Whether displayed		
			-v or --verbose option		Specification of -s or --single option
			Omitted	Specified	
		- `false` Does not exist. Nothing is displayed if the EADS server is not running or a communication error has occurred.			
8	`ExCI`	Whether the cache information file exists. One of the following is displayed: - `true` Exists. - `false` Does not exist. Nothing is displayed in the following cases: - The EADS server is not running. - A communication error has occurred. - The cache has not been imported to the EADS server and `ExCP` is `false`. - The cache has not been imported to the EADS server and the parameter required for acquiring `ExCI` information is not specified in the cache property file.	N	Y	N
9	`ExCD`	Whether the cache data file exists. One of the following is displayed: - `true` Exists. - `false` Does not exist. Nothing is displayed in the following cases: - The EADS server is not running. - A communication error has occurred. - The cache has not been imported to the EADS server and `ExCP` is `false`. - The cache has not been imported to the EADS server and the parameter required for acquiring `ExCD` information is not specified in the cache property file.	N	Y	N
10	`UnusedFC`	Number of unused files. Nothing is displayed in the following cases: - The EADS server is not running. - A communication error has occurred. - The cache does not exist on the EADS server.	Y	Y	Y
11	`MaxCE`	Maximum compaction effects on the cache data files in the range.[#2] Nothing is displayed in the following cases: - The EADS server is not running. - A communication error has occurred. - The cache does not exist on the EADS server.	Y	Y	Y
12	`FilterCE(`*x*`%)`[#3]	Number of files whose compaction effects are equal to or greater than the threshold value.[#4]	Y	Y	Y

No.	Column name	Description	Whether displayed		
			-v or --verbose option		Specification of -s or --single option
			Omitted	Specified	
		Nothing is displayed in the following cases: • The EADS server is not running. • A communication error has occurred. • The cache does not exist on the EADS server.			
13	`CE(`*x*`-`*y*`%)`[#5]	Distribution of the numbers of files in each compaction effects range.[#4] Nothing is displayed in the following cases: • The EADS server is not running. • A communication error has occurred. • The cache does not exist on the EADS server.	N	Y	Y (displayed only when the `-v` or `--verbose` option is specified)

Legend:

Y: Displayed

N: Not displayed

Notes:

The execution results are displayed according to the following priorities:

1. Displayed in natural order of the `Cache` values.
2. Rows with the same `Cache` value are displayed in ascending order of the `Range` values.
3. Rows with the same `Cache` and `Range` values are displayed in the order of the `Server` values (in the order of EADS server that stores data and EADS server from which data was copied). This does not apply if the `-s` or `--single` option is specified.

#1

If the execution results contain multiple consecutive cells with the same value, only the first such cell is displayed and the other cells are omitted.

#2

All digits following the decimal point are discarded from the value indicating the compaction effects.

#3

The value set as the threshold is displayed for *x*.

#4

The number of files is totaled, with digits following the decimal point discarded from the value indicating compaction effects.

#5

The range obtained from the number of segments is displayed. For columns starting in 2, only `(`*x*`-`*y*`%)` is displayed as the range.

(6) Return code

The following table lists the return codes that this subcommand returns.

Table 14–50: Return codes returned by the eztool listecf command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	0	0	Command execution was successful.
2	101	101	Initialization of the command failed.
3	110		Connection establishment failed.
4	111		The command failed due to a communication timeout.
5	120		The command failed due to a syntax error.
6	130		The command failed because it could not be executed.
7	150		The command failed during execution.
8	200		The command failed due to a timeout.

When the `--match` option is specified:

> If the condition was satisfied, the subcommand returns `0`; otherwise, the subcommand returns `1`. If the command's execution failed, the return code is the same as when the `--match` option is not specified.

(7) Notes

If the command times out during output processing, the output results might not be complete. The command might time out even if output processing is complete. If this happens, increase the command's timeout value, and then re-execute the command.

14.3.25 resume (resumes caches)

This subsection is applicable when you are using disk caches or two-way caches.

(1) Description

This subcommand resumes disk caches and two-way caches.

When this subcommand is executed, the caches that have both cache property files and cache information files are resumed.

Use this subcommand to resume caches in the following cases:

- For restarting the EADS servers after the cluster was stopped normally
- For restoring the cluster after it has been in unavailable (`NOT_AVAILABLE`) or partially available (`PARTIALLY_AVAILABLE`) status

(2) Rules

- This subcommand can be executed only when the status of the cluster is the following:

    - Cluster available (`AVAILABLE`)

- The target of this subcommand is the EADS servers whose cluster participation status is `online`. This subcommand cannot be executed if the cluster contains any EADS server whose cluster participation status is `offline` or `standby`. You can determine the cluster participation status with the `eztool status` command.
- This subcommand can be executed when the target EADS servers are in the following status:
    - Initialized
- To prevent a full garbage collection (`FullGC`) from occurring while operations are underway, each EADS server performs `FullGC` when this subcommand terminates.
- The EADS servers are not isolated while this subcommand has obtained a lock from the EADS servers. However, if the process is shut down or an EADS server terminates while the subcommand has the lock, the lock is released, and then the EADS server becomes isolated.

(3) Format

```
eztool resume
```

(4) Return code

The following table lists the return codes that this subcommand returns.

Table 14–51: Return codes returned by the eztool resume command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	0	0	Command execution was successful.
2	101	101	Initialization of the command failed.
3	110		Connection establishment failed.
4	111		The command failed due to a communication timeout.
5	120		The command failed due to a syntax error.
6	130		The command failed because it could not be executed.
7	131		The command failed because another command was executing.
8	150		The command failed during execution.
9	200		The command failed due to a timeout.

(5) Notes

- Make sure that the properties of the caches that will not be resumed are also identical on all EADS servers that make up the cluster (except for the `eads.cache.disk.info.dir` and `eads.cache.disk.`*`n`*`.dir` parameters).
- If the cluster was not stopped normally, deleted data might be restored.

- If the cache resume processing terminates prematurely for a reason such as abnormal termination or a command timeout, locks are not released. If locks have not been released after an attempt has been made to resume caches, data might no longer be consistent. If you continue operation in such a status, data might become corrupted or lost. To avoid this, take the following steps:
    1. Execute the `eztool status -v` command to check the cluster status.
    2. Execute the `eztool unlock` command to release the lock.
    3. Execute the `eztool listcache` command to check the list of caches.
    4. If the cache resume processing has failed, execute the `eztool deletecache` command to delete the caches.
    5. If EADS servers are isolated, restore the EADS servers.
    6. Perform cache resume processing again.
- If the cache resume processing has failed, check the following items and then perform the cache restart processing again:
    - Check for errors in the parameters specified in the cluster property files or shared property files.
    - Check for errors in the parameters specified in the cache property files.
    - Check the cache files for any invalid status.
- The total data restriction function is disabled while this subcommand is executing.

14.3.26 importecf (relocates persistent data)

This subsection is applicable when you are using disk caches or two-way caches.

(1) Description

This subcommand relocates persistent data by importing (`put`) data from cache data files and cache index files located in a specified directory.

When you add EADS servers to the cluster, delete EADS servers from the cluster, or change the multiplicity and size of data, you can use this subcommand to relocate data from cache data files.

(2) Rules

- This subcommand can be executed only when the status of the cluster is the following:
    - Cluster available (`AVAILABLE`)
- The target of this subcommand is the EADS servers whose cluster participation status is `online`. This subcommand cannot be executed if the cluster contains any EADS server whose cluster participation status is `standby`. You can determine the cluster participation status with the `eztool status` command.
- This subcommand can be executed when the target EADS servers are in the following status:
    - Initialized
- To prevent a full garbage collection (`FullGC`) from occurring while operations are underway, each EADS server performs `FullGC` when this subcommand terminates.
- An EADS server is not isolated while this subcommand has obtained a lock from the EADS server. However, if the process is shut down or the EADS server terminates while the subcommand has the lock, the lock is released, and then the EADS server becomes isolated.

(3) Format

```
eztool importecf [--convertid EADS-server-ID-conversion-rule]
                 path-name-of-storage-for-cache-data-files-and-cache-index-
files
```

(4) Options and arguments

(a) --convertid EADS-server-ID-conversion-rule

Specify this option if you have grouped keys by specifying the EADS server IDs of the storage locations (EADS server ID specified groups are used) and you want to convert the specified EADS server IDs to other EADS server IDs and import the data.

Specify each EADS server ID conversion rule in the following format:

```
source-EADS-server-ID>target-EADS-server-ID
```

For the source EADS server ID and the target EADS server ID, you can specify an integer in the range from `1` to `96` (a two-digit integer beginning with zero, such as `01` and `02`, cannot be specified).

The target EADS server ID must differ from the source EADS server ID.

If you specify multiple EADS server ID conversion rules, delimit them with the comma. When multiple EADS server ID conversion rules are specified, the order in which the rules are specified has no effect on the priority. Note that the same source EADS server ID cannot be specified more than once.

The following characters and character strings are ignored:

- Comma specified at the beginning or at the end
  Example: `--convertid ,1>2,`
- Null character string delimited by commas or a character string consisting only of spaces that is delimited by commas
  Example: `--convertid 1>2,,  ,`

(b) path-name-of-storage-for-cache-data-files-and-cache-index-files

This option specifies the path name of the storage location for the cache data files and cache index files that you want to import.

The path name cannot be a directory that contains an asterisk (`*`), double quotation mark (`"`), question mark (`?`), vertical bar (`|`), less-than sign (<), or greater-than sign (>).

If a relative path is specified as the path of the cache data file and cache index file storage location, it is treated as being relative to the management directory.

(5) Return code

The following table lists the return codes that this subcommand returns.

Table 14–52: Return codes returned by the eztool importecf command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`110`		Connection establishment failed.
4	`111`		The command failed due to a communication timeout.
5	`120`		The command failed due to a syntax error.
6	`130`		The command failed because it could not be executed.
7	`131`		The command failed because another command was executing.
8	`150`		The command failed during execution.
9	`200`		The command failed due to a timeout.

(6) Notes

- Output directories for cache information files, cache index files, and cache data files cannot be specified.
- When this subcommand is executed, all cache persistent data files and cache index files in the directory subject to import processing are imported.
- If the cache data files and cache index files that are to be imported are corrupted or were open when a failure occurred, deleted data might be restored.
- If an error occurs while the subcommand is processing a set of a cache data file and a cache index file subject to import processing, the subcommand processes the next set of a cache data file and a cache index file. In such a case, the subcommand results in an error. If this happens, execute the following commands to determine the problem:
    - Execute the `eztool status` command to check the cluster status.
    - Execute the `eztool listcache` command to check the list of caches.
- This subcommand's processing might require a considerable amount of time depending on the number of data items and the amount of data, because data items are added again individually.
- If data with the key to be imported (`put`) already exists, the subcommand checks the key update dates and times and overwrites the data only when the update date and time of the key to be imported is the more recent.
- When data with keys including EADS server ID specified groups is added and there is no range for a specified EADS server ID, the subcommand does not add that data. If this happens, a one-time warning message is issued.
- The total data restriction function is disabled while this subcommand is executing.

14.3.27 deleteecf (deletes cache files)

This subsection is applicable when you are using disk caches or two-way caches.

(1) Description

This subcommand deletes the cache files for a specified cache.

When this subcommand is executed, the following cache files are deleted in accordance with parameters specified in the cache property file on each EADS server:

- *directory-specified-in-the-*eads.cache.disk.info.dir*-parameter-in-the-cache-properties*/*cache-name*/
  `eads_info_[`*EADS-server-ID*[#]`]_[`*cache-name*`].ecf`
  #: Integer from `01` to `96`
- *directory-specified-in-the-*eads.cache.disk.info.dir*-parameter-in-the-cache-properties*/*cache-name*/
  `eads_index_[`*EADS-server-ID*[#1]`]_[`*cache-name*`]_[`*range-ID*[#1]`]_[`*nnnnn*[#2]`].ecf`
  #1: Integer from `01` to `96`
  #2: Sequential file number (5-digit integer)
- *directory-specified-in-the-*eads.cache.disk.n.dir*-parameter-in-the-cache-properties*/*cache-name*/
  `eads_data_[`*EADS-server-ID*[#1]`]_[`*cache-name*`]_[`*range-ID*[#1]`]_[`*nnnnn*[#2]`].ecf`
  #1: Integer from `01` to `96`
  #2: Sequential file number (5-digit integer)

The following directories are also deleted if they are empty after the cache files have been deleted:

- *directory-specified-in-the-*eads.cache.disk.info.dir*-parameter-in-the-cache-properties*/*cache-name*
- *directory-specified-in-the-*eads.cache.disk.n.dir*-parameter-in-the-cache-properties*/*cache-name*

(2) Rules

- This subcommand can be executed only when the status of the cluster is one of the following:
  - Cluster available (`AVAILABLE`)
  - Cluster partially available (`PARTIALLY_AVAILABLE`)
- The target of this subcommand is the EADS servers whose cluster participation status is `online`. This subcommand cannot be executed if the cluster contains any EADS server whose cluster participation status is `standby`. You can determine the cluster participation status with the `eztool status` command.
- This subcommand can be executed when the target EADS servers are in the following status:
  - Initialized
  - Running
  - Closed

  If the `-l` or `--local` option is specified, the subcommand can be executed only when the local EADS server is stopped.

(3) Format

```
eztool deleteecf cache-name [-l]
```

(4) Options and arguments

(a) cache-name

This option specifies the cache name for the cache files to be deleted.

Specify the cache name as a maximum of 32 single-byte alphanumeric characters (`0` to `9`, `A` to `Z`, `a` to `z`).

(b) -l or --local

Specify this option if you want to delete the cache files only on the EADS server on which the command is executed.

For example, specify this option when a system in which data was updated and deleted frequently has become isolated and you want to delete cache files that are not needed for restoration before performing restoration in order to reduce the time required for importing data.

When this option is specified, the cache files are deleted only if the target EADS server is stopped. The cache files are not deleted while the target EADS server is running.

(5) Return code

The following table lists the return codes that this subcommand returns.

Table 14–53: Return codes returned by the eztool deleteecf command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`110`		Connection establishment failed.
4	`111`		The command failed due to a communication timeout.
5	`120`		The command failed due to a syntax error.
6	`130`		The command failed because it could not be executed.
7	`131`		The command failed because another command was executing.
8	`150`		The command failed during execution.
9	`200`		The command failed due to a timeout.

(6) Notes

- Cache files cannot be deleted while they are being used by EADS servers. If there is no cache property file for the specified cache, the cache files cannot be deleted.

14.3.28 compaction (performs compaction on cache data files)

This subsection is applicable when you are using disk caches or two-way caches.

(1) Description

This subcommand performs compaction on the cache data files on the EADS server on which the subcommand is executed.

(2) Rules

- This subcommand can be executed on an EADS server that is in any of the following statuses, regardless of the cluster's status:
  - Initialized
  - Running
  - Closed
  - Isolated
- The subcommand selects the cache data files that will yield the greatest compaction effects on the EADS server on which the subcommand is executed. If there are multiple candidate files, the subcommand processes them in descending order of the compaction effects.

(3) Format

```
eztool compaction [--cache cache-name [--range range-ID]]
                  [--limit execution-count|--unused_fc number-of-unused-
files]
                  [--threshold threshold-value]
```

```
eztool compaction [--break]
```

(4) Options and arguments

(a) --cache cache-name

This option specifies the name of the cache on which compaction is to be performed.

Specify for the cache name a maximum of 32 single-byte alphanumeric characters (0 to 9, A to Z, a to z).

If this option is omitted, compaction is performed on all caches.

(b) --range range-ID

A range ID is a number (integer 1 to 96) used to identify a range in a cache. The range ID matches the server ID of the EADS server on which the data is stored.

This option specifies the range of compaction processing. If no data belongs to the range with the specified range ID on the EADS server on which the subcommand is executed, compaction processing will not be performed.

If this option is omitted, all ranges of the specified cache are subject to compaction processing.

(c) --limit execution-count

This option specifies the maximum compaction count (maximum number of cache data files on which compaction is to be performed by one execution of the command). If there are no more target files, the subcommand terminates even before the actual count reaches the specified value.

Specify for the execution count an integer from 1 to 20971520.

If this option is omitted, the default is 20971520.

(d) --unused_fc number-of-unused-files

Specify this option if you want to perform compaction processing until the number of unused cache data files reaches the specified value. If the number of unused files is already equal to or greater than the specified value in all ranges, compaction is not performed. If there are no more target files or the compaction count has reached 20,971,520, the subcommand terminates with a warning even though the number of unused files has not reached the specified value.

For the number of unused files, you can specify an integer from 1 to 32766.

(e) --threshold threshold-value

Specify this option if you want to perform compaction on the cache data files that will yield at least the effects specified as the threshold value (%). If there are no cache data files that will yield the specified level of effects, compaction will not be performed.

Specify for the threshold value an integer from 1 to 100.

If this option is omitted, the value of the eads.command.compaction.effect.threshold parameter in the command properties is assumed as the threshold value (default is 50).

(f) --break

Specify this option to stop compaction processing.

You can stop compaction processing in units of files. Compaction processing cannot be stopped in the middle of a file.

(5) Return code

The following table lists the return codes that this subcommand returns.

Table 14–54: Return codes returned by the eztool compaction command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	0	0	Command execution was successful.
2	1	1	Compaction was not performed because there were no target files.
3	2	2	Compaction was not performed because the number of unused files was equal to or greater than the value specified in the --unused_fc option.

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
4	`3`	`3`	There were no more target files before the number of unused files reached the value specified in the `--unused_fc` option. Compaction processing has been completed.
5	`10`	`10`	The `--break` option was specified, but compaction was not being performed.
6	`11`	`11`	The `--break` option was specified, but cancellation of compaction processing had already been queued.
7	`101`	`101`	Initialization of the command failed.
8	`110`		Connection establishment failed.
9	`111`		The command failed due to a communication timeout.
10	`120`		The command failed due to a syntax error.
11	`130`		The command failed because it could not be executed.
12	`131`		The command failed because another command was executing.
13	`150`		The command failed during execution.
14	`200`		The command failed due to a timeout.

(6) Notes

- This subcommand can be executed while the EADS servers are running, but performance might be affected adversely.
- If cancellation of compaction processing is queued and this subcommand times out or its processing stops, the process waiting for cancellation is not completed. If this happens, either wait until the process times out or terminate the process.
- No compaction is underway between the time compaction on one file is completed and compaction on the next file begins. If an attempt is made during this interval to stop compaction processing by specifying the `--break` option, a message indicating that compaction processing is not in progress might be displayed, prohibiting the compaction processing from being stopped. If this happens, repeat execution of the `eztool compaction --break` command until compaction processing stops.

14.3.29 threaddump (outputs a thread dump)

(1) Description

This subcommand outputs an EADS server thread dump.

A thread dump is output to the directory specified in the `eads.logger.dir` parameter in the server properties.

If the eads.logger.dir parameter is not specified in the server properties, thread dumps are output to *management-directory*/logs.

(2) Rules

- This subcommand can be executed regardless of the cluster's status.
- This subcommand can be executed when the EADS server is in the following status:
    - Initializing
    - Initialized
    - Running
    - Closing
    - Closed
    - Isolated
    - Stopping

(3) Format

```
eztool threaddump
```

(4) Return code

1: No EADS server was found.

Other than 1: Termination code of the jheapprof command in JavaSE

14.3.30 snapshot (collects logs, settings, hardware information, and network information)

(1) Description

This subcommand collects the files listed below and saves them as archive files.

Note that files with the extensions .tar, .tar.gz, and .tgz are excluded.

- EADS log information
    - Active EADS servers' thread dumps
    - Files in the directory specified in the eads.logger.dir parameter in the server properties
      Note that eads_command_* files (*: any string of zero or more characters) are excluded.
    - eads_command_* files (*: any string of zero or more characters) directly under the directory specified in the eads.command.logger.dir parameter in the command properties
    - eads_command_* files (*: any string of zero or more characters) directly under the maintenance directory in the directory specified in the eads.command.logger.dir parameter in the command properties
    - *management-directory*/hs_err_pid*.log (*: any string of zero or more characters)

Note: This is an output file name for a JavaVM error report file.

- EADS settings
    - Function property files (property files under *management-directory*/app)
    - Property files (files under *management-directory*/conf)
    - Management directory configuration information (*management-directory*/logs/snapshot_info/eads_info/eads_info_ls.txt)
    - List of cache data files (*management-directory*/logs/snapshot_info/cache_file_info/eads_data_*cache-name_cache-property-file-sequence-number*.txt)
    - List of cache information files (*management-directory*/logs/snapshot_info/cache_file_info/eads_info *cache-name*.txt)
    - List of cache index files (*management-directory*/logs/snapshot_info/cache_file_info/eads_index_*cache-name*.txt)
- Hardware and network information
    - OS version information (*management-directory*/logs/snapshot_info/eads_info/eads_info_version.txt)
    - CPU information (*management-directory*/logs/snapshot_info/eads_info/eads_info_cpu.txt)
    - Memory information (*management-directory*/logs/snapshot_info/eads_info/eads_info_memory.txt)
    - Disk information (*management-directory*/logs/snapshot_info/eads_info/eads_info_df.txt)
    - Network connection information (*management-directory*/logs/snapshot_info/eads_info/eads_info_netstat.txt)
    - Kernel parameter net.core information (*management-directory*/logs/snapshot_info/eads_info/eads_info_netcore.txt)
    - Correspondence between host IPs and IP addresses (/etc/hosts file)
    - Network interface information (*management-directory*/logs/snapshot_info/eads_info/eads_info_ifconfig.txt)
    - System resources limitation information (*management-directory*/logs/snapshot_info/eads_info/eads_info_ulimit.txt)

Note that a gz file is output under the directory specified in the eads.logger.dir parameter in the server properties.

If the eads.logger.dir parameter is not specified in the server properties, the file is output under *management-directory*/logs.

A file name is in the following format:

snapshot_[*EADS-server-name*]_[*YYYYMMDDhhmmss*].tar.gz

Legend:

YYYYMMDDhhmmss: Command execution date and time

YYYY: year, *MM*: month, *DD*: day, *hh*: hour (00 through 23), *mm*: minute, *ss*: second

(2) Rules

- This subcommand can be executed regardless of the status of the cluster or the EADS servers.
- If the `-sd` or `--safedump` option is specified and the cluster status is available (`AVAILABLE`) or partically available (`PARTIALLY_AVAILABLE`), this subcommand can be executed only if the status of each target EADS server is one of the following:
  - Initialized
  - Running
  - Closed
  - Isolated
- If the `-sd` or `--safedump` option is specified and the cluster status is unavailble (`NOT_AVAILABLE`), this subcommand can be executed if a target EADS server is in the following status:
  - Isolated
- When the `-fd` or `--forcedump` option is specified, the subcommand can be executed on EADS servers that are in any of the following statuses, regardless of the cluster's status:
  - Initializing
  - Initialized
  - Running
  - Closing
  - Closed
  - Isolated
  - Stopping

(3) Format

```
eztool snapshot [-sd|-fd]
```

(4) Options and arguments

(a) -sd or --safedump

Specify this option to collect logs and settings files after executing the `eztool threaddump` command.

An error results in the following cases:

- Acquisition of a lock failed.
- An EADS server is not running.

(b) -fd or --forcedump

Specify this option to collect logs and settings files after forcibly executing the `eztool threaddump` command.

If an EADS server is not running, an error results.

If the `eztool threaddump` command is forcibly executed, EADS servers might become isolated.

(5) Return code

`0`: Normal termination

Other than `0`: Error

(6) Notes

- A `gz` file created by this subcommand is not deleted automatically. There is no limit to the number of files that can be created. If you execute this subcommand periodically, pay attention to the amount of disk space used.
- If this subcommand's processing does not terminate, the file system might be corrupted. In such a case, forcibly terminate the subcommand, and then check the file system for any corruption.
- If the subcommand does not have read permission for the files to be collected, the files cannot be archived.
- While this subcommand is executing, the logs shown below are output to the message log file that is output during command execution. These logs are used internally by EADS to collect information. They are not available for use by the user.
  - When no options are specified
    ```
    snapshot, -v, -t, 60
    snapshot --listconf, -s, -t, 60
    ```
  - When the `-fd` option is specified
    ```
    snapshot, -v, -t, 60
    snapshot, -s, -v, -t, 60
    snapshot --listconf, -s, -t, 60
    ```
  - When the `-sd` option is specified
    ```
    snapshot, -v, -t, 60
    snapshot, -s, -v, -t, 60
    snapshot, --lock, -t, 60#
    snapshot, --unlock, -t, 60#
    snapshot --listconf, -s, -t, 60
    ```
    #: The log is not output for an isolated EADS server.
- When this subcommand is executed, the directories and files listed below are created temporarily. These directories and files are deleted after archive files have been output. If this deletion processing fails, delete them manually as necessary.
  - EADS server thread dumps immediately under *management-directory*`/logs/`

    Once the subcommand has output archive files, it deletes thread dumps that were in existence before the subcommand executed. These thread dumps are deleted even if the `-sd` or `--safedump` option or the `-fd` or `--forcedump` option is not specified. The archive files also contain past thread dumps. If you need a thread dump after you have executed the subcommand, expand the archive files to obtain the thread dump.
  - `snapshot_info` directory, that is created directly under *management-directory*`/logs/` and the files directly under the `snapshot_info` directory

14.3.31 stop (terminates the cluster)

(1) Description

This subcommand terminates all EADS servers in the cluster.

When the option is not specified, this subcommand exports data from memory caches to store data files and then terminates the EADS servers.

(2) Rules

- You can execute this subcommand when the cluster status is AVAILABLE.
- The target of this subcommand is the EADS servers whose cluster participation status is online. This subcommand cannot be executed if the cluster contains any EADS server whose cluster participation status is standby. You can determine the cluster participation status with the eztool status command.
- This subcommand can be executed when the target EADS servers are in the following status:
  - Initialized
  - Closed
- While this subcommand has obtained a lock from the EADS server, the EADS server is not isolated. However, if a process is shut down or the EADS server terminates while the subcommand has the lock, the lock is released, and then the EADS server is isolated.

(3) Format

```
eztool stop [--no_export]
```

(4) Options and arguments

(a) --no_export

Specify this option if you want to not output data from memory caches to store data files when the EADS servers are terminated.

(5) Return code

The following table lists the return codes that this subcommand returns.

Table 14–55: Return codes returned by the eztool stop command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	0	0	Command execution was successful.
2	101	101	Initialization of the command failed.
3	110		Connection establishment failed.

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
4	111		The command failed due to a communication timeout.
5	120		The command failed due to a syntax error.
6	130		The command failed because it could not be executed.
7	131		The command failed because another command was executing.
8	150		The command failed during execution.
9	200		The command failed due to a timeout.

(6) Notes

For details about the store data files that are output when this command is executed, see *7.6.2(3) Store data file output when the eztool stop command is executed*.

For details about the output destinations of the store data files, see *7.4.2 Specifying the file output destinations*.

14.3.32 forcestop (forcibly terminates an EADS server)

(1) Description

This subcommand forcibly terminates an EADS server.

(2) Rules

- This subcommand can be executed regardless of the cluster's status.
- This subcommand can be executed when the EADS server is in the following status:
  - Initializing
  - Initialized
  - Running
  - Closing
  - Closed
  - Isolated
  - Stopping

(3) Format

```
eztool forcestop
```

(4) Return code

`1`: No EADS server was found.

Other than `1`: Termination code of the `kill` command

(5) Notes

This subcommand terminates the EADS server without exporting data from memory caches to store data files. Therefore, if no other EADS server contains the data in the same range, the data in memory caches will be lost.

14.3.33 isolate (isolates an EADS server)

(1) Description

This subcommand isolates an EADS server.

(2) Rules

- You can execute this subcommand when the cluster's status is `AVAILABLE`.
- This subcommand can be executed when the EADS server is in the following status:
  - Initialized
  - Running
  - Closed
  - Isolated
- If this subcommand is executed while the EADS server is running, the subcommand waits until any running user function has terminated, and then isolates the EADS server.

(3) Format

```
eztool isolate [--stop]
```

(4) Options and arguments

(a) --stop

Specify this option to both isolate the EADS server and then terminate it.

If this subcommand is executed on an EADS server that is already isolated, it terminates the EADS server.

(5) Return code

The following table lists the return codes that this subcommand returns.

Table 14–56: Return codes returned by the eztool isolate command

No.	Return code		Description
	eads.command.compat parameter omitted in the command properties	0300 specified in the eads.command.compat parameter in the command properties	
1	`0`	`0`	Command execution was successful.
2	`101`	`101`	Initialization of the command failed.
3	`110`		Connection establishment failed.
4	`111`		The command failed due to a communication timeout.
5	`120`		The command failed due to a syntax error.
6	`130`		The command failed because it could not be executed.
7	`131`		The command failed because another command was executing.
8	`150`		The command failed during execution.
9	`200`		The command failed due to a timeout.

(6) Notes

- After the EADS server has been isolated by this subcommand, an EADS server connection error might occur if an API function is executed from an EADS client whose cluster information has not been updated.
- If the range becomes disabled and data operations can no longer be continued due to execution of this subcommand, an error results.

14.4 Information displayed as execution results by the eztool command's subcommands

You can use the following options to change the information to be displayed by the `eztool` subcommands that display execution results:

- `--format`
- `--columns`
- `--filter`
- `--match`

14.4.1 Components of the displayed information

The information displayed as execution results by `eztool` subcommands consists of the following elements:

- Summary

  This element displays the information that is common to the entirety of the information that is displayed. Its format is a summary name and a summary value.
- Content

  This element displays execution results. Its format is a header and a body.

  A set of data arranged vertically is called a *column*, and a set of data arranged horizontally is called a *row*.
  - Header

    This is a row displaying the types of data (column names) included in the execution results.
  - Body

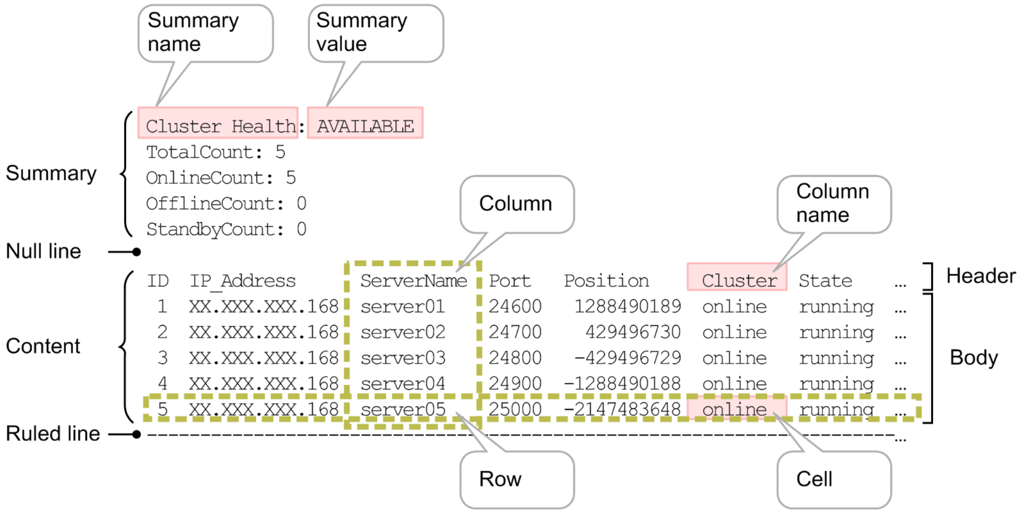
    This is the section that displays execution results.
- Null line

  This is a blank line used to separate data items.
- Ruled line

  This is a line used to separate items for readability.

Each data item in the displayed information is called a cell. If multiple consecutive cells in the same column have the same value, the cell value might be omitted from the second and subsequent rows for readability purposes. In the case of a cell for which there is no data, a hyphen (`-`) or a null character string is output.

The following figure shows the configuration of the elements using the `eztool status -v` command as an example.



```
Cluster Health: AVAILABLE
TotalCount: 5
OnlineCount: 5
OfflineCount: 0
StandbyCount: 0

ID  IP_Address       ServerName  Port   Position     Cluster  State    ...
 1  XX.XXX.XXX.168  server01    24600   1288490189  online   running  ...
 2  XX.XXX.XXX.168  server02    24700    429496730  online   running  ...
 3  XX.XXX.XXX.168  server03    24800   -429496729  online   running  ...
 4  XX.XXX.XXX.168  server04    24900  -1288490188  online   running  ...
 5  XX.XXX.XXX.168  server05    25000  -2147483648  online   running  ...
-----------------------------------------------------------------------...
```

14.4.2 How to specify the display format

You can change the display formats by specifying the `--format` option in the subcommands of the `eztool` command that display execution results.

Whenever the `--format` option is not specified, the default display format (`ALIGN` format) is used. For details about the `ALIGN` format, see *14.4.2(2)(a) format-name*.

(1) Option specification format

```
--format format-name
```

(2) Arguments

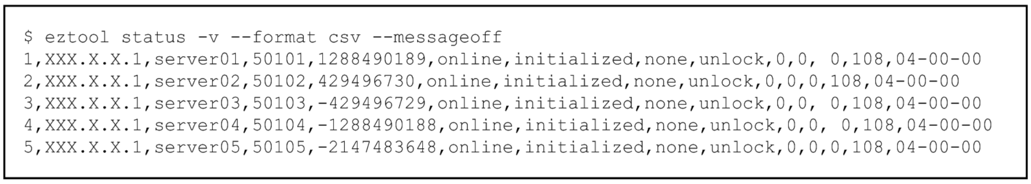
(a) format-name

The supported format names are shown below. They are not case-sensitive.

- `ALIGN`
  This is the default display format. This format displays the summary, content, null lines, and ruled lines. If multiple consecutive cells in the same column have the same value, the cell value might be omitted from the second and subsequent rows. In the case of a cell for which there is no data, a hyphen (-) or a null character string is output.
- `CSV`
  Only the cells in the body are displayed, separated by commas. The summary, null lines, and ruled lines are not displayed. Unlike the `ALIGN` format, cell values are not omitted when multiple consecutive cells in the same column have the same value. In the case of a cell for which there is no data, a null character string is output.
  Consider using this format if you will be automating the use of the `eztool` command's subcommands. We recommend that you specify this argument together with the `--messageoff` option.

(3) Execution example

The following example specifies `CSV` in the `--format` option in the `eztool status -v` command:



```
$ eztool status -v --format csv --messageoff
1,XXX.X.X.1,server01,50101,1288490189,online,initialized,none,unlock,0,0, 0,108,04-00-00
2,XXX.X.X.1,server02,50102,429496730,online,initialized,none,unlock,0,0,0,108,04-00-00
3,XXX.X.X.1,server03,50103,-429496729,online,initialized,none,unlock,0,0, 0,108,04-00-00
4,XXX.X.X.1,server04,50104,-1288490188,online,initialized,none,unlock,0,0, 0,108,04-00-00
5,XXX.X.X.1,server05,50105,-2147483648,online,initialized,none,unlock,0,0,0,108,04-00-00
```

14.4.3 How to specify column filters

You can display only selected columns by specifying the `--columns` option in the subcommands of the `eztool` command that display execution results. You can also rearrange the order in which columns are displayed.

If the `--columns` option is not specified, the items (columns) predefined for each subcommand are displayed in the predefined order.

The command will fail to execute in the following cases:

- The same column name is specified more than once.
- A nonexistent column name is specified.

(1) Option specification format

```
--columns column-name[,column-name]...
```

(2) Arguments

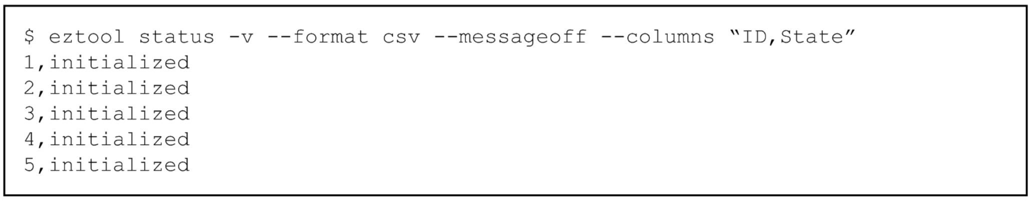
(a) column-name[,column-name]...

This argument specifies the names of the columns to be displayed in the execution results in the order in which they are to be displayed. Column names are case-sensitive.

The `--columns` option is used to filter the columns to be displayed. The names of columns that are not displayed cannot be specified in this option. For example, the names of the columns that are displayed only when the `-v` option is specified cannot be specified if the `-v` option is not specified.

(3) Execution example

The following example specifies the `--columns` option in the `eztool status -v` command:



```
$ eztool status -v --format csv --messageoff --columns “ID,State”
1,initialized
2,initialized
3,initialized
4,initialized
5,initialized
```

14.4.4 How to specify row filters

You can display only the rows that satisfy conditions by specifying the conditions in the `--filter` option.

When conditions are evaluated, values that are not displayed are treated as if they were displayed. Cells with no data are treated as null character strings.

If a specified condition is invalid, command execution fails.

(1) Option specification format (BNF notation)

```
--filter filter-condition
filter-condition ::= row-condition

row-condition ::= column-condition|column-condition logical-operator column-
condition
column-condition ::= column-name comparison-operator-(character-string)
character-string
               |column-name comparison-operator-(numeric-value) numeric-
value

logical-operator ::= "&&"|"||"
comparison-operator-(character-string) ::= "=="|"!="
comparison-operator-(numeric-value) ::= ">"|"<"|">="|"<="

character-string ::= String-character-string
numeric-value ::= numeric-value-that-can-be-converted-by-Long.parseLong()
```

Important note

Enclose the entire condition in double quotation marks (`"`) in the following cases

- *column-name* or *character-string* contains a space.
- *comparison-operator-(numeric-value)* is specified.

If such a condition is not enclosed in double quotation marks (`"`), it might be treated as separate arguments or redirected incorrectly.

Reference note

How to interpret the BNF notation

The item on the left of `::=` is to be specified in the format shown by the items on the right. The following example explains the specification format of *filter-condition*:

1. *filter-condition*`::=`*row-condition* means that *filter-condition* is to be specified in the format *row-condition*.
2. *row-condition*`::=`*column-condition* | *column-condition logical-operator column-condition* means that *row-condition* is to be specified in the format *column-condition* or the format *column-condition logical-operator column-condition*.
3. From 1 and 2, *filter-condition* must be specified in the format *column-condition* or the format *column-condition logical-operator column-condition*.

(2) Arguments

(a) filter-condition

This argument specifies *row-condition*.

(b) row-condition

This argument specifies *column-condition* or *column-condition logical-operator column-condition*.

(c) column-condition

This argument specifies *column-name comparison-operator-(character-string) character-string* or *column-name comparison-operator-(numeric-value) numeric-value*.

(d) column-name

This argument specifies column names for the content to be displayed.

Column names can be specified regardless of whether they are specified in the `--columns` option. However, the names of columns that are not displayed cannot be specified regardless of the `--columns` option. For example, the names of columns that are displayed only when the `-v` option is specified cannot be specified if the `-v` option is not specified.

(e) logical-operator

You can specify the logical operators shown in the table below.

If multiple logical operators are specified, they are evaluated sequentially from left to right.

Table 14–57: Logical operators that can be specified in the --filter option

No.	Logical operator	Example	Description
1	`&&`	`A && B`	`A` and `B` (If `A` and `B` are both true, the condition is true; otherwise, the condition is false)
2	`||`	`A || B`	`A` or `B` (If `A` and `B` are both false, the condition is false; otherwise, the condition is true)

(f) comparison-operator-(character-string)

You can specify the comparison operators shown in the following table.

Table 14–58: Comparison operators that can be specified in the --filter option (character string)

No.	Comparison operator	Example	Description
1	`==`	`A == B`	Whether the pattern of character string `A` matches the pattern of character string `B` (case-sensitive)
2	`!=`	`A != B`	Whether the pattern of character string `A` does not match the pattern of character string `B` (case-sensitive)

(g) comparison-operator-(numeric-value)

You can specify the comparison operators shown in the following table.

Table 14–59: Comparison operators that can be specified in the --filter option (numeric value)

No.	Comparison operator	Example	Description
1	>	`A > B`	Whether `A` as converted to a `long` value is greater than `B` as converted to a `long` value (if there is no data for `A`, the condition is false)
2	<	`A < B`	Whether `A` as converted to a `long` value is less than `B` as converted to a `long` value (if there is no data for `A`, the condition is false)
3	>=	`A >= B`	Whether `A` as converted to a `long` value is equal to or greater than `B` as converted to a `long` value (if there is no data for `A`, the condition is false)
4	<=	`A <= B`	Whether `A` as converted to a `long` value is equal to or less than `B` as converted to a `long` value (if there is no data for `A`, the condition is false)

(h) character-string

This argument specifies a `String` character string. You can use the two wildcards `*` and `?`.

(i) numeric-value

This argument specifies a numeric value that can be converted by `Long.parseLong()`.

(3) Execution example

This example specifies the `--filter` option in the `eztool status -v` command.

- Displaying the EADS server IDs of the EADS servers that have 3,000 or more keys and the total number of keys

```
$ eztool status -v --format csv --messageoff --filter "KeyCount>=3000" --columns "ID,KeyCount"
5,3034
```

- Displaying only parameters related to logs (output example is omitted)

```
$ eztool listconf --messageoff --filter "Parameter==eads.logger.*"
```

14.4.5 How to specify a condition match

If you specify conditions in the `--match` option, you can return as the return code a value that indicates whether the execution results satisfy the condition. If the specified condition is satisfied, `0` is returned; if not, `1` is returned. For details, see the description of each subcommand's return codes. The information displayed as the results of subcommand processing is the same as when this option is omitted.

If the `--match` option is specified together with the `--filter` option, a value is returned as the return code that indicates whether the results obtained after filtering satisfy the condition. If the results are not displayed due to a warning, for example, `1` is returned.

When conditions are evaluated, values that are not displayed in cells are treated as if they were displayed. Cells with no data are treated as null character strings.

If the specified conditions are invalid, the command's execution fails.

(1) Option specification format (BNF notation)

```
--match matching-condition
matching-condition ::= Boolean-function-condition|Value-function-condition|
summary-condition

Boolean-function-condition ::= Boolean-function-name"("row-condition")"
                      Boolean-function-name ::= ALL|EXIST
Value-function-condition ::= Value-function-name"("row-
condition")"comparison-operator-(character-string) character-string
                   |Value-function-name"("row-condition")"comparison-
operator-(numeric-value) numeric-value
                   Value-function-name ::= COUNT
summary-condition ::= summary-name comparison-operator-(character-string)
character-string
                   |summary-name comparison-operator-(numeric-value)
numeric-value

row-condition ::= column-condition|column-condition logical-operator column-
condition
column-condition ::= column-name comparison-operator-(character-string)
character-string
               |column-name comparison-operator-(numeric-value) numeric-
value

comparison-operator-(character-string) ::= "=="|"!="
comparison-operator-(numeric-value) ::= ">"|"<"|">="|"<="
logical-operator ::= "&&"|"||"

character-string ::= String-character-string
numeric-value ::= numeric-value-that-can-be-converted-by-Long.parseLong()
```

Important note

Enclose the entire condition in double quotation marks (") in the following cases:

- *summary-name*, *column-name*, or *character-string* contains a space.
- *comparison-operator-(numeric-value)* is specified.

If such a condition is not enclosed in double quotation marks ("), it might be treated as separate arguments or redirected incorrectly.

Reference note

For details about how to interpret the BNF notation, see *14.4.4(1) Option specification format (BNF notation)*.

(2) Arguments

(a) matching-condition

This argument specifies *Boolean-function-condition*, *Value-function-condition*, or *summary-condition*.

(b) Boolean-function-condition

This argument specifies *Boolean-function-name*`"("`*row-condition*`")"`.

(c) Boolean-function-name

You can specify the functions shown in the following table.

Table 14–60: Boolean functions that can be specified in the --match option

No.	Function	Description
1	`ALL`(*row-condition*)	Whether all rows satisfy the row condition
2	`EXIST`(*row-condition*)	Whether at least one row satisfies the row condition

(d) Value-function-condition

This argument specifies *Value-function-name*`"("`*row-condition*`")"`*comparison-operator-(character-string) character-string* or *Value-function-name*`"("`*row-condition*`")"`*comparison-operator-(numeric-value) numeric-value*.

(e) Value-function-name

You can specify the `Value` function shown in the following table.

Table 14–61: Value function that can be specified in the --match option

No.	Function	Description
1	`COUNT`(*row-condition*)	Returns the number of rows that satisfy the row condition

(f) summary-condition

This argument specifies *summary-name comparison-operator-(character-string) character-string* or *summary-name comparison-operator-(numeric-value) numeric-value*.

(g) summary-name

This argument specifies a summary name for the information that is displayed.

A summary name can be specified regardless of whether the `--format` and `--column` options are specified.

Summary names cannot be specified if their display depends on whether an option other than `--format` or `--column` is specified. For example, the summary names that are displayed only when the `-v` option is specified cannot be specified if the `-v` option is not specified.

(h) row-condition

This argument specifies *column-condition* or *column-condition logical-operator column-condition*.

(i) column-condition

This argument specifies *column-name comparison-operator-(character-string) character-string* or *column-name comparison-operator-(numeric-value) numeric-value*.

(j) column-name

This argument specifies column names for the content to be displayed.

A column name can be specified regardless of whether the `--format` and `--column` options are specified. However, column names cannot be specified if their display depends on whether an option other than `--format` or `--column` is specified. For example, the column names that are displayed only when the `-v` option is specified cannot be specified if the `-v` option is not specified.

(k) comparison-operator-(character-string)

You can specify the comparison operators shown in the following table.

Table 14–62: Comparison operators that can be specified in the --match option (character string)

No.	Comparison operator	Example	Description
1	`==`	`A == B`	Whether the pattern of character string `A` matches the pattern of character string `B` (case-sensitive)
2	`!=`	`A != B`	Whether the pattern of character string `A` does not match the pattern of character string `B` (case-sensitive)

(l) comparison-operator-(numeric-value)

You can specify the comparison operators shown in the following table.

Table 14–63: Comparison operators that can be specified in the --match option (numeric value)

No.	Comparison operator	Example	Description
1	`>`	`A > B`	Whether `A` as converted to a `long` value is greater than `B` as converted to a `long` value (if there is no data for `A`, the condition is false)
2	`<`	`A < B`	Whether `A` as converted to a `long` value is less than `B` that as converted to a `long` value (if there is no data for `A`, the condition is false)
3	`>=`	`A >= B`	Whether `A` as converted to a `long` value is equal to or greater than `B` as converted to a `long` value (if there is no data for `A`, the condition is false)
4	`<=`	`A <= B`	Whether `A` as converted to a `long` value is equal to or less than `B` as converted to a `long` value (if there is no data for `A`, the condition is false)

(m) logical-operator

You can specify the logical operators shown in the table below.

If multiple logical operators are specified, they are evaluated sequentially from left to right.

Table 14–64: Logical operators that can be specified in the --match option

No.	Logical operator	Example	Description
1	&&	A && B	A and B (if A and B are both true, the condition is true; otherwise, the condition is false)
2	||	A || B	A or B (if A and B are both false, the condition is false; otherwise, the condition is true)

(n) character-string

This argument specifies a `String` character string. You can specify the two wildcards `*` and `?`.

(o) numeric-value

This argument specifies a numeric value that can be converted by `Long.parseLong()`.

(3) Execution example

This example specifies the `--match` option in the `eztool status -v` command (output example is omitted).

- Checking whether the cluster has started

```
$ eztool status -v --match "ALL(State==initialized)"
```

- Checking for an EADS server whose explicit heap usage rate is 70% or higher

```
$ eztool status -v --match "EXIST(UsedMemoryRatio>=70)"
```

- Checking for a cache with a specified cache name

```
$ eztool listcache --match "EXIST(CacheName==cache)"
```

- Checking whether an EADS server has been restored

```
$ eztool status -s -v --match "ALL(State==initialized || State==running || State==closed)"
```

- Monitoring the number of cache files

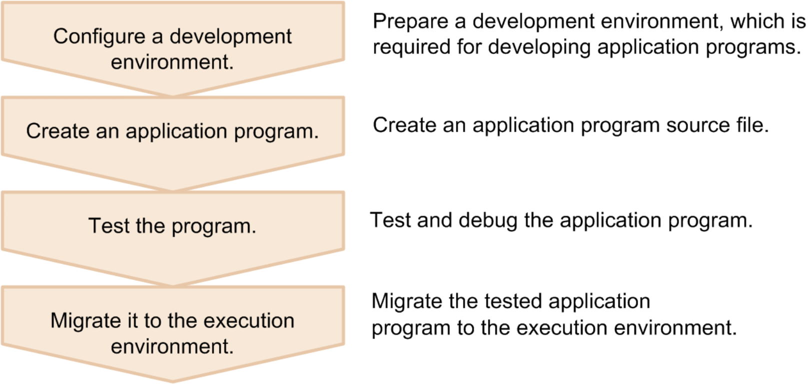
```
$ eztool listecf -s --match "EXIST(UnusedFC<=3)"
```

15 General Procedure for Developing Application Programs

This chapter explains the general procedure and prerequisites for developing application programs.

15.1 General procedure for developing application programs

The following figure shows the general procedure for developing application programs.


Configure a development environment.
Prepare a development environment, which is required for developing application programs.
Create an application program.
Create an application program source file.
Test the program.
Test and debug the application program.
Migrate it to the execution environment.
Migrate the tested application program to the execution environment.

15.1.1 Configure a development environment

Prepare a development environment in which to create application programs.

The program products required for the application program development environment depend on the language being used to create application programs (Java or C). Install the required program products, and then set up the application program development environment.

- Java
    - Hitachi Elastic Application Data Store
- C
    - Hitachi Elastic Application Data Store
    - Hitachi Elastic Application Data Store Client for C

Referring to the following chapters, configure the EADS servers and EADS clients in the same manner as you would for the execution environment, and then use them for the development environment:

- *5. Installing and Setting Up (EADS Servers)*
- *6. Installing and Setting Up (EADS Clients)*

15.1.2 Create application programs

After you have configured the development environment, create application programs.

If you are using Java to create client application programs, see *16. Creating Client Application Programs (in Java)*.

If you are creating user functions, see *17. Creating User Functions*.

If you are using C to create client application programs, see *19. Creating a Client Application Program (in C)*.

15.1.3 Test the application programs

After you have finished creating each application program, test and debug it.

15.1.4 Migrate the created application programs to the execution environment

After you have finished testing each application program, migrate it to the execution environment.

15.2 Prerequisites for the development of application programs

This section explains prerequisites for developing application programs in EADS, including the programming languages that can be used to create application programs, the EADS client program products to use, and the supported keys, group names, values, and cache names.

15.2.1 Programming languages for application programs and EADS clients

This subsection explains the programming languages that EADS supports for application programs, and the types of EADS clients.

(1) Programming languages used to create application programs

You can use the following languages to create application programs for EADS:

- Java
- C

(2) Types of EADS clients

There are two types of EADS client program products (client libraries) available, depending on which programming language you are using to create application programs:

- Java
  Hitachi Elastic Application Data Store Client for Java
- C
  Hitachi Elastic Application Data Store Client for C

To use one of the above program products in an execution environment, install the product in each execution environment.

Important note

- If you use an application program created with a client library whose version is more recent than the version of the EADS client, valid operation is not guaranteed. In such a case, you need to edit and recompile the source programs.

  An application program created with a client library whose version is 03-00 or later can be used with more recent versions of EADS clients.
- EADS clients and EADS servers whose version is earlier than 03-00 are not compatible with EADS clients and EADS servers whose version is 03-00 or later. For example, EADS clients and EADS servers whose version is 03-50 are not compatible with EADS clients and EADS servers whose version is 02-00 or earlier. If an attempt is made to establish a connection between incompatible programs, valid operation is not guaranteed.
- If you use an EADS client within a user function, the EADS client's version needs to be the same as the EADS server's version. If you use an EADS client whose version differs from the EADS server's version, valid operation is not guaranteed.

15.2.2 Data that is supported as keys, group names, values, cache names, and EADS client names

This subsection explains the data that can be specified for keys, group names, values, cache names, and EADS client names.

(1) Data types that can be specified as keys

This subsection explains the format of keys and the data that can be specified.

(a) Format of keys

```
key=
  [group-name:]element-name
```

When a key is not grouped, *element-name* is the key. When a key is grouped, a set of *group-names*, delimiters (`:`), and an *element-name* constitutes the key. Therefore, keys with the same *element-name* but a difference in the *group-names* are different keys.

For the format of *group-name*, see *15.2.2(2) Data that can be specified as group names*.

The following shows examples of key specifications:

- Key that is not grouped

```
key1
```

- Key that is grouped

```
group1:key1
```

- Key whose groups are arranged hierarchically

```
group1:group2:key1
```

(b) Data types and number of characters that can be specified as keys

The table below lists and describes the data types and the number of characters that can be specified as keys in each programming language used to create application programs. When a key is grouped, the maximum length is 1,024 characters including the group names, delimiters (`:`), and element name.

Table 15–1: Data types that can be specified as keys

Programming language for application programs	Data type supported as keys	Number of characters	Remarks
Java	Character string (`java.lang.String`)	1 to 1,024	Specifying `null` or the null character string is invalid.
C	Character string ending with the terminal symbol `\0` (`char *`)	1 to 1,024	• The maximum size of a character string is in bytes with `\0` excluded. • A character string with a length of zero (`\0` only) is not permitted.

Important note

The `eztool put`, `eztool get`, and `eztool remove` commands can be used to test the configured execution environment to determine whether it is running normally. Therefore, the data types and sizes permitted in the commands are not completely compatible with those permitted in API functions (`put`, `get`, `remove`).

(c) Key specification rules

This subsection explains the rules for specifying keys.

- The ASCII codes `0x20` through `0x7E` are supported for specifying keys.

  Note that the characters listed in the following table cannot be used in keys, except as noted, because they are EADS reserved characters.

Table 15–2: Characters that cannot be used in keys

ASCII code	Character	Exception
`0x22`	Double quotation mark (")	None
`0x28`	Left parenthesis (( )	None
`0x29`	Right parenthesis ( ))	None
`0x3a`	Colon (:)	This character can be specified only as a delimiter between group hierarchy names and an element name. This character cannot be used within a group hierarchy name or an element name.
`0x3c`	Less-than sign (<)	None
`0x3e`	Greater-than sign (>)	None
`0x5b`	Left square bracket ([ )	These characters can be used only when EADS server IDs of storage EADS servers are specified in EADS server ID specified groups.
`0x5d`	Right square bracket (])	
`0x7c`	Vertical bar (|)	None

(2) Data that can be specified as group names

This subsection explains the format of group names and the data that can be specified.

(a) Format of group name

```
group-name=
 group-hierarchy-name[:group-hierarchy-name]...
```

You can specify as a group name a set consisting of a group hierarchy name in a desired hierarchy through the group hierarchy name in the highest hierarchy. A name beginning with an intermediate group hierarchy name cannot be specified.

When group names are specified in an API function, a name beginning with an intermediate group hierarchy name cannot be specified. For example, if a key is `groupA:groupB:groupC:key`, a group name beginning with `groupB:groupC` cannot be specified.

The following examples show the relationship between group names.

- Group names that can be specified when the key is `groupA:groupB:groupC:key`

  ```
  groupA
  ```

  ```
  groupA:groupB
  ```

  ```
  groupA:groupB:groupC
  ```

- Group names that can be specified when the key is `[10]group1:group2:key`

  ```
  [10]group1
  ```

  ```
  [10]group1:group2
  ```

(b) EADS server ID specified groups

If you group keys by specifying a key storage EADS server, specify the EADS server ID of the EADS server where the keys are stored. This type of group is called an EADS server ID specified group.

The following shows the format of keys that contain an EADS server ID specified group:

```
[EADS-server-ID]group-hierarchy-name:[group-hierarchy-name:]...element-name
```

When you use an EADS server ID specified group to group a key, specify the EADS server ID enclosed in square brackets (`[]`) as the key's first group hierarchy name (the left square bracket (`[`) and the right square bracket (`]`) in `[`*EADS-server-ID*`]` must be specified, as shown in the format; these square brackets do not mean that the enclosed item can be omitted). Specify for the EADS server ID an integer that does not begin with zero.

A set of a left square bracket (`[`), an EADS server ID, and a right square bracket (`]`) is treated as a group name.

The following example shows a key that contains an EADS server ID specified group:

```
[1]group1:key1
```

(c) Data types and the number of characters that can be specified as group names

For details about the data types supported for group names, see *15.2.2(1)(b) Data types and number of characters that can be specified as keys*. When keys are grouped, the maximum length is 1,024 characters including the group names, delimiters (`:`), and element name. Therefore, a maximum of 1,022 characters are permitted for the group names.

(d) Group name specification rules

This subsection explains the rules for specifying group names. For other rules and notes, see *15.2.2(1)(c) Key specification rules*.

- When keys are grouped, there is no upper limit to the number of groups and group hierarchies that can be created.

  Note that when groups are arranged in a hierarchy, the processing time increases as the hierarchy goes to lower levels. It is advisable to configure hierarchies appropriately.

- When an EADS server ID specified group is used, the EADS server ID of an EADS server that does not exist in the cluster cannot be specified.

(3) Data types that can be specified as values

The following table lists and describes the data types and sizes that can be specified as values in each programming language used to create application programs.

Table 15–3: Data types that can be specified as values

Programming language for application programs	Data type supported as values	Size (bytes)	Remarks
Java	Any object that can be serialized (`java.lang.Object`)	1 to 262,144	• This is the length of a serialized byte array. • Specifying `null` is invalid.
C	Any byte array (`void *`)	1 to 262,144	This is the length of any byte array.

Important note

- The permitted data types and sizes are not completely compatible between the API functions (`put`, `get`, `remove`) and the commands (`eztool put`, `eztool get`, `eztool remove`).
- If your Java client application program or user function and C client application program handle the same keys, use byte arrays for values. The values stored as byte arrays by the Java client application program or user function can be acquired by the C client application program. If a C client application program acquires values that are not byte arrays, an error results.

  The values stored by the C client application program can be acquired as byte arrays by the Java client application program or a user function.

(4) Data types that can be specified as cache names

The following table lists and describes the data types and the number of characters that can be specified as cache names in each programming language used to create application programs.

Table 15–4: Data types that can be specified as cache names

Programming language for application programs	Data type supported as cache names	Number of characters	Remarks
Java	Character string (`java.lang.String`)	1 to 32	• The permitted characters are `0x20` through `0x7E` in ASCII codes. • Specifying `null` or the null character string is invalid.
C	Character string ending with the terminal symbol `\0` (`char *`)	1 to 32	• The permitted characters are `0x20` through `0x7E` in ASCII codes. • The maximum size of a character string is in bytes with `\0` excluded. • A character string with a length of zero (`\0` only) is not permitted.

(5) Data that can be specified as EADS client names

The following table lists and describes the data types and the number of characters that can be specified as EADS client names in each programming language that can be used to create application programs.

Table 15–5: Data that can be specified as EADS client names

Programming language for application programs	Data types supported as EADS client names	Number of characters	Remarks
Java	Character string (`java.lang.String`)	0 to 16	• Alphanumeric characters (`0` to `9`, `A` to `Z`, `a` to `z`) and underscores (_) are permitted. • A null character string is permitted. • Specifying `null` is invalid.
C	Character string ending with the terminal symbol `\0` (`char *`)	0 to 16	• Alphanumeric characters (`0` to `9`, `A` to `Z`, `a` to `z`) and underscores (_) are permitted. • The maximum size of a character string is in bytes with `\0` excluded. • A character string with a length of zero (`\0` only) is permitted.

15.2.3 Reserved package and system property names (applicable to Java)

In application programs, do not use the following reserved package and system property names:

- Reserved package names
  Package names beginning with `com.hitachi.software.xeads.`
- Reserved system property names
  Property names beginning with `eads.`

16 Creating Client Application Programs (in Java)

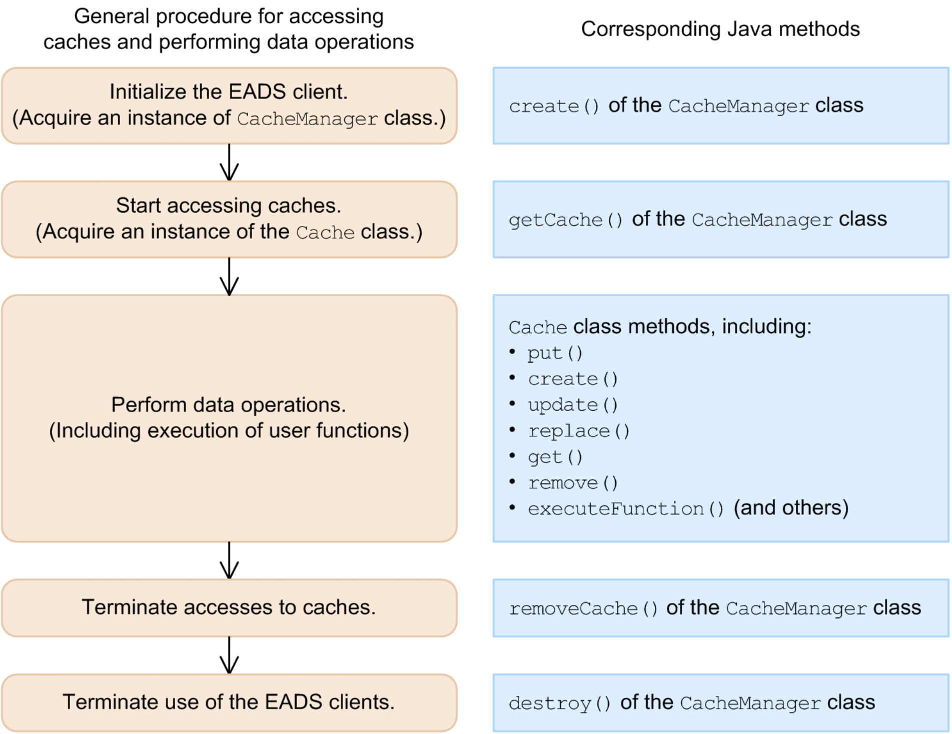
This chapter explains how to create client application programs using Java.

16.1 Creating source programs (in Java)

This section explains the general procedure for accessing caches and manipulating data, and shows an example of creating a source program.

16.1.1 General procedure for accessing caches and manipulating data

The following figure shows the general procedure for accessing caches and manipulating data:


General procedure for accessing caches and performing data operations
Corresponding Java methods
Initialize the EADS client. (Acquire an instance of CacheManager class.)
create() of the CacheManager class
Start accessing caches. (Acquire an instance of the Cache class.)
getCache() of the CacheManager class
Perform data operations. (Including execution of user functions)
Cache class methods, including:
• put()
• create()
• update()
• replace()
• get()
• remove()
• executeFunction() (and others)
Terminate accesses to caches.
removeCache() of the CacheManager class
Terminate use of the EADS clients.
destroy() of the CacheManager class

(1) Example of a source program in Java

The following shows an example of a source program in Java (one that stores keys and values):

```
// Import the package provided by EADS
import com.hitachi.software.xeads.client.api.*;

public class PutSample {

public static void main(String[] args) {
    // Initialize the EADS client

    final String CONFPATH = "./conf/eads_sample_client.properties";
    final String CACHENAME = "cache1";
    CacheManager cacheManager = null;
    Cache cache = null;

    try {
        cacheManager = CacheManager.create(CONFPATH);
        // Start accessing caches
        cache = cacheManager.getCache(CACHENAME);
        System.out.println("cache start succeeded. (cache name = " +
CACHENAME + ")");
        // Store keys and values
        final String KEY = "key1";
        final String VALUE = "value1";

        cache.put(KEY, VALUE);
        System.out.println("PUT succeeded. (key = " + KEY + ", value = " +
VALUE + ")");

        String value = (String)cache.get(KEY);
        System.out.println("GET succeeded. (key = " + KEY + ", value = " +
value + ")");

    } catch (CacheException e) {
        System.out.println("cache operation failed.(cache name = " +
CACHENAME + ", error code = " + e.getErrorCode() + ")");
    }finally{
        if(cacheManager != null){
            // Terminate accesses to caches
            if(cache != null){
                try{
                    cacheManager.removeCache(CACHENAME);
                    System.out.println("cache stop succeeded. (cache name =
" + CACHENAME + ")");
                }catch(CacheException e){
                    System.out.println("CacheManager.removeCache() failed.
(error code = " + e.getErrorCode() + ")");
                }
            }
            // Terminate the use of EADS client
            try{
                cacheManager.destroy();
            }catch(CacheException e){
                System.out.println("CacheManager.destroy() failed. (error
code = " + e.getErrorCode() + ")");
            }
        }
    }
  }
}
```

(2) Importing the package provided by EADS

Import the following package provided by EADS:

```
import com.hitachi.software.xeads.client.api.*;
```

(3) Initializing the EADS client

To initialize the EADS client, use `create()` of the `CacheManager` class to create an instance of the `CacheManager` class.

The settings, including the EADS servers to be connected, are specified according to the client properties.

If you want to use multiple instances of a CacheManager class whose settings are different, such as when a connection is established with multiple clusters, edit the client properties and then execute create() of the CacheManager class multiple times. To terminate use of the EADS client when you have executed create() of the CacheManager class multiple times, execute destroy() of the CacheManager class on each of the acquired instances of the CacheManager class.

(4) Starting to access caches

After you have finished initializing the EADS client, start accessing caches.

To start accessing caches, use getCache() of the CacheManager class to create an instance for manipulating data (an instance of the Cache class).

(5) Storing keys and values

To store keys and values in a cache, use put() of the Cache class.

In put(), specify the keys and values to be stored in the cache.

(6) Acquiring values

To acquire values from a cache, use get() of the Cache class.

In get(), specify the key associated with the value you want to acquire.

If the value is acquired successfully by get(), the value associated with the key is returned as a return value.

The following shows an example of a source program for acquiring values.

Example of a source program (for acquiring values)

```
    // Acquire value
    final String KEY = "key1";
    try {
        String value = (String) cache.get(KEY);
        System.out.println("GET succeeded. (key = " + KEY + ", value = " +
value + ")");
    } catch (CacheException e) {
        int errcode = e.getErrorCode();
        System.out.println("GET failed. (key = " + KEY + ", error code = "
+ errcode + ")");
    }
```

(7) Deleting keys and values

To delete a specified key and the value associated with that key from a cache, use remove() of the Cache class.

In remove(), specify the key associated with the value you want to delete.

(8) Executing user functions

To execute user functions, use executeFunction() of the Cache class.

Specify in executeFunction() the name of the key or the group for executing the user function or an instance of the Node class, the name of the user function to be executed, and arguments to be passed to the user function.

If the user function is executed by executeFunction(), the user function execution results are returned.

(9) Terminating accesses to caches

To terminate accesses to caches, use removeCache() of the CacheManager class.

In removeCache(), specify the name of the cache to which you want to terminate access.

(10) Terminating use of the EADS client

To terminate use of the EADS client, use destroy() of the CacheManager class.

16.2 Notes about creating client application programs (in Java)

This section provides notes about creating client application programs.

16.2.1 Notes about initializing an EADS client

The following notes apply to initializing an EADS client:

- If you execute multiple EADS clients concurrently on the same machine, specify a different log output destination for each EADS client. If you specify the same log output destination, valid operation is not guaranteed. For details about specifying the log output destination, see *8.4.2 Specifying the file output destinations*.
- If an EADS client is initialized with the EADS client name omitted, the result is the same as when initialization is performed with a null character string specified as the EADS client name. For details about the relationship between the EADS client name and the log output destination, see *8.4.2 Specifying the file output destinations*.
- If you run client application programs on a Java EE server (uCosminexus Application Server), initialize the EADS client when you start the application programs by using one of the following methods:
    - `ServletContextListener`
    - Servlet's `init` method
- If you run client application programs on a Java EE server (uCosminexus Application Server), make sure that you execute `create()` of the `CacheManager` class, and then execute `destroy()` of the `CacheManager` class. If you fail to do this, a memory leak will occur.
- Use the following procedure to change client properties:
    1. Use `destroy()` of the `CacheManager` class to terminate use of the EADS client.
    2. Update the client property file.
    3. Use `create()` of the `CacheManager` class to initialize the EADS client again.

16.2.2 Notes about starting access to caches

For cache names, specify the name that was created beforehand by using the `eztool createcache` command. If the specified cache name does not exist, `CacheException.EAD_ERROR_NET_CLUSTERINFO` is returned.

16.2.3 Notes about manipulating data

(1) Notes about manipulating data

The following notes apply to manipulating data:

- If a specified key is already in cache, `put()` unconditionally updates the value. If you do not want to update values unconditionally, use the following methods:
    - `create()`

      Only when a new key is stored, this method associates a value with the key and stores the value.
    - `update()`

Only when a specified key is already stored, this method associates a value with the key and stores the value.

  - `replace()`

    Compares the value associated with a specified key with the value (`comparativeValue`) specified as the condition. Then, only if their values match, this method associates the value with the key and stores the value.

- If a key specified during the execution of `get()` does not exist in the cache, `null` is returned.
- If a key specified during the execution of `replace()` does not exist, `CacheException.EAD_ERROR_SERVER_REPLACE_METHOD_KEY_NOT_EXIST` is returned.
- If comparison results do not match during the execution of `replace()`, `CacheException.EAD_ERROR_SERVER_REPLACE_METHOD_NOT_MATCHED` is returned.
- If a key specified during the execution of `create()` already exists, `CacheException.EAD_ERROR_SERVER_CREATE_METHOD_KEY_EXIST` is returned.
- If a key specified during the execution of `update()` does not exist, `CacheException.EAD_ERROR_SERVER_UPDATE_METHOD_KEY_NOT_EXIST` is returned.

(2) Notes about batch data operations

The following notes apply to batch data operations.

- If a specified key has already been stored in the cache, `putAll()` updates the value unconditionally.
- Because caches are not locked while operations are ongoing, a target value might be changed by another cache operation during a batch operation.
- When batch operations are performed on a large amount of data, a large amount of memory might be required by EADS clients and EADS servers.
- When batch operations are performed on a large amount of data, it might take a long time to complete the processing. To ensure proper operation, make sure that you design a timeout value that is appropriate for the processing time.
- If manipulation of a key fails or the cluster configuration is changed while batch operation with multiple keys specified is underway, the batch operation will be terminated and any operations that have not yet been performed will be cancelled. Similarly, when the cluster configuration is changed by a user operation, such as scale-out processing (adding EADS servers) or restoration processing, during batch operation, any operations that have not yet been performed will be cancelled.

  Identify the key resulting in an error and determine the cause of the error from each API method's return value. You can identify a key whose manipulation was cancelled based on the error code of `CacheException.EAD_ERROR_CLIENT_BATCH_CANCEL`.
- When all key manipulations fail when batch operation with multiple keys specified is attempted, `CacheException.EAD_ERROR_BATCH_FAILED_ALL` is returned.
- When only some key manipulations fail when batch operation with multiple keys specified is performed, `CacheException.EAD_ERROR_BATCH_FAILED_PART` is returned.

16.2.4 Notes about terminating access to caches

The following notes apply to terminating access to caches:

- To access a cache whose access has been terminated by `removeCache()` of the `CacheManager` class, issue `getCache()` of the `CacheManager` class.

- removeCache() of the CacheManager class terminates access to all caches obtained with the same cache name specified (instances of the Cache class). Therefore, use removeCache() of the CacheManager class carefully when it is executed in multiple threads.

16.2.5 Notes about terminating use of the EADS client

The following notes apply to terminating use of the EADS client:

- Execute destroy() of the CacheManager class paired with create() of the CacheManager class. Execute destroy() only once for each instance acquired by create() of the CacheManager class.
- If you run client application programs on a Java EE server (uCosminexus Application Server), make sure that you execute destroy() of the CacheManager class. If you fail to do this, a memory leak will occur.
- If the EADS client had already been terminated when destroy() of the CacheManager class was executed, CacheException is returned.

16.3 Compiling source programs (in Java)

To compile a created source program, use JDK's `javac` command.

For details about the `javac` command, see the Java compiler-related documentation.

The following class libraries are required for compiling source programs:

- `/opt/hitachi/xeads/javaclient/lib/eads-client.jar`
- `/opt/hitachi/xeads/javaclient/lib/eads-common.jar`
- `/opt/hitachi/xeads/javaclient/lib/hntrlib2-eads-j.jar`

If you plan to run your application programs on a Java EE server (uCosminexus Application Server), you must include the following libraries in the application programs (`WAR` file (`WEB-INF/lib` directory)):#

- `/opt/hitachi/xeads/javaclient/lib/eads-client.jar`
- `/opt/hitachi/xeads/javaclient/lib/eads-common.jar`
- `/opt/hitachi/xeads/javaclient/lib/hntrlib2-eads-j.jar`

#

Can be executed from a Servlet or JSP (includes Filter and Listener).

17 Creating User Functions

This chapter explains how to create and then run user functions.

17.1 Prerequisites for creating user functions

This section explains prerequisites for creating user functions.

For an overview of user functions, see *2.7 Efficient data processing using user functions*.

17.1.1 Programming language for user functions

You can use Java to create user functions.

User functions must be implemented with the EADS-provided `Function` interface and archived as `jar` files.

> **Important note**
>
> If you attempt to use a user function that has been implemented with a `Function` interface whose version is more recent than the EADS server's version, valid operation is not guaranteed. In such a case, you need to edit and recompile the source programs.
>
> A user function created with a `Function` interface whose version is 03-00 or later can be used with more recent versions of EADS servers.

You can place multiple `jar` files on a single EADS server. Therefore, if you want to execute multiple user functions on the same EADS server, you can use different `jar` files according to purpose.

17.1.2 User function execution methods

There are two ways to execute user functions:

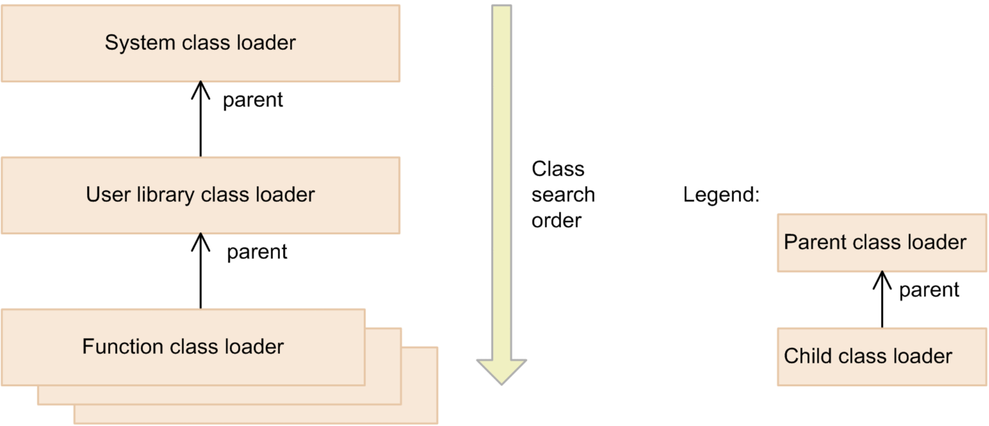
- By specifying a key or a group
- By specifying an EADS server

For details about the mechanism of user functions, see *2.7.1 Mechanism of user functions*.

17.1.3 Java class loaders used by EADS servers

This subsection explains the Java class loaders used by EADS servers.

The role of a Java class loader is to dynamically load Java classes to a Java Virtual Machine. The following figure shows the configuration of the Java class loaders that are used by EADS servers.



- System class loader

  Loads the classes that are used by `jdk` libraries and EADS servers.
- User library class loader

  Loads the library classes that are used by the user functions placed under *management-directory*`/app/lib`.

  One user library class loader is created for each EADS server.
- Function class loader

  Loads the classes contained in user functions' `jar` files and the classes contained in the libraries specified in the `jar` files' manifest files (classes used only by user functions).

  One function class loader is created for each `jar` file. Therefore, if multiple `jar` files are placed for user functions, multiple function class loaders are created.

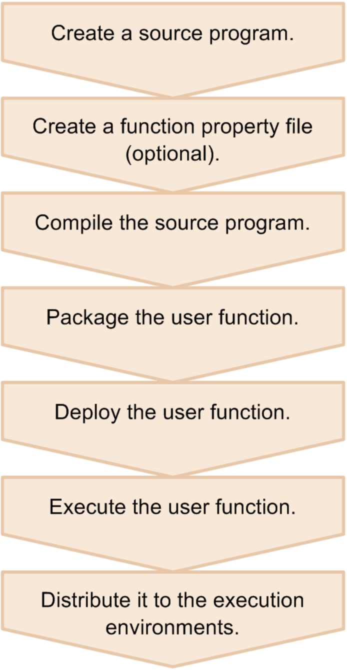
Class loader search processing begins with the system class loaders that are parent class loaders.

User functions are loaded in ASCII code order of the `jar` file names.

You can load classes with the same fully qualified class name to different function class loaders, but only the first user function loaded can be executed. For a user function loaded later, no instance is created and the `Function` interface's `init()` is not executed.

17.1.4 General procedure for creating user functions

The following figure shows the general procedure for creating user functions.


Create a source program.
Create a function property file (optional).
Compile the source program.
Package the user function.
Deploy the user function.
Execute the user function.
Distribute it to the execution environments.

Create a user function with the `Function` interface implemented.

Create a function property file, if necessary.

Compile it to a `class` file.

Create a `jar` file.

Place the `jar` file.

Execute the user function to check its operation.

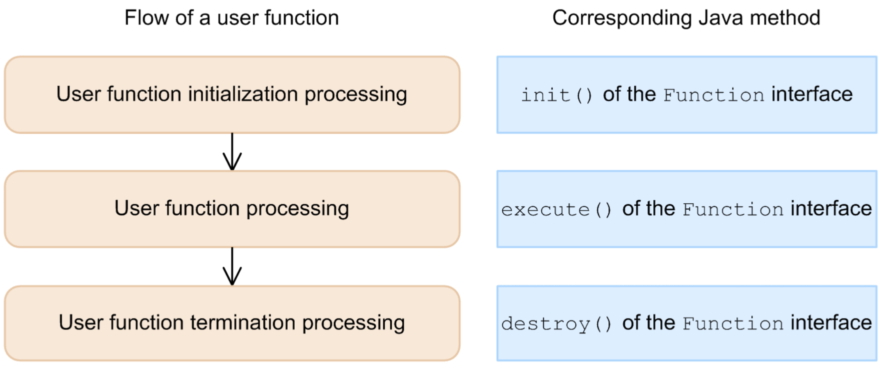
Distribute the `jar` file to the execution environment.

17.2 Creating a source program (user function)

This section explains how to create user functions and provides related notes.

17.2.1 Flow of a user function

The following figure shows a flow of a user function.


Flow of a user function
Corresponding Java method
User function initialization processing
init() of the Function interface
User function processing
execute() of the Function interface
User function termination processing
destroy() of the Function interface

(1) Importing the EADS-provided packages

Import the following EADS-provided packages:

```
import com.hitachi.software.xeads.func.Function;
import com.hitachi.software.xeads.func.FunctionContext;
```

(2) Implement user function initialization processing

Use `init()` of the `Function` interface to implement the user function initialization processing that is called when the EADS server starts up.

You can acquire the EADS server information with the `FunctionContext` argument of `init()`. Note that data in cache cannot be obtained because `init()` is called when the EADS server starts up.

If an exception occurs during the execution of `init()`, this user function cannot be used.

(3) Implement user function processing

Use `execute()` of the `Function` interface to implement the user function processing that is executed on the EADS server when it is requested by the EADS client.

You can acquire the EADS server information and manipulate data in cache with the `FunctionContext` argument of `execute()`.

For details about the operations that can be performed by user functions, see *18.2 API interfaces supported in user functions*.

(4) Implement user function termination processing

Use `destroy()` of the `Function` interface to implement the user function termination processing that is called when the EADS server is terminated.

You can acquire the EADS server information with the `FunctionContext` argument of `destroy()`.

Note that if user function initialization processing fails, `destroy()` is not called.

17.3 Notes about creating user functions

This section provides notes about creating user functions.

17.3.1 Notes about jar file names

The following notes apply to `jar` file names.

- Do not use a file name that begins with `eads`.
- Specify the file names in the following format:
  any-character-string`_function.jar`
  any-character-string can consist of alphanumeric characters (`0` to `9`, `A` to `Z`, `a` to `z`) and underscores (_).

17.3.2 Notes about package names

The following notes apply to package names.

- Do not use a package name beginning with `com.hitachi.software.xeads`.
- The package names and class names of user functions can consist of alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`) and underscores (_).

17.3.3 Notes about implementing user functions

The following notes apply to implementing user functions.

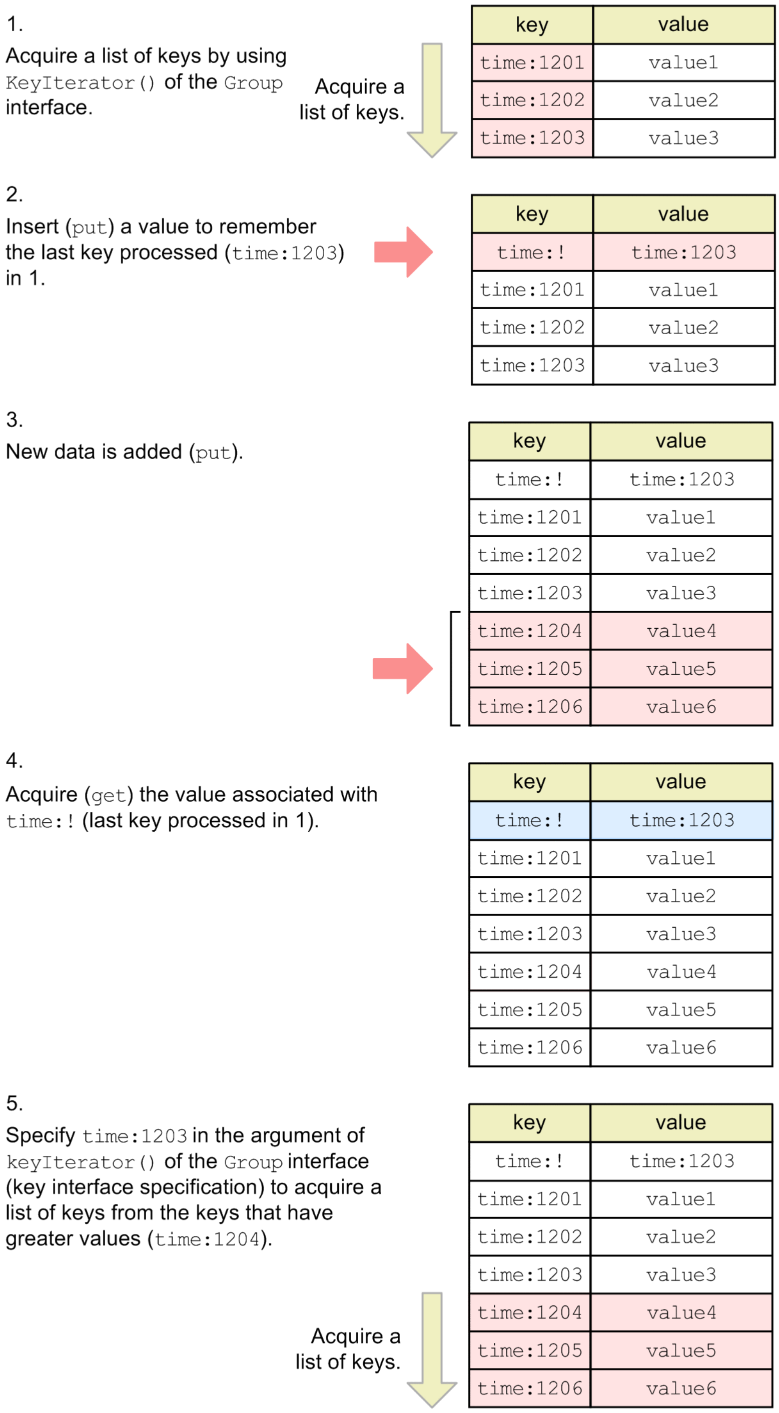
- Set the access qualifier of the default constructor of user functions to `public` so that it can be used by the EADS servers.
- Use serializable objects for the arguments and the return value from the EADS client.
- If you acquire data by using the `Group` interface, use one of the following methods to add to the class path the information needed to deserialize objects:
  - Specify the path of the `jar` file in the `Class-Path` attribute of the `jar` manifest.
  - Place the `jar` file under the *management-directory*`/app/lib` directory.
- Only one instance of a user function is supported. If multiple instances of a user function are executed concurrently on multiple EADS clients, the EADS servers attempt to use the same instance in multiple threads. Therefore, do not implement the following processes in the user functions:
  - Updating instance variables and the `static` variable without locking
  - Using an API function that is not thread-safe without locking
- The scope of `FunctionContext` and the objects that can be acquired from it is only within the `Function` interface's methods. Operation is not guaranteed if these objects are referenced outside the range of those methods.
- Do not create threads within a user function. Operation is not guaranteed if threads are created and executed within a user function.
- Because user functions are run in an EADS server processes, the current directory is the management directory.

- When user functions are executed from C client application programs, the only objects that can be handled as user function return values are byte arrays and `null`. An error will result if an object that is not a byte array or `null` is set as a return value.
- When user function arguments are passed to C client application programs, the only objects supported by the user functions are byte arrays.

17.3.4 How to acquire a list of keys efficiently

If a group of keys is stored in ascending order of the ASCII codes and a user function is used to acquire a list of keys asynchronously, a set of keys acquired once can be excluded from the keys capable of being acquired thereafter (so that afterwards, a list is acquired of only those keys that have not already been acquired). This is done by combining `keyIterator()` and `keyIterator()` (key interface specification) of the `Group` interface.

This method allows you to acquire a list of keys efficiently because you do not have to acquire a list of all keys each time data is added.



1.
Acquire a list of keys by using `KeyIterator()` of the `Group` interface.

key	value
time:1201	value1
time:1202	value2
time:1203	value3

2.
Insert (`put`) a value to remember the last key processed (`time:1203`) in 1.

key	value
time:!	time:1203
time:1201	value1
time:1202	value2
time:1203	value3

3.
New data is added (`put`).

key	value
time:!	time:1203
time:1201	value1
time:1202	value2
time:1203	value3
time:1204	value4
time:1205	value5
time:1206	value6

4.
Acquire (`get`) the value associated with `time:!` (last key processed in 1).

key	value
time:!	time:1203
time:1201	value1
time:1202	value2
time:1203	value3
time:1204	value4
time:1205	value5
time:1206	value6

5.
Specify `time:1203` in the argument of `keyIterator()` of the `Group` interface (key interface specification) to acquire a list of keys from the keys that have greater values (`time:1204`).

key	value
time:!	time:1203
time:1201	value1
time:1202	value2
time:1203	value3
time:1204	value4
time:1205	value5
time:1206	value6

For details about the `Group` interface, see *18.2.9 Group interface*.

To acquire a list of keys in descending order of the ASCII codes, combine `descendingKeyIterator()` and `descendingKeyIterator()` (key interface specification) of the `Group` interface.

Tip

You can acquire a list efficiently by specifying as keys a sequence (such as date and time) that corresponds to the key storage sequence.

17.4 Creating a function property file (optional)

Create a function property file, if necessary.

There are two ways to create function property files:

Creating a function property file that is applied to all user functions

The file name is `eads_function.properties`.

If no function property file is provided, default values are assumed for all parameters required for execution.

For details about the parameters of function properties, see *9.2.3 Parameters related to thread pools and connection pools*.

Creating a function property file for each user function's jar file

The file name is in the following format:

any-character-string`_function.properties`

For *any-character-string*, specify the name of the `jar` file that you want to apply but without the `_function.jar` part.

Example: The `jar` file name is `sample_function.jar`

The function property file name is `sample_function.properties`.

You can specify `eads_function.properties` together with *any-character-string*`_function.properties`. When they are specified together and the same parameters are defined in both property files, the parameters in *any-character-string*`_function.properties` are applied.

If no function property file is provided, the contents of `eads_function.properties` are applied.

> **Important note**
>
> In the case of the `jar` file name (`user-function.jar`) of a user function whose version is 03-00 or earlier, the function property file name is `user_function.properties`.

In the function properties, you can specify the maximum number of simultaneous threads for each user function by using the `eads.function.`*user-function-name*`.maxExecuteThreads` parameter.

You can also define user-specific parameters.

For details about designing the maximum number of simultaneous threads for each user function, see *9.2.2 Setting the maximum number of simultaneous threads*.

You can acquire the parameters specified in the function properties within the user function by using the `InitConfig` interface.

To create a function property file:

1. Edit the function property file.

2. Store the function property file in the following directory:

```
management-directory/app
```

The following shows an example of a function property file.

```
eads.function.com.abc.def.outputGroupKeys.max_execute_threads=10
eads.function.com.abc.def.getSpectialKey.max_execute_threads=5
eads.function.com.abc.def.updateInstance.max_execute_threads=2

# User-specific parameters
com.FunctionSample.def.version=0100
com.FunctionSample.def.isUpdateEnabled=true
```

Important note

The name of a user-specific parameter cannot begin with `eads.`. If a user-specific parameter whose name begins with `eads.` is used, operation is not guaranteed.

17.5 Compiling source programs (user functions)

This section explains how to compile user functions.

17.5.1 How to compile user functions

This subsection explains how to compile user functions.

(1) Placement of the source program

Place a created user function at a desired location.

(2) Compiling

To compile the source program of a created user function, use the JDK `javac` command, which is stored in the directory shown below.

For details about the `javac` command, see the Java compiler-related documentation.

```
/opt/hitachi/xeads/PSB/jdk
```

Specify the following libraries in the class path:

```
/opt/hitachi/xeads/server/lib/eads-function.jar
```

17.6 Packaging the user functions

This section explains how to create a `jar` file to store the `class` files compiled in *17.5 Compiling source programs (user functions)*.

17.6.1 How to package user functions

This subsection explains how to create a `jar` file to store the `class` files compiled as explained in *17.5 Compiling source programs (user functions)*.

(1) Specify a manifest (optional)

Specify a class that is used only by the user function in the `Class-Path` attribute in the manifest file (`MANIFEST.MF`) of the `jar` file.

(2) Create the jar file

Create the `jar` file by using the JDK `jar` command, which is stored in the directory below.

For details about the `jar` command, see the related Java documentation.

```
/opt/hitachi/xeads/PSB/jdk
```

17.7 Deploying user functions

This section explains how to deploy user functions. Deploy user functions while the EADS server is stopped.

17.7.1 How to deploy user functions

This subsection explains how to deploy user functions.

(1) Terminate the EADS server

Use the `eztool stop` command to terminate the EADS server.

(2) Copy the created jar file

Copy the `jar` file created in *17.6.1(2) Create the jar file* to the following directory:

```
management-directory/app
```

(3) Copy the function property file (optional)

If you have created a function property file, copy it to the following directory:

```
management-directory/app
```

(4) Place the libraries in the appropriate directory (optional)

If necessary, place the libraries that will be used by user functions in the directory shown below. Only those libraries whose extension is `jar` are valid.

```
management-directory/app/lib
```

If the path of the `jar` file is specified in the `Class-Path` attribute in the manifest file (`MANIFEST.MF`) of the `jar` file, copy the `jar` file to that path.

(5) Start the EADS server

Execute the `ezserver` command to start the EADS server.

(6) Check whether the user function has been deployed (optional)

Use the `eztool listfunc` command to check whether the user function has been deployed successfully.

For details about this procedure, see *11.11 Checking whether user functions have been placed correctly on individual EADS servers and whether they can be executed*.

17.8 Executing user functions

This section explains how to execute the user functions to make sure that they operate correctly.

17.8.1 Call a user function

To call a user function, use the following method of the EADS client:

(1) User functions that are executed with a key or a group specified

Java

`executeFunction()` (key specification or group specification) of the `Cache` class

C

`ead_execute_function()`

(2) User functions that are executed with an EADS server specified

Java

`executeFunction()` (EADS server specification) of the `Cache` class

C

`ead_execute_node_function()`

17.8.2 Output information to the user logs

You can output information about user function execution (user logs) by using the `UserLogger` interface. The user logs are output under the file name `eads_user_message[`*n*`].log` (`[`*n*`]` indicates the sequence number of the file) in the directory specified in the `eads.logger.dir` parameter in the server properties.

17.8.3 Notes about running user functions

- User functions can be executed only on active EADS servers. If an EADS server's status is changed while a user function is running, any data operations performed thereafter by that user function will result in an error.
- If the EADS server process is forcibly terminated by the `eztool forcestop` command or an OS function, the `destroy` method of the `Function` interface is not called.
- The user is responsible for maintaining the consistency of user functions among the EADS servers. If an attempt is made to execute a user function when it is not located on a specific EADS server, an error is returned to the EADS client.
- The EADS servers execute the user functions even if their implementation differs among the EADS servers.

17.9 Distributing the directory to the execution environment

Copy the *management-directory*`/app` directory to the *management-directory*`/app` directory on the EADS server in the execution environment.

18 Application Programming Interface Reference (Java)

This chapter explains the application programming interface (API) for Java supported by EADS.

18.1 Classes provided by the Java client libraries

The Java client libraries provide the classes listed in the table below as an EADS application program interface (API). You can use these classes by specifying the EADS-provided package names in the source programs coded in Java.

Note that the classes supported by the Java client libraries are all thread-safe.

Table 18–1: Class and package names supported by the Java client libraries

No.	Class name	Description	Package name
1	`Cache`	This class is used for manipulating data.	`com.hitachi.software.xeads.client.api`
2	`CacheManager`	This class is used for managing caches.	
3	`Node`	This class is used for obtaining information about an EADS server.	
4	`FailureOperationInfo`	This class is used for storing information about failed operations when part or all of a batch operation fails.	
5	`CacheException`	This is an exception class that is returned when an operation on the `Cache` and `CacheManager` classes fails.	
6	`InitializeException`	This is a subclass of `CacheException` that is returned when initialization of the `CacheManager` class results in an error.	
7	`InternalClientException`	This is a subclass of `CacheException` that is returned when an internal error occurs on an EADS client.	
8	`InternalServerException`	This is a subclass of `CacheException` that is returned when an internal error occurs on an EADS server.	
9	`ServerCommunicationException`	This is a subclass of `CacheException` that is returned in the event of a communication error.	
10	`UserOperationException`	This is a subclass of `CacheException` that is returned when an error occurs due to an illegal user operation.	
11	`BatchOperationException`	This is a subclass of `CacheException` that is returned when part or all of a batch operation fails.	
12	`AllFailureException`	This is a subclass of `BatchOperationException` that is returned when all of a batch operation fails.	

No.	Class name	Description	Package name
13	PartFailureException	This is a subclass of BatchOperationException that is returned when part of a batch operation fails.	

18.1.1 Cache class

(1) Description

This is a class used for manipulating data.

(2) Inheritance relationship

```
java.lang.Object
 └com.hitachi.software.xeads.client.api.Cache
```

(3) Format

```
public class Cache
extends java.lang.Object
```

(4) List of methods

The following table lists and describes the methods provided by the Cache class:

Method name	Description
getName()	Acquires a cache name associated with an instance of the Cache class.
put()	Associates a value with a key, and then stores it.
putAll()	Using a batch operation, this method stores the keys and values that are associated in a specified map.
create()	Associates a value with a key, and then stores it only when a new key is stored.
update()	Associates a value with a key, and then stores it only if the specified key has already been stored.
replace()	Compares the value associated with a specified key with a value (comparativeValue) specified as a condition. Only if the values match, this method associates the value with the key, and then stores it.
get()	Acquires a value associated with a specified key.
getAll() (set specification)	Using a batch operation, this method acquires the values associated with a specified list of keys. The acquired values are associated with keys and stored in a map specified in the argument.
getAll() (group specification)	Using a batch operation, this method acquires the values associated with the keys that belong to a specified group and its lower hierarchy groups. The acquired values are associated with keys and stored in a map specified in the argument.
remove()	Deletes a specified key and the value associated with that key.
removeAll() (set specification)	Using a batch operation, this method deletes the values associated with a specified list of keys.

Method name	Description
removeAll() (group specification)	Using a batch operation, this method deletes the keys and values that belong to a specified group, including the keys and values that belong to lower hierarchy groups.
removeAll() (EADS server specification)	Using a batch operation, this method deletes the keys and values that were copied from a specified EADS server.
getGroupNameSet()	Acquires a list of the group names of the groups in the highest hierarchy that are stored on a specified EADS server. The group names are listed in ascending order based on their ASCII code values.
getKeySet() (group specification)	Acquires a list of keys that belong to a specified group. The list of keys includes the keys that belong to the groups under the specified group's hierarchy. The keys are listed in ascending order based on their ASCII code values.
getKeySet() (EADS server specification)	Acquires a list of keys stored on a specified EADS server. The keys are listed in ascending order based on their ASCII code values.
getGroupCount()	Acquires the number of groups in the highest hierarchy that are stored on a specified EADS server.
getKeyCount() (group specification)	Acquires the number of keys that belong to a specified group. The number of keys that will be acquired includes the keys that belong to the groups under the specified group's hierarchy.
getKeyCount() (EADS server specification)	Acquires the number of keys stored on a specified EADS server.
getFirstKey() (group specification)	Acquires the first key in ascending order based on its ASCII code value from among all the keys that belong to a specified group. The keys that belong to the groups under the specified group's hierarchy are also subject to this acquisition processing.
getFirstKey() (EADS server specification)	Acquires the first key in ascending order based on its ASCII code value from among all the keys that are stored on a specified EADS server.
getNextKey() (group specification and key specification)	Acquires the key that immediately follows a specified key in ascending order based on its ASCII code value from among all the keys that belong to a specified group. The keys that belong to the groups under the specified group's hierarchy are also subject to this acquisition processing.
getNextKey() (EADS server specification and key specification)	Acquires the key that immediately follows a specified key in ascending order based on its ASCII code value from among all the keys that are stored on a specified EADS server.
executeFunction() (key specification or group specification)	Determines from a specified key or group the EADS server on which a user function is to be executed and then executes that user function.
executeFunction() (key specification or group specification, and reception timeout specification)	Determines from a specified key or group the EADS server on which a user function is to be executed and then executes that user function. This method also sets a reception timeout value.
executeFunction() (EADS server specification)	Executes a user function with an EADS server specified.
executeFunction() (EADS server specification and reception timeout specification)	Executes a user function with an EADS server specified and sets a reception timeout value.

(5) getName()

(a) Description

This method acquires a cache name associated with an instance of the `Cache` class.

You can also use this method after an access to a cache is terminated by `removeCache()` of the `CacheManager` class.

(b) Format

```
public String getName()
```

(c) Return value

This method returns the name of the cache associated with the instance of the `Cache` class.

(6) put()

(a) Description

This method associates a value with a key, and then stores it.

If a problem occurs when the value is stored, the method returns an exception.

(b) Format

```
public void put(String key,
                Object value)
          throws CacheException
```

(c) Parameters

`key`

Specifies a key to be associated with the value.
For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`value`

Specifies the value to be stored
For details about the data types that can be specified, see *15.2.2(3) Data types that can be specified as values*.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(7) putAll()

(a) Description

Using a batch operation, this method stores the keys and values that are associated in a specified map.

If a problem occurs when a value is stored, the method returns an exception.

(b) Format

```
public void putAll(java.util.Map<String,? extends Object> map)
           throws CacheException
```

(c) Parameters

`map`

Using a batch operation, this parameter specifies a map that associates keys and values that are stored.

If `null` or a map containing no element is specified, an error results.

For details about the keys that can be stored in caches, see *15.2.2(1) Data types that can be specified as keys*.

For details about the values that can be stored in caches, see *15.2.2(3) Data types that can be specified as values*.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `AllFailureException` (failure of entire batch operation)
- `PartFailureException` (failure of part of the batch operation)
- `InternalClientException` (EADS client internal error)

(e) Notes

- If all value storage operations failed, exception `AllFailureException` and error code `CacheException.EAD_ERROR_BATCH_FAILED_ALL` are returned.

  If some but not all of the value storage operations failed, exception `PartFailureException` and error code `CacheException.EAD_ERROR_BATCH_FAILED_PART` are returned.

  By obtaining information from each exception, you can determine the key operation that failed and the cause of the failure.
- If some of the cache operations failed, determine the operations that failed from the exception and re-execute this method, if necessary.
- Because caches are not locked while operations are ongoing, a target value might be changed by another cache operation during batch operation.
- When batch operations are performed on a large amount of data, a large amount of memory might be required by EADS clients and EADS servers.
- When batch operations are performed on a large amount of data, it might take a long time to complete the processing. To ensure proper operation, make sure that you design a timeout value that is appropriate for the processing time.

(8) create()

(a) Description

This method associates a value with a key, and then stores it only when a new key is stored.

If a problem occurs when the value is stored, the method returns an exception.

(b) Format

```
public void create(String key,
                   Object value)
            throws CacheException
```

(c) Parameters

`key`

Specifies a key to be associated with the value.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`value`

Specifies the value to be stored.

For details about the data types that can be specified, see *15.2.2(3) Data types that can be specified as values*.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(9) update()

(a) Description

This method associates a value with a key, and then stores it only when the specified key has already been stored.

If a problem occurs when the value is stored, the method returns an exception.

(b) Format

```
public void update(String key,
                   Object value)
            throws CacheException
```

(c) Parameters

`key`

Specifies a key to be associated with the value.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`value`

Specifies the value to be stored.

For details about the data types that can be specified, see *15.2.2(3) Data types that can be specified as values*.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(10) replace()

(a) Description

This method compares the value associated with a specified key with a value (`comparativeValue`) specified as a condition. Only if the values match, this method associates the value with the key, and then stores it.

If a problem occurs when the value is replaced, the method returns an exception.

(b) Format

```
public void replace(String key,
                    Object value,
                    Object comparativeValue)
             throws CacheException
```

(c) Parameters

`key`

Specifies a key that is associated with the value to be replaced.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`value`

Specifies the value to be stored.

For details about the data types that can be specified, see *15.2.2(3) Data types that can be specified as values*.

`comparativeValue`

Specifies the value to be compared.

For details about the data types that can be specified, see *15.2.2(3) Data types that can be specified as values*.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(11) get()

(a) Description

This method acquires a value associated with a specified key.

If a problem occurs when the value is acquired, the method returns an exception.

(b) Format

```
public java.lang.Object get(String key)
                     throws CacheException
```

(c) Parameters

`key`

Specifies a key that is associated with the value to be acquired.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

(d) Return value

This method returns the value associated with the key.

If nothing is associated with the specified key, it returns `null`.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(f) Notes

Do not change the context class loader for the thread that executes `get()`. If the context class loader settings are invalid, deserialization of the return value object fails.

(12) getAll() (set specification)

(a) Description

Using a batch operation, this method acquires the values associated with a specified list of keys. The acquired values are associated with keys and stored in a map specified in the argument.

If a problem occurs when a value is acquired, the method returns an exception.

(b) Format

```
public void getAll(java.util.Set<String> keys,
                   java.util.Map<String, Object> returnMap)
            throws CacheException
```

(c) Parameters

`keys`

Specifies a list of keys that are associated with the values to be acquired.

If `null` or a list of keys that contains no elements is specified, an error results.

For details about the data that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`returnMap`

Specifies a map for storing the acquired values.

If `null` is specified, this parameter is invalid.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `AllFailureException` (failure of entire batch operation)
- `PartFailureException` (failure of part of the batch operation)
- `InternalClientException` (EADS client internal error)

(e) Notes

- If all value acquisition operations failed, exception `AllFailureException` and error code `CacheException.EAD_ERROR_BATCH_FAILED_ALL` are returned.

  If some but not all of the value acquisition operations failed, exception `PartFailureException` and error code `CacheException.EAD_ERROR_BATCH_FAILED_PART` are returned.

  By obtaining information from each exception, you can determine the key operation that failed and the cause of the failure.
- If the map specified in the argument already contains keys and values to be acquired, those keys and values are overwritten by the acquired values.
- If no value is stored in cache for a specified key, nothing is stored in the map specified in the argument.
- If some of the cache operations failed, determine the operation that failed from the exception and re-execute this method, if necessary.
- Because caches are not locked while operations are ongoing, a target value might be changed by another cache operation during batch operation.
- When batch operations are performed on a large amount of data, a large amount of memory might be required by EADS clients and EADS servers.
- When batch operations are performed on a large amount of data, it might take a long time to complete the processing. To ensure proper operation, make sure that you design a timeout value that is appropriate for the processing time.

(13) getAll() (group specification)

(a) Description

Using a batch operation, this method acquires the values associated with the keys that belong to a specified group and its lower hierarchy groups. The acquired values are associated with the keys and stored in a map specified in the argument.

If acquisition of a value fails for some reason during batch acquisition of values, only the values that were acquired successfully are stored in the map. The method returns an exception indicating the cause of the failure.

(b) Format

```
public void getAll(String groupName,
                   java.util.Map<String, Object> returnMap)
            throws CacheException
```

(c) Parameters

`groupName`

Specifies the name of the group for which values are to be acquired.

For details about the data that can be specified, see *15.2.2(2) Data that can be specified as group names*.

`returnMap`

Specifies a map for storing the acquired values.

If `null` is specified, an error results.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(e) Notes

- If the map specified in the argument already contains keys and values to be acquired, those keys and values are overwritten by the acquired values.
- If no value is stored in cache for a specified key, nothing is stored in the map specified in the argument.
- If some but not all of the cache operations have failed, check the cache operation results, and then re-execute this method, if necessary.
- Because caches are not locked while operations are ongoing, a target value might be changed by another cache operation during batch operation.
- When batch operations are performed on a large amount of data, a large amount of memory might be required by EADS clients and EADS servers.
- When batch operations are performed on a large amount of data, it might take a long time to complete the processing. To ensure proper operation, make sure that you design a timeout value that is appropriate for the processing time.

(14) remove()

(a) Description

This method deletes a specified key and the value associated with that key.

If a problem occurs when a value is deleted, the method returns an exception.

(b) Format

```
public void remove(String key)
             throws CacheException
```

(c) Parameters

`key`

Specifies the key that is associated with the value to be deleted.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)

- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(15) removeAll() (set specification)

(a) Description

Using a batch operation, this method deletes the values associated with a specified list of keys.

If a problem occurs when a value is deleted, the method returns an exception.

(b) Format

```
public void removeAll(java.util.Set<String> keys)
               throws CacheException
```

(c) Parameters

`keys`

Specifies a list of keys that are associated with the values that are to be deleted.

If `null` or a list of keys that contains no elements is specified, an error results.

For details about the data that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `AllFailureException` (failure of entire batch operation)
- `PartFailureException` (failure of part of the batch operation)
- `InternalClientException` (EADS client internal error)

(e) Notes

- If all value deletion operations failed, exception `AllFailureException` and error code `CacheException.EAD_ERROR_BATCH_FAILED_ALL` are returned.

  If some but not all of the value deletion operations failed, exception `PartFailureException` and error code `CacheException.EAD_ERROR_BATCH_FAILED_PART` are returned.

  By obtaining information from each exception, you can determine the key operation that failed and the cause of the failure.
- If an exception is returned, data that was to be deleted might still remain. For this reason, it is important to determine from the exception which operation failed and then take appropriate action. If necessary, re-execute `removeAll()` (set specification).
- Because caches are not locked while operations are ongoing, a target value might be changed by another cache operation during batch operation.
- When batch operations are performed on a large amount of data, a large amount of memory might be required by EADS clients and EADS servers.
- When batch operations are performed on a large amount of data, it might take a long time to complete the processing. To ensure proper operation, make sure that you design a timeout value that is appropriate for the processing time.

(16) removeAll() (group specification)

(a) Description

Using a batch operation, this method deletes the keys and values that belong to a specified group, including the keys and values that belong to lower hierarchy groups.

If a problem occurs when a value is deleted, the method returns an exception.

(b) Format

```
public void removeAll(String groupName)
                throws CacheException
```

(c) Parameters

`groupName`

Specifies the name of the group to be deleted.

For details about the data that can be specified, see *15.2.2(2) Data that can be specified as group names*.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(e) Notes

- If an exception is returned, data that was to be deleted might still remain. For this reason, it is important to check the execution results and then take appropriate action. If necessary, re-execute `removeAll()` (group specification).
- Because caches are not locked while operations are ongoing, a target value might be changed by another cache operation during batch operation.
- When batch operations are performed on a large amount of data, a large amount of memory might be required by EADS clients and EADS servers.
- When batch operations are performed on a large amount of data, it might take a long time to complete the processing. To ensure proper operation, make sure that you design a timeout value that is appropriate for the processing time.

(17) removeAll() (EADS server specification)

(a) Description

Using a batch operation, this method deletes the keys and values that were copied from a specified EADS server.

If a problem occurs when a value is deleted, the method returns an exception.

(b) Format

```
public void removeAll(Node targetNode)
                throws CacheException
```

(c) Parameters

targetNode

Specifies the EADS server that will be performing the batch deletion processing.

You can specify only an instance of the Node class that has been acquired from the CacheManager class. If any other instance is specified, valid operation is not guaranteed.

(d) Exceptions

- UserOperationException (illegal user operation)
- ServerCommunicationException (communication error)
- InternalServerException (EADS server internal error)
- InternalClientException (EADS client internal error)

(e) Notes

- If an exception is returned, data that was to be deleted might still remain. For this reason, it is important to check the execution results and then take appropriate action. If necessary, re-execute removeAll() (EADS server specification).
- Because caches are not locked while operations are ongoing, a target value might be changed by another cache operation during batch operation.
- When batch operations are performed on a large amount of data, a large amount of memory might be required by EADS clients and EADS servers.
- When batch operations are performed on a large amount of data, it might take a long time to complete the processing. To ensure proper operation, make sure that you design a timeout value that is appropriate for the processing time.

(18) getGroupNameSet()

(a) Description

This method acquires a list of the group names of the groups in the highest hierarchy that are stored on a specified EADS server.

The group names are listed in ascending order based on their ASCII code values.

(b) Format

```
public java.util.Set<String> getGroupNameSet(Node targetNode)
                                    throws CacheException
```

(c) Parameters

targetNode

Specifies the EADS server whose group names are to be acquired.

You can specify only an instance of the Node class that has been acquired from the CacheManager class. If any other instance is specified, valid operation is not guaranteed.

(d) Return value

This method returns a list of the group names of the groups in the highest hierarchy that are stored on the specified EADS server.

If the specified EADS server does not contain any keys that belong to the group, the method returns `null`.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(f) Notes

As the number of groups on the specified EADS server increases, the time required for acquisition processing increases. The amount of resources required for the acquisition processing also increases.

(19) getKeySet() (group specification)

(a) Description

This method acquires a list of the keys that belong to a specified group. The list of keys includes the keys that belong to the groups under the specified group's hierarchy.

The keys are listed in ascending order based on their ASCII code values.

(b) Format

```
public java.util.Set<String> getKeySet(String groupName)
                                    throws CacheException
```

(c) Parameters

`groupName`

Specifies a group name.
For details about the data that can be specified, see *15.2.2(2) Data that can be specified as group names*.

(d) Return value

The method returns a list of keys that belong to the specified group.

If no keys belong to the specified group, the method returns `null`.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(f) Notes

As the number of keys in the specified group increases, the time required for acquisition processing increases. The amount of resources required for the acquisition processing also increases.

(20) getKeySet() (EADS server specification)

(a) Description

This method acquires a list of the keys that are stored on a specified EADS server.

The keys are listed in ascending order based on their ASCII code values.

(b) Format

```
public java.util.Set<String> getKeySet(Node targetNode)
                                throws CacheException
```

(c) Parameters

`targetNode`

Specifies the EADS server from which a list of keys is to be acquired.

You can specify only an instance of the `Node` class that has been acquired from the `CacheManager` class. If any other instance is specified, valid operation is not guaranteed.

(d) Return value

This method returns a list of keys stored on the specified EADS server.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(f) Notes

As the number of keys on the specified EADS server increases, the time required for acquisition processing increases. The amount of resources required for the acquisition processing also increases.

(21) getGroupCount()

(a) Description

This method acquires the number of groups in the highest hierarchy that are stored on a specified EADS server.

(b) Format

```
public int getGroupCount(Node targetNode)
                  throws CacheException
```

(c) Parameters

`targetNode`

Specifies the EADS server from which the number of groups is to be acquired.

You can specify only an instance of the `Node` class that has been acquired from the `CacheManager` class. If any other instance is specified, valid operation is not guaranteed.

(d) Return value

This method returns the number of groups in the highest hierarchy that are stored on the specified EADS server.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(22) getKeyCount() (group specification)

(a) Description

This method acquires the number of keys that belong to a specified group. The number of keys that will be acquired includes the keys that belong to the groups under the specified group's hierarchy.

(b) Format

```
public int getKeyCount(String groupName)
                throws CacheException
```

(c) Parameters

`groupName`

Specifies a group name.

For details about the data that can be specified, see *15.2.2(2) Data that can be specified as group names*.

(d) Return value

This method returns the number of keys that belong to the specified group.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(23) getKeyCount() (EADS server specification)

(a) Description

This method acquires the number of keys stored on a specified EADS server.

(b) Format

```
public int getKeyCount(Node targetNode)
                throws CacheException
```

(c) Parameters

`targetNode`

Specifies the EADS server from which the number of keys is to be acquired.

You can specify only an instance of the `Node` class that has been acquired from the `CacheManager` class. If any other instance is specified, valid operation is not guaranteed.

(d) Return value

This method returns the number of keys that belong to the specified EADS server.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(24) getFirstKey() (group specification)

(a) Description

This method acquires the first key in ascending order based on its ASCII code value from among all the keys that belong to a specified group. The keys that belong to the groups under the specified group's hierarchy are also subject to this acquisition processing.

(b) Format

```
public String getFirstKey(String groupName)
                   throws CacheException
```

(c) Parameters

`groupName`

Specifies a group name.

For details about the data that can be specified, see *15.2.2(2) Data that can be specified as group names*.

(d) Return value

This method returns the first key in ascending order based on its ASCII code value from among all the keys that belong to the specified group.

If no keys belong to the specified group, the method returns `null`.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(25) getFirstKey() (EADS server specification)

(a) Description

This method acquires the first key in ascending order based on its ASCII code value from among all the keys that are stored on a specified EADS server.

(b) Format

```
public String getFirstKey(Node targetNode)
                    throws CacheException
```

(c) Parameters

`targetNode`

Specifies the EADS server from which a key is to be acquired.

You can specify only an instance of the `Node` class that has been acquired from the `CacheManager` class. If any other instance is specified, valid operation is not guaranteed.

(d) Return value

This method acquires the first key in ascending order based on its ASCII code value from among all the keys that are stored on the specified EADS server.

If the specified EADS server contains no keys, the method returns `null`.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(26) getNextKey() (group specification and key specification)

(a) Description

This method acquires the key that immediately follows a specified key in ascending order based on its ASCII code value from among all the keys that belong to a specified group. The keys that belong to the groups under the specified group's hierarchy are also subject to this acquisition processing.

If the specified key does not exist on the connection-target EADS server, the method similarly acquires the key that immediately follows the specified key.

(b) Format

```
public String getNextKey(String groupName, String key)
                    throws CacheException
```

(c) Parameters

`groupName`

Specifies a group name.

For details about the data that can be specified, see *15.2.2(2) Data that can be specified as group names*.

`key`

Specifies the reference key.

For details about the data that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

(d) Return value

This method returns the key that immediately follows the specified key in ascending order based on its ASCII code value from among all the keys that belong to the specified group.

If no key follows the specified key, the method returns `null`.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(f) Notes

- This method determines the connection-target EADS server based on the specified group.
- The EADS server that stores the keys belonging to the specified group might have changed due to isolation, restoration, or addition (scale-out) processing on EADS servers other than the connection-target EADS server. Therefore, if you have obtained the reference key by executing `getFirstKey()` (group specification), a connection might be established with a different EADS server.
- Because groups are not locked on EADS servers, keys that belong to groups might be inserted or deleted by another process after `getFirstKey()` (group specification) or `getNextKey()` (group specification and key specification) was executed.

(27) getNextKey() (EADS server specification and key specification)

(a) Description

This method acquires the key that immediately follows a specified key in ascending order based on its ASCII code value from among all the keys that are stored on a specified EADS server.

If the specified key does not exist on the connection-target EADS server, the method similarly acquires the key that immediately follows the specified key.

(b) Format

```
public String getNextKey(Node targetNode, String key)
                     throws CacheException
```

(c) Parameters

`targetNode`

Specifies the EADS server from which a key is to be acquired.

You can specify only an instance of the `Node` class that has been acquired from the `CacheManager` class. If any other instance is specified, operation is not guaranteed.

`key`

Specifies the reference key.

For details about the data that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

(d) Return value

This method returns the key that immediately follows the specified key in ascending order based on its ASCII code value from among all the keys that are stored on the specified EADS server.

If no key follows the specified key, the method returns `null`.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(f) Notes

- The EADS server that stores the specified key might have changed due to isolation, restoration, or addition (scale-out) processing on EADS servers other than the connection-target EADS server.
- Because groups are not locked on EADS servers, keys that belong to groups might be inserted or deleted by another process after `getFirstKey()` (EADS server specification) or `getNextKey()` (EADS server specification and key specification) was executed.

(28) executeFunction() (key specification or group specification)

(a) Description

This method determines from a specified key or group the EADS server on which a user function is to be executed and then executes that user function.

If a problem occurs when the user function executes, the method returns an exception.

(b) Format

```
public Object executeFunction(String keyOrGroupName,
                              String funcName,
                              Object arg)
                       throws CacheException
```

(c) Parameters

`keyOrGroupName`

Specifies a key or a group name.

For details about the data that can be specified, see *15.2.2(1) Data types that can be specified as keys* or *15.2.2(2) Data that can be specified as group names*.

`funcName`

Specifies a user function name.

A user function name can consist of single-byte alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`), underscores (_), periods (`.`), and dollar signs (`$`).

There is no limit to the number of characters.

Specifying `null` or the null character string is invalid.

`arg`

Specifies the arguments to be passed to the user function.

If the object cannot be serialized, an error results.

There is no limit to the size of the object.

If no argument is passed, specify `null`.

(d) Return value

This method returns the execution results set by the user function.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(f) Notes

Do not change the context class loader for the thread that executes `executeFunction()`. If the context class loader settings are invalid, deserialization of the return value object fails.

(29) executeFunction() (key specification or group specification, and reception timeout specification)

(a) Description

This method determines from a specified key or group the EADS server on which a user function is to be executed and then executes that user function. This method also sets a reception timeout value.

The value specified in the `recvTimeout` argument is used as the reception timeout value, not the value of the `eads.client.connection.receive.timeout` parameter in the client properties.

If a problem occurs when the user function executes, the method returns an exception.

If a timeout occurs, the method returns `CacheException.EAD_ERROR_NET_TIMEOUT`.

(b) Format

```
public Object executeFunction(String keyOrGroupName,
                              String funcName,
                              Object arg,
                              int recvTimeout)
                       throws CacheException
```

(c) Parameters

`keyOrGroupName`

Specifies a key or a group name.

For details about the data that can be specified, see *15.2.2(1) Data types that can be specified as keys* or *15.2.2(2) Data that can be specified as group names*.

`funcName`

Specifies a user function name.

A user function name can consist of single-byte alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`), underscores (_), periods (`.`), and dollar signs (`$`).

There is no limit to the number of characters.

Specifying `null` or the null character string is invalid.

`arg`

Specifies the arguments to be passed to the user function.

If the object cannot be serialized, an error results.

There is no limit to the size of the object.

If no argument is passed, specify `null`.

`recvTimeout`

Specifies a data reception timeout value (in milliseconds).

For details about the data that can be specified, see *9.3.3(3)(b) eads.client.connection.receive.timeout*.

(d) Return value

This method returns the execution results set by the user function.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(f) Notes

Do not change the context class loader for the thread that executes `executeFunction()`. If any context class loader settings are invalid, deserialization of the return value object will fail.

(30) executeFunction() (EADS server specification)

(a) Description

This method executes a user function with an EADS server specified.

If a problem occurs when the user function executes, the method returns an exception.

(b) Format

```
public Object executeFunction(Node targetNode,
                              String funcName,
                              Object arg)
                        throws CacheException
```

(c) Parameters

`targetNode`

Specifies an EADS server (an instance of the `Node` class obtained from the `CacheManager` class). If any other instance is specified, correct operation is not guaranteed.

This parameter is invalid in the following cases:

- `null` is specified.
- The address information managed by the specified `Node` class (IP address and port number) does not match any EADS server address information maintained by the EADS client.

`funcName`

Specifies a user function name.

A user function name can consist of single-byte alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`), underscores (_), periods (`.`), and dollar signs (`$`).

There is no limit to the number of characters.

Specifying `null` or the null character string is invalid.

`arg`

Specifies the arguments to be passed to the user function.

If the object cannot be serialized, an error results.

There is no limit to the size of the object.

If no argument is passed, specify `null`.

(d) Return value

This method returns the execution results set by the user function.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(f) Notes

Do not change the context class loader for the thread that executes `executeFunction()`. If the context class loader settings are invalid, deserialization of the return value object fails.

(31) executeFunction() (EADS server specification and reception timeout specification)

(a) Description

This method executes a user function with an EADS server specified and sets a reception timeout value.

The value specified in the `recvTimeout` argument is used as the reception timeout value, not the value of the `eads.client.connection.receive.timeout` parameter in the client properties.

If a problem occurs when the user function executes, the method returns an exception.

If a timeout occurs, the method returns `CacheException.EAD_ERROR_NET_TIMEOUT`.

(b) Format

```
public Object executeFunction(Node targetNode,
                              String funcName,
                              Object arg,
                              int recvTimeout)
                       throws CacheException
```

(c) Parameters

`targetNode`

Specifies an EADS server (an instance of the `Node` class obtained from the `CacheManager` class). If any other instance is specified, valid operation is not guaranteed.

This parameter is invalid in the following cases:

- `null` is specified.
- The address information managed by the specified `Node` class (IP address and port number) does not match any EADS server address information maintained by the EADS client.

`funcName`

Specifies a user function name.

A user function name can consist of single-byte alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`), underscores (_), periods (`.`), and dollar signs (`$`).

There is no limit to the number of characters.

Specifying `null` or the null character string is invalid.

`arg`

Specifies the arguments to be passed to the user function.

If the object cannot be serialized, an error results.

There is no limit to the size of the object.

If no argument is passed, specify `null`.

`recvTimeout`

Specifies a data reception timeout value (in milliseconds).

For details about the data that can be specified, see *9.3.3(3)(b) eads.client.connection.receive.timeout*.

(d) Return value

This method returns the execution results set by the user function.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(f) Notes

Do not change the context class loader for the thread that executes `executeFunction()`. If any context class loader settings are invalid, deserialization of the return value object will fail.

18.1.2 CacheManager class

(1) Function

This is a class used for managing EADS's caches.

The class executes `create()` and initializes the EADS client.

Always make sure that you execute `destroy()` when you stop using the EADS client.

(2) Inheritance relationship

```
java.lang.Object
 └com.hitachi.software.xeads.client.api.CacheManager
```

(3) Format

```
public class CacheManager
extends java.lang.Object
```

(4) List of methods

The following table lists and describes the methods provided by the CacheManager class:

Method name	Description
create() (path specification)	Initializes an EADS client according to the client properties.
create() (EADS client name and path specification)	Initializes an EADS client according to the client properties.
create() (input stream specification)	Initializes an EADS client according to the client properties.
create() (EADS client name and input stream specification)	Initializes an EADS client according to the client properties.
create() (client properties specification)	Initializes an EADS client according to the client properties.
create() (EADS client name and client properties specification)	Initializes an EADS client according to the client properties.
getCache()	Acquires an instance of the Cache class.
removeCache()	Terminates accesses to the cache.
destroy()	Terminates usage of the EADS client.
getNodeList()	Acquires information about the connection-target EADS servers maintained by the EADS client.
[Deprecated] getNode()	[Deprecated] Acquires information about the original source EADS server that stores a specified key (or group) (and from which data has been copied).
getSlaveNodeList()	Acquires information about the original target EADS servers to which data stored on a specified EADS server is copied.
getCurrentMasterNode()	Acquires information about the source EADS server that currently stores a specified key (or group).
getOriginalMasterNode()	Acquires information about the original source EADS server that stores a specified key (or group).

(5) create() (path specification)

(a) Description

This method initializes an EADS client according to the client properties.

Each time this method is executed, a thread for monitoring communication timeouts and a thread for monitoring the cluster are generated. These threads are terminated when destroy() is executed.

If a problem occurs when the EADS client is initialized, the method returns an exception.

(b) Format

```
public static CacheManager create(String fileName)
                           throws CacheException
```

(c) Parameters

`fileName`

Specifies the path name of the storage location of the EADS client's client property file.

Specifying `null` or the null character string is invalid.

(d) Return value

This method returns an instance of the `CacheManager` class.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InitializeException` (`CacheManager` class initialization failure)
- `InternalClientException` (EADS client internal error)

(f) Notes

- If you will be executing multiple EADS clients concurrently on the same machine, specify a different log output destination for each EADS client. Valid operation cannot be guaranteed if you specify the same log output destination for multiple EADS clients that execute concurrently. For details about specifying the log output destination, see *8.4.2 Specifying the file output destinations*.
- If you execute this method on a Java EE server (uCosminexus Application Server), you must execute `destroy()` after executing this method. If you fail to do this, a memory leak will occur.

(6) create() (EADS client name and path specification)

(a) Description

This method initializes an EADS client according to the client properties.

Each time this method is executed, a thread for monitoring communication timeouts and a thread for monitoring the cluster are generated. These threads are terminated when `destroy()` is executed.

If a problem occurs when the EADS client is initialized, the method returns an exception.

(b) Format

```
public static CacheManager create(String clientName, String fileName)
                           throws CacheException
```

(c) Parameters

`clientName`

Specifies an EADS client name.

For details about the data that can be specified, see *15.2.2(5) Data that can be specified as EADS client names*.

For the relationships between EADS client names and log file output destinations, see *8.4.2 Specifying the file output destinations*.

`fileName`

Specifies the path name of the client property file storage location on the EADS client.

Specifying `null` or the null character string is invalid.

(d) Return value

This method returns an instance of the `CacheManager` class.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InitializeException` (`CacheManager` class initialization failure)
- `InternalClientException` (EADS client internal error)

(f) Notes

- If you will be executing multiple EADS clients concurrently on the same machine, specify a different log output destination for each EADS client. Valid operation cannot be guaranteed if you specify the same log output destination for multiple EADS clients that execute concurrently. For details about specifying the log output destination, see *8.4.2 Specifying the file output destinations*.
- If you execute this method on a Java EE server (uCosminexus Application Server), you must execute `destroy()` after executing this method. If you fail to do this, a memory leak will occur.

(7) create() (input stream specification)

(a) Description

This method initializes an EADS client according to the client properties.

Each time this method is executed, a thread for monitoring communication timeouts and a thread for monitoring the cluster are generated. These threads are terminated when `destroy()` is executed.

If a problem occurs when the EADS client is initialized, the method returns an exception.

(b) Format

```
public static CacheManager create(InputStream in)
                           throws CacheException
```

(c) Parameters

`in`

Specifies an input stream for importing the EADS client's client property file.

Specifying `null` is invalid.

(d) Return value

An instance of the `CacheManager` class is returned.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InitializeException` (`CacheManager` class initialization failure)
- `InternalClientException` (EADS client internal error)

(f) Notes

- If you will be executing multiple EADS clients concurrently on the same machine, specify a different log output destination for each EADS client. Valid operation cannot be guaranteed if you specify the same log output destination for multiple EADS clients that execute concurrently. For details about specifying the log output destination, see *8.4.2 Specifying the file output destinations*.
- If you execute this method on a Java EE server (uCosminexus Application Server), you must execute `destroy()` after executing this method. If you fail to do this, a memory leak will occur.

(8) create() (EADS client name and input stream specification)

(a) Description

This method initializes an EADS client according to the client properties.

Each time this method is executed, a thread for monitoring communication timeouts and a thread for monitoring the cluster are generated. These threads are terminated when `destroy()` is executed.

If a problem occurs when the EADS client is initialized, the method returns an exception.

(b) Format

```
public static CacheManager create(String clientName, InputStream in)
                               throws CacheException
```

(c) Parameters

`clientName`

Specifies an EADS client name.

For details about the data that can be specified, see *15.2.2(5) Data that can be specified as EADS client names*.

For the relationships between EADS client names and log file output destinations, see *8.4.2 Specifying the file output destinations*.

`in`

Specifies an input stream for importing the EADS client's client property file.

Specifying `null` is invalid.

(d) Return value

This method returns an instance of the `CacheManager` class.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InitializeException` (`CacheManager` class initialization failure)

- InternalClientException (EADS client internal error)

(f) Notes

- If you will be executing multiple EADS clients concurrently on the same machine, specify a different log output destination for each EADS client. Valid operation cannot be guaranteed if you specify the same log output destination for multiple EADS clients that execute concurrently. For details about specifying the log output destination, see *8.4.2 Specifying the file output destinations*.
- If you execute this method on a Java EE server (uCosminexus Application Server), you must execute destroy() after executing this method. If you fail to do this, a memory leak will occur.

(9) create() (client properties specification)

(a) Description

This method initializes an EADS client according to the client properties.

Each time this method is executed, a thread for monitoring communication timeouts and a thread for monitoring the cluster are generated. These threads are terminated when destroy() is executed.

If a problem occurs when the EADS client is initialized, the method returns an exception.

(b) Format

```
public static CacheManager create(Properties properties)
                           throws CacheException
```

(c) Parameters

properties

Specifies client properties of the EADS client.

Specifying null is invalid.

(d) Return value

This method returns an instance of the CacheManager class.

(e) Exceptions

- UserOperationException (illegal user operation)
- InitializeException (CacheManager class initialization failure)
- InternalClientException (EADS client internal error)

(f) Notes

- If you will be executing multiple EADS clients concurrently on the same machine, specify a different log output destination for each EADS client. Valid operation cannot be guaranteed if you specify the same log output destination for multiple EADS clients that execute concurrently. For details about specifying the log output destination, see *8.4.2 Specifying the file output destinations*.
- If you execute this method on a Java EE server (uCosminexus Application Server), you must execute destroy() after executing this method. If you fail to do this, a memory leak will occur.

(10) create() (EADS client name and client properties specification)

(a) Description

This method initializes an EADS client according to the client properties.

Each time this method is executed, a thread for monitoring communication timeouts and a thread for monitoring the cluster are generated. These threads are terminated when `destroy()` is executed.

If a problem occurs when the EADS client is initialized, the method returns an exception.

(b) Format

```
public static CacheManager create(String clientName, Properties properties)
                           throws CacheException
```

(c) Parameters

`clientName`

Specifies an EADS client name.

For details about the data that can be specified, see *15.2.2(5) Data that can be specified as EADS client names*.

For the relationships between EADS client names and log file output destinations, see *8.4.2 Specifying the file output destinations*.

`properties`

Specifies client properties of the EADS client.

Specifying `null` is invalid.

(d) Return value

This method returns an instance of the `CacheManager` class.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InitializeException` (`CacheManager` class initialization failure)
- `InternalClientException` (EADS client internal error)

(f) Notes

- If you will be executing multiple EADS clients concurrently on the same machine, specify a different log output destination for each EADS client. Valid operation cannot be guaranteed if you specify the same log output destination for multiple EADS clients that execute concurrently. For details about specifying the log output destination, see *8.4.2 Specifying the file output destinations*.
- If you execute this method on a Java EE server (uCosminexus Application Server), you must execute `destroy()` after executing this method. If you fail to do this, a memory leak will occur.

(11) getCache()

(a) Description

This method starts access to a cache and acquires an instance of the `Cache` class.

The method also places the instance of the cache whose access was terminated by `removeCache()` in active status again.

If a problem occurs when access to the cache begins, the method returns an exception.

(b) Format

```
public Cache getCache(String name)
                throws CacheException
```

(c) Parameters

`name`

Specifies the cache name of the `Cache` class that is to be acquired.

For details about the data types that can be specified, see *15.2.2(4) Data types that can be specified as cache names*.

(d) Return value

This method returns an instance of the `Cache` class that is associated with the specified cache name.

If `getCache()` has already been executed with the same cache name specified, it returns the same instance as for the initial execution.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `ServerCommunicationException` (communication error)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(12) removeCache()

(a) Description

This method terminates cache access.

If a problem occurs when access to the cache terminates, the method returns an exception.

(b) Format

```
public void removeCache(String name)
                  throws CacheException
```

(c) Parameters

`name`

Specifies the `Cache` class cache name that is to be terminated.

For details about the data types that can be specified, see *15.2.2(4) Data types that can be specified as cache names*.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalClientException` (EADS client internal error)

(e) Notes

Use this method carefully, especially when multiple threads are running, because this method terminates accesses to all caches (instances of the `Cache` class) that were acquired by specifying the same cache name.

(13) destroy()

(a) Description

This method performs termination processing on all caches and terminates usage of the EADS client.

If the EADS client has already been terminated by executing `destroy()`, this method returns `CacheException.EAD_ERROR_CLIENT_FINALIZED`.

(b) Format

```
public void destroy()
             throws CacheException
```

(c) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalClientException` (EADS client internal error)

(d) Notes

- Execute `destroy()` paired with `create()`. Execute `destroy()` only once for an instance acquired by `create()`.
- In the case of a Java EE server (uCosminexus Application Server), you must execute this method. If you fail to do so, a memory leak will occur.

(14) getNodeList()

(a) Description

This method acquires information about the connection-target EADS servers maintained by the EADS client.

If a problem occurs when information about the connection-target EADS server is acquired, the method returns an exception.

(b) Format

```
public Node[] getNodeList()
                   throws CacheException
```

(c) Return value

This method returns information about the connection-target EADS servers maintained by the EADS client.

The method returns information about all the EADS servers regardless of their EADS server status.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalClientException` (EADS client internal error)

(e) Notes

- Execution of this method does not establish communication with the EADS servers. Therefore, the acquired information might not be the most recent information.
- When this method is executed, whether the EADS servers are connected is not checked. Therefore, the acquired information might contain EADS servers that cannot be connected (because, for example, EADS servers are isolated). If you plan to use an acquired EADS server as a connection target, use `isEnable()` of the `Node` class to check whether a connection can be established with that EADS server.
  For details about `isEnable()` of the `Node` class, see *18.1.3(6) isEnable()*.

(15) [Deprecated] getNode()

> **Reference note**
>
> This method is deprecated. Instead, use the `getOriginalMasterNode()` method of the `CacheManager` class.

(a) Description

This method acquires information about the original source EADS server that stores a specified key (or group).

By original source EADS server is meant the EADS server that stores the original data of a specified key (or group) when all EADS servers making up the cluster can be connected successfully.

(b) Format

```
public Node getNode(String key)
             throws CacheException
```

(c) Parameters

`key`

Specifies a key (or a group).
For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

(d) Return value

This method returns an instance of the `Node` class that indicates the EADS server from which the specified key (or group) was copied.

Whether the connection-target EADS server can be connected has no effect on the return value.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalClientException` (EADS client internal error)

(f) Notes

- Execution of this method does not establish communication with the EADS server. Therefore, the acquired information might not be the most recent information.
- If there have been no changes to the cluster configuration, information about the same EADS server will always be acquired, regardless of whether that EADS server can be connected.
- If you plan to use the acquired EADS server as a connection target, use `isEnable()` of the `Node` class to check whether a connection can be established with that EADS server.
  For details about `isEnable()` of the `Node` class, see *18.1.3(6) isEnable()*.

(16) getSlaveNodeList()

(a) Description

This method acquires information about the original target EADS servers to which data stored on a specified EADS server is copied.

By original target EADS server is meant an EADS server to which data stored on a specified EADS server (source EADS server) is copied when all EADS servers making up the cluster can be connected successfully.

(b) Format

```
public Node[] getSlaveNodeList(Node masterNode)
                        throws CacheException
```

(c) Parameters

`masterNode`

Specifies the EADS server (an instance of the `Node` class obtained from the `CacheManager` class) that stores the copy source data. If any other instance is specified, correct operation is not guaranteed.
This parameter is invalid in the following cases:

- `null` is specified.
- The address information managed by the specified `Node` class (IP address and port number) does not match any EADS server address information maintained by the EADS client.

(d) Return value

This method returns information about the original target EADS servers as an array of the `Node` class.

Whether a connection-target EADS server can be connected has no effect on the return value.

If the data multiplicity is 1, the method returns an array containing no elements.

(e) Exceptions

- `UserOperationException` (illegal user operation)

- `InternalClientException` (EADS client internal error)

(f) Notes

- Execution of this method does not establish communication with the EADS servers. Therefore, the acquired information might not be the most recent information.
- When this method is executed, whether the specified EADS server and the target EADS servers for information acquisition are connected is not checked. Therefore, the acquired information might contain EADS servers that cannot be connected (because, for example, the EADS servers are isolated).

(17) getCurrentMasterNode()

(a) Description

This method acquires information about the source EADS server that currently stores a specified key (or group).

(b) Format

```
public Node getCurrentMasterNode(String key)
                          throws CacheException
```

(c) Parameters

`key`

Specifies a key (or a group).

For details about the data that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

(d) Return value

This method returns an instance of the `Node` class that indicates the source EADS server that currently stores the specified key (or group).

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `InternalClientException` (EADS client internal error)

(f) Notes

Execution of this method does not establish communication with the EADS server. Therefore, the acquired information might not be the most recent information.

(18) getOriginalMasterNode()

(a) Description

This method acquires information about the original source EADS server that stores a specified key (or group).

By original source EADS server is meant the EADS server that stores the master copy (source data) of a specified key (or group) when all EADS servers making up the cluster can be connected successfully. This EADS server might be different from the current source EADS server.

If there have been no changes to the cluster configuration, information about the same EADS server will always be acquired, regardless of whether the EADS server can be connected.

(b) Format

```
public Node getOriginalMasterNode(String key)
                              throws CacheException
```

(c) Parameters

`key`

Specifies a key (or a group).

For details about the data that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

(d) Return value

This method returns an instance of the `Node` class that indicates the original source EADS server that stores the specified key (or group).

Whether the connection-target EADS server can be connected has no effect on the return value.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalClientException` (EADS client internal error)

(f) Notes

- Execution of this method does not establish communication with the EADS server. Therefore, the acquired information might not be the most recent information.
- If you plan to use the acquired EADS server as a connection target, use `isEnable()` of the `Node` class to check whether a connection can be established with that EADS server.

  For details about `isEnable()` of the `Node` class, see *18.1.3(6) isEnable()*.

18.1.3 Node class

(1) Function

This is a class used for obtaining information about an EADS server.

(2) Inheritance relationship

```
java.lang.Object
 └com.hitachi.software.xeads.client.api.Node
```

(3) Format

```
public class Node
extends java.lang.Object
```

(4) List of methods

The following table lists and describes the methods provided by the `Node` class:

Method name	Description
`getNodeId()`	Acquires the EADS server ID.
`isEnable()`	Acquires a value indicating whether a connection can be established with an EADS server.
`getAddress()`	Acquires the IP address and port number of an EADS server.
`getPosition()`	Acquires the location (hash value) of an EADS server.
`toString()`	Acquires information about an EADS server (string representation).

(5) getNodeId()

(a) Description

This method acquires the EADS server ID.

(b) Format

```
public int getNodeId()
```

(c) Return value

This method returns the EADS server ID.

(6) isEnable()

(a) Description

This method acquires a value indicating whether a connection can be established with an EADS server.

(b) Format

```
public boolean isEnable()
```

(c) Return value

This method returns a value indicating whether a connection can be established with an EADS server.

`true`

Connection can be established.

`false`

Connection cannot be established.

(7) getAddress()

(a) Description

This method acquires the IP address and port number of an EADS server.

(b) Format

```
public java.net.InetSocketAddress getAddress()
```

(c) Return value

This method returns a `java.net.InetSocketAddress` instance indicating the IP address and port number of an EADS server.

(8) getPosition()

(a) Description

This method acquires the location (hash value) of an EADS server.

(b) Format

```
public int getPosition()
```

(c) Return value

This method returns the location (hash value) of an EADS server.

(9) toString()

(a) Description

This method acquires information about an EADS server (string representation).

(b) Format

```
public String toString()
```

(c) Return value

This method returns information about an EADS server (string representation) in the following format:

```
"node id = EADS-server-ID, status = ENABLE | DISABLE, position = EADS-
server-position, address = IP-address-and-port-number-of-EADS-server"
```

18.1.4 FailureOperationInfo class

(1) Description

This is a class used for storing information about failed operations when part or all of a batch operation fails.

(2) Inheritance relationship

```
java.lang.Object
 └com.hitachi.software.xeads.client.api.FailureOperationInfo
```

(3) Format

```
public class FailureOperationInfo
extends java.lang.Object
```

(4) List of methods

The following table lists and describes the methods provided by the `FailureOperationInfo` class:

Method name	Description
`getKey()`	Acquires the key that was used in a failed operation.
`getErrorCode()`	Acquires the error code that indicates the cause of an operation error.
`getException()`	Acquires the exception that occurred in a failed operation.

(5) getKey()

(a) Description

This method acquires the key that was used in a failed operation.

(b) Format

```
public String getKey()
```

(c) Return value

This method returns the key that was used by the failed operation.

(6) getErrorCode()

(a) Description

This method acquires the error code that indicates the cause of an operation error.

(b) Format

```
public int getErrorCode()
```

(c) Return value

This method returns the error code that indicates the cause of the operation error.

(7) getException()

(a) Description

This method acquires the exception that occurred in a failed operation.

(b) Format

```
public CacheException getException()
```

(c) Return value

This method returns the exception that occurred in the failed operation.

18.1.5 CacheException class

(1) Description

This is an exception class that is returned when an operation on the `Cache` and `CacheManager` classes fails.

Use `getErrorCode()` to obtain an error code to determine the nature of the error.

(2) Inheritance relationship

```
java.lang.Object
└java.lang.Throwable
  └java.lang.Exception
    └com.hitachi.software.xeads.client.api.CacheException
```

(3) Format

```
public class CacheException
extends Exception
```

(4) List of methods

The following table lists and describes the methods provided by the `CacheException` class:

Method name	Description
`getErrorCode()`	Acquires an error code for an exception that has occurred.

(5) getErrorCode()

(a) Description

This method acquires an error code for an exception that has occurred.

(b) Format

```
public int getErrorCode()
```

(c) Return value

This method returns an error code as a return value. The following table lists the error codes and describes the nature and cause of the error:

Error code	Error code literal	Exception class	Nature of error	Cause of error	Processing status of data updating API method#	Error code when 0300 is specified in the eads.client.compat parameter in the client properties
1000	EAD_ERROR_UNEXPECTED	CacheException	An unexpected error occurred.	An unexpected error occurred within the program.	U	1000
1010	EAD_ERROR_INVALID_PARAMETER	UserOperationException	A specified parameter is invalid.	An invalid parameter was specified in the API method argument.	N	1010
1030	EAD_ERROR_CLIENT_FINALIZED	UserOperationException	The processing cannot be performed because usage of the EADS client has been terminated.	A method of CacheManager class was executed after destroy() of the CacheManager class was executed.	N	1030
1040	EAD_ERROR_CACHE_NOT_STARTED	UserOperationException	The processing cannot be performed because the cache has not been started.	Possible causes are as follows: • An attempt was made to manipulate data after the cache was terminated (after removeCache() of the CacheManager class was executed). • An attempt was made to manipulate data after usage of the EADS client was terminated (after destroy() of the CacheManager class was executed).	N	1040
1050	EAD_ERROR_NOT_SERIALIZABLE	UserOperationException	Serialization processing failed on the EADS client.	The object specified in the API method argument cannot be serialized.	N	1050
1060	EAD_ERROR_NOT_DESERIALIZABLE	UserOperationException	Deserialization processing failed on the EADS client.	The object acquired from the EADS server could not be deserialized on the EADS client.	--	1060

Error code	Error code literal	Exception class	Nature of error	Cause of error	Processing status of data updating API method#	Error code when 0300 is specified in the eads.client.compat parameter in the client properties
				The object might not be included in the class path of the EADS client.		
1100	EAD_ERROR_CACHE_NOT_NEED_STOP	UserOperationException	An attempt was made to stop a cache that has already stopped.	The following are possible causes of the error: • An attempt was made to terminate a cache after the cache had already been terminated (after removeCache() of the CacheManager class was executed). • removeCache() of the CacheManager class was executed without executing getCache() of the CacheManager class.	--	1100
1110	EAD_ERROR_INVALID_NODE_ADDRESS	UserOperationException	The specified EADS server's address information does not match any EADS server address information maintained by the EADS client.	The address information (IP address and port number) of the EADS server specified in the argument of the API method does not match any EADS server address information maintained by the EADS client.	--	1110
1120	EAD_ERROR_EXCEED_MAX_CONNECTION_POOL_SIZE	UserOperationException	The number of connections to be pooled for the same connection target has already reached the maximum value and all of them are in use.	The number of concurrent threads issuing requests to the same EADS server has exceeded the maximum number of connections.	N	1120
2000	EAD_ERROR_INIT	InitializeException	EADS client initialization processing resulted in an error.	An unexpected error occurred during execution of create() of the CacheManager class.	--	2000
2010	EAD_ERROR_INIT_PROPERTIES	InitializeException	The client property file could not be imported.	Possible causes are as follows: • There was no client property file. • The client property file does not have read permissions. • The specified storage destination path name is a directory, not a file. • There is a problem in the input stream specified in create() of the CacheManager class.	--	2010

Error code	Error code literal	Exception class	Nature of error	Cause of error	Processing status of data updating API method#	Error code when 0300 is specified in the eads.client.compat parameter in the client properties
2020	EAD_ERROR_INIT_INVALID_PROPERTY	InitializeException	A definition in the client property file was invalid.	A definition in the client property file is invalid.	--	2020
2030	EAD_ERROR_INIT_LOGGER	InitializeException	Initialization of logs failed.	Possible causes are as follows: • The specified directory or log file at the output destination does not have write permissions. • The specified directory contains a file with the same name. • The specified path name or file name is invalid. • There is a directory that has the same name as the log file name. • There is not enough memory to start outputting logs.	--	2030
2040	EAD_ERROR_INIT_CLUSTERINFO	InitializeException	Establishment of a connection with the EADS server specified in the client property file failed.	Possible causes are as follows: • There is an error in the specification of the connection-target EADS server in the client properties. • Communication with the connection-target EADS server failed, or a failure occurred on the connection-target EADS server. • The maximum number of simultaneous connections to the EADS server has been exceeded. • The connection-target EADS server is not ready to accept requests.	--	2040
3000	EAD_ERROR_NET	ServerCommunicationException	An error occurred during communication with the EADS server.	Possible causes are as follows: • A network error occurred during communication. • A problem occurred on the EADS server with which the client was communicating.	U	3000

Error code	Error code literal	Exception class	Nature of error	Cause of error	Processing status of data updating API method#	Error code when 0300 is specified in the eads.client.compat parameter in the client properties
				• A problem occurred on the host with which the client was communicating.		
3001	EAD_ERROR_NET_SEND_REQUEST	ServerCommunicationException	A communication error occurred while a request was being sent to an EADS server.	Possible causes are as follows: • A network error occurred during communication. • A problem occurred on the EADS server with which the client was communicating. • A problem occurred on the host with which the client was communicating.	U	3000
3002	EAD_ERROR_NET_RECEIVE_RESPONSE	ServerCommunicationException	A communication error occurred while a response was being received from an EADS server.	Possible causes are as follows: • A network error occurred during communication. • A problem occurred on the EADS server with which the client was communicating. • A problem occurred on the host with which the client was communicating.	U	3000
3010	EAD_ERROR_NET_TIMEOUT	ServerCommunicationException	A timeout occurred during communication with the EADS server.	Possible causes are as follows: • A problem occurred on the connection-target EADS server. • A problem occurred on the connection-target host. • A problem occurred on the network. • The specified timeout value is invalid.	U	3010
3011	EAD_ERROR_NET_SEND_TIMEOUT	ServerCommunicationException	A timeout occurred while a request was being sent to an EADS server.	Possible causes are as follows: • A problem occurred on the connection-target EADS server. • A problem occurred on the connection-target host.	U	3010

Error code	Error code literal	Exception class	Nature of error	Cause of error	Processing status of data updating API method#	Error code when 0300 is specified in the eads.client.compat parameter in the client properties
				• A problem occurred on the network. • The specified timeout value is invalid.		
3012	EAD_ERROR_NET_RECEIVE_TIMEOUT	ServerCommunicationException	A timeout occurred while a response was being received from an EADS server.	Possible causes are as follows: • A problem occurred on the connection-target EADS server. • A problem occurred on the connection-target host. • A problem occurred on the network. • The specified timeout value is invalid.	U	3010
3020	EAD_ERROR_NET_CONNECTION	ServerCommunicationException	Connection establishment with the EADS server failed.	Possible causes are as follows: • A problem occurred on the connection-target EADS server. • The settings related to the connection-target EADS server are invalid. • A problem occurred on the network. • The specified timeout value is invalid.	N	3020
3030	EAD_ERROR_NET_PROTOCOL	ServerCommunicationException	A protocol error occurred during communication with the EADS server.	The connection-target EADS server might be invalid.	U	3030
3040	EAD_ERROR_NET_CLUSTERINFO	ServerCommunicationException	Establishment of a connection with all connectable EADS servers failed.	Possible causes are as follows: • A problem occurred on the connection-target EADS server. • The settings related to the connection-target EADS server are invalid. • A problem occurred on the network. • The specified cache does not exist on the connection-target EADS server. • The connection-target EADS server has been closed. • The cluster information maintained by the EADS	--	3040

Error code	Error code literal	Exception class	Nature of error	Cause of error	Processing status of data updating API method#	Error code when 0300 is specified in the eads.client.compat parameter in the client properties
				client does not match the cluster information maintained by the restarted connection-target EADS server. • The maximum number of simultaneous connections to the EADS server has been exceeded.		
4000	EAD_ERROR_SERVER	InternalServerException	An unexpected internal error occurred on the EADS server.	An unexpected problem occurred on the connection-target EADS server.	U	4000
4010	EAD_ERROR_SERVER_UNSUPPORTED_REQUEST	InternalServerException	The connection-target EADS server could not process the request sent by the EADS client.	Possible causes are as follows: • The connection-target EADS server cannot process the request for a reason such as corrupted data. • The API method used is not supported by the connection-target EADS server.	N	4010
4030	EAD_ERROR_SERVER_UNAVAILABLE	InternalServerException	The connection-target EADS server process is temporarily unavailable.	The maximum number of simultaneous connections to the EADS server has been exceeded.	N	4030
4040	EAD_ERROR_SERVER_INCOMPATIBLE_CLUSTERINFO	InternalServerException	The cluster information maintained by the connection-target EADS server is not compatible with the cluster information maintained by the EADS client.	The cluster information maintained by the restarted connection-target EADS server does not match the cluster information maintained by the EADS client.	N	4040
4060	EAD_ERROR_SERVER_REPLACE_METHOD_NOT_MATCHED	InternalServerException	The value could not be stored because the stored value did not match comparativeValue during execution of replace().	The value specified for the condition in replace() of the Cache class did not match the value in cache.	N	4060
4070	EAD_ERROR_SERVER_REPLACE_METHOD_KEY_NOT_EXIST	InternalServerException	The value could not be stored because the specified key did not exist during execution of replace() (the value associated with the key did not exist).	Values could not be compared because the value associated with the key specified in replace() of the Cache class did not exist.	N	4070

Error code	Error code literal	Exception class	Nature of error	Cause of error	Processing status of data updating API method#	Error code when 0300 is specified in the eads.client.compat parameter in the client properties
4080	EAD_ERROR_SERVER_CREATE_METHOD_KEY_EXIST	InternalServerException	The value could not be stored because the key had already been stored during execution of `create()`.	The value associated with the key specified in `create()` of the `Cache` class has already been stored.	N	4080
4090	EAD_ERROR_SERVER_UPDATE_METHOD_KEY_NOT_EXIST	InternalServerException	The value could not be stored because the key was not stored during execution of `update()`.	The value associated with the key specified in `update()` of the `Cache` class has not been stored.	N	4090
4100	EAD_ERROR_SERVER_NOT_RUNNING	InternalServerException	No EADS server is available for processing requests.	Possible causes are as follows: • The EADS server that processes requests from the EADS client and the EADS servers to which data is to be copied are all isolated or stopped. • The cluster is not available.	N	4100
4110	EAD_ERROR_SERVER_STATUS	InternalServerException	The EADS server is in a status in which requests cannot be processed.	The request could not be processed due to the status of the connection-target EADS server.	N	4000
4200	EAD_ERROR_SERVER_CACHE	InternalServerException	A cache operation failed.	An operation could not be performed on a cache because a problem occurred on the connection-target EADS server. Stop the operation and check the EADS server's status.	N	4000
4210	EAD_ERROR_SERVER_CACHE_NOT_FOUND	InternalServerException	A cache operation failed because the specified cache did not exist.	An operation could not be performed on a cache because the specified cache did not exist. Stop operation on the specified cache and check the EADS server's status.	N	4000
4230	EAD_ERROR_SERVER_CACHE_CLUSTER_UPDATE	InternalServerException	A cache operation failed because the cluster configuration was changed during request processing.	An operation could not be performed on a cache because the cluster configuration was changed during request processing. Perform the cache operation again after the cluster configuration change processing has been completed.	N	4000
4300	EAD_ERROR_SERVER_CACHE_BEFORE_	InternalServerException	An internal error occurred during a cache operation, but	An internal error occurred during a cache operation on the connection-target EADS	N	4000

Error code	Error code literal	Exception class	Nature of error	Cause of error	Processing status of data updating API method#	Error code when 0300 is specified in the eads.client.compat parameter in the client properties
	REPLICATION		redundant copies of data had not been created.	server. No other normal EADS servers are affected because redundant copies of data had not been created. You can restart the same operation after the EADS server is isolated and then the connection target is changed to a normal EADS server.		
4310	EAD_ERROR_SERVER_CACHE_AFTER_REPLICATION	InternalServerException	An internal error occurred on the EADS server during cache operation and the data update operation failed.	An internal error occurred on the connection-target EADS server during cache operation. Because redundant copies of data had already been created, once the erroneous connection-target EADS server is isolated and the connection target is changed to a normal server, you can resume the operation from the status in which data had been updated.	U	4000
4700	EAD_ERROR_SERVER_FUNCTION_EXECUTE	InternalServerException	An error occurred in the user function on the EADS server.	An error occurred in the user function on the connection-target EADS server. Check the user function's processing.	--	4000
4710	EAD_ERROR_SERVER_FUNCTION_RETURN_SERIALIZE	InternalServerException	Serialization processing on the return value of the user function failed on the EADS server.	An object that is not serializable is specified for the return value of the user function executed on the connection-target EADS server.	--	4000
4720	EAD_ERROR_SERVER_FUNCTION_ARG_DESERIALIZE	InternalServerException	An object that cannot be deserialized by the EADS server is specified in the argument of the user function.	An object that cannot be deserialized by the connection-target EADS server is specified in the argument of the user function.	--	4100
4730	EAD_ERROR_SERVER_FUNCTION_NOT_FOUND	InternalServerException	No user function with the specified user function name exists on the EADS server.	No user function with the specified user function name exists on the connection-target EADS server.	--	4000
4800	EAD_ERROR_SERVER_LIMIT_EXTERNAL_MEMORY	InternalServerException	There is a shortage of memory for storing data.	The request could not be processed because the memory for storing data (explicit heap) was insufficient on the connection-target EADS server.	N	4000

Error code	Error code literal	Exception class	Nature of error	Cause of error	Processing status of data updating API method#	Error code when 0300 is specified in the eads.client.compat parameter in the client properties
4810	EAD_ERROR_SERVER_LIMIT_CACHE_FILE	InternalServerException	There is a shortage of capacity in the cache files for storing data.	The request could not be processed because the capacity of cache files for storing data was insufficient on the connection-target EADS server.	N	4000
4820	EAD_ERROR_SERVER_LIMIT_KV_COUNT	InternalServerException	The number of keys that can be stored on the EADS server has reached the upper limit.	The request could not be processed because the number of keys that can be specified on the connection-target EADS server had reached the upper limit.	N	4000
4830	EAD_ERROR_SERVER_LIMIT_KEY_VALUE_LENGTH	InternalServerException	The size of the specified key, group name, or value is greater than the maximum size permitted in the cluster.	The request could not be processed because the size of the specified key, group name, or value was greater than the maximum size permitted in the cluster.	N	4000
4999	EAD_ERROR_SERVER_UNKNOWN	InternalServerException	A nonanalyzable internal error occurred on the EADS server.	An internal error occurred on the connection-target EADS server, but the error could not be analyzed because the version of the connection-target EADS server was later than the version of the EADS client libraries.	U	4000
5000	EAD_ERROR_CLIENT	InternalClientException	An internal error occurred on the EADS client.	An unexpected error occurred in client libraries.	U	5000
5010	EAD_ERROR_CLIENT_OUT_OF_MEMORY	InternalClientException	Memory allocation in the EADS client failed.	Memory allocation failed in client libraries.	U	5010
5020	EAD_ERROR_CLIENT_BATCH_CANCEL	InternalClientException	Batch operation was cancelled.	Unperformed operations were cancelled because batch operation could not be continued.	N	5020
6000	EAD_ERROR_BATCH_FAILED_ALL	BatchOperationException	All of the batch operations failed.	An attempt was made to perform a batch operation on data by using an API method, but all operations failed.	--	6000
6010	EAD_ERROR_BATCH_FAILED_PART	BatchOperationException	Part of the batch operations failed.	An attempt was made to perform a batch operation on data by using an API method, but some of the operations failed.	--	6010

#

Indicates whether data updating had occurred when the error code was issued during execution of an API method for updating data, such as `put()` or `remove()`.

The meanings of the letters in this column are as follows:

U: Whether the data had been updated is unknown. Check whether the processing was completed.

N: The data has not been updated.

--: This error code is not issued when an API method for updating data, such as `put()` or `remove()`, is executed.

18.1.6 InitializeException class

(1) Description

This is a subclass of `CacheException` that is returned when initialization of the `CacheManager` class results in an error.

(2) Inheritance relationship

```
java.lang.Object
└java.lang.Throwable
  └java.lang.Exception
    └com.hitachi.software.xeads.client.api.CacheException
      └com.hitachi.software.xeads.client.api.InitializeException
```

(3) Format

```
public class InitializeException
extends CacheException
```

18.1.7 InternalClientException class

(1) Description

This is a subclass of `CacheException` that is returned when an internal error occurs on an EADS client.

(2) Inheritance relationship

```
java.lang.Object
└java.lang.Throwable
  └java.lang.Exception
    └com.hitachi.software.xeads.client.api.CacheException
      └com.hitachi.software.xeads.client.api.InternalClientException
```

(3) Format

```
public class InternalClientException
extends CacheException
```

18.1.8 InternalServerException class

(1) Description

This is a subclass of `CacheException` that is returned when an internal error occurs on an EADS server.

(2) Inheritance relationship

```
java.lang.Object
└java.lang.Throwable
  └java.lang.Exception
    └com.hitachi.software.xeads.client.api.CacheException
      └com.hitachi.software.xeads.client.api.InternalServerException
```

(3) Format

```
public class InternalServerException
extends CacheException
```

18.1.9 ServerCommunicationException class

(1) Description

This is a subclass of `CacheException` that is returned in the event of a communication error.

(2) Inheritance relationship

```
java.lang.Object
└java.lang.Throwable
  └java.lang.Exception
    └com.hitachi.software.xeads.client.api.CacheException
      └com.hitachi.software.xeads.client.api.ServerCommunicationException
```

(3) Format

```
public class ServerCommunicationException
extends CacheException
```

18.1.10 UserOperationException class

(1) Description

This is a subclass of `CacheException` that is returned when an error occurs due to an illegal user operation.

(2) Inheritance relationship

```
java.lang.Object
└java.lang.Throwable
  └java.lang.Exception
    └com.hitachi.software.xeads.client.api.CacheException
      └com.hitachi.software.xeads.client.api.UserOperationException
```

(3) Format

```
public class UserOperationException
extends CacheException
```

18.1.11 BatchOperationException class

(1) Description

This is a subclass of `CacheException` that is returned when part or all of a batch operation fails.

(2) Inheritance relationship

```
java.lang.Object
└java.lang.Throwable
  └java.lang.Exception
    └com.hitachi.software.xeads.client.api.CacheException
      └com.hitachi.software.xeads.client.api.BatchOperationException
```

(3) Format

```
public class BatchOperationException
extends CacheException
```

(4) List of methods

The following table lists and describes the methods provided by the `BatchOperationException` class:

Method name	Description
`getSuccessOperationNumber()`	Acquires the number of keys whose manipulation was successful during batch operation.
`getFailureOperationInfo()`	Acquires a listing of information about processing that failed during batch operation.

(5) getSuccessOperationNumber()

(a) Description

This method acquires the number of keys whose manipulation was successful during batch operation.

(b) Format

```
public int getSuccessOperationNumber()
```

(c) Return value

This method returns the number of keys whose manipulation was successful.

(6) getFailureOperationInfo()

(a) Description

This method acquires a listing of information about processing that failed during batch operation.

(b) Format

```
public java.util.Set<FailureOperationInfo> getFailureOperationInfo()
```

(c) Return value

This method returns a list the of `FailureOperationInfo` classes that store information about processing that failed.

18.1.12 AllFailureException class

(1) Description

This is a subclass of `BatchOperationException` that is returned when all of a batch operation fails.

(2) Inheritance relationship

```
java.lang.Object
└java.lang.Throwable
  └java.lang.Exception
    └com.hitachi.software.xeads.client.api.CacheException
      └com.hitachi.software.xeads.client.api.BatchOperationException
        └com.hitachi.software.xeads.client.api.AllFailureException
```

(3) Format

```
public class AllFailureException
extends BatchOperationException
```

18.1.13 PartFailureException class

(1) Description

This is a subclass of `BatchOperationException` that is returned when part of a batch operation fails.

(2) Inheritance relationship

```
java.lang.Object
└java.lang.Throwable
  └java.lang.Exception
    └com.hitachi.software.xeads.client.api.CacheException
      └com.hitachi.software.xeads.client.api.BatchOperationException
        └com.hitachi.software.xeads.client.api.PartFailureException
```

(3) Format

```
public class PartFailureException
extends BatchOperationException
```

18.2 API interfaces supported in user functions

The following table lists and describes the API interfaces supported in user functions.

Table 18–2: API interfaces supported in user functions

No.	Interface name, class name, or enumeration	Description	Package name
1	`Function`	This interface must be implemented in the user functions used on the EADS servers.	`com.hitachi.software.xeads.func.Function`
2	`FunctionContext`	This interface passes to the `Function` interface information needed to execute a user function.	`com.hitachi.software.xeads.func.FunctionContext`
3	`InitConfig`	This interface acquires information about the function properties used to initialize a user function.	`com.hitachi.software.xeads.func.InitConfig`
4	`ClientInfo`	This interface acquires information about the EADS client that executed the API method.	`com.hitachi.software.xeads.common.ClientInfo`
5	`ServerInfo`	This interface acquires information about an EADS server.	`com.hitachi.software.xeads.common.ServerInfo`
6	`ClusterInfo`	This interface acquires information about the cluster.	`com.hitachi.software.xeads.common.ClusterInfo`
7	`CacheInfo`	This interface acquires information about caches.	`com.hitachi.software.xeads.common.CacheInfo`
8	`Store`	This interface manipulates data during execution of a user function.	`com.hitachi.software.xeads.func.store.Store`
9	`Group`	This interface manipulates the group belonging to the EADS server that is executing a user function.	`com.hitachi.software.xeads.func.store.Group`
10	`Key`	This interface represents keys in the API methods that can be used in user functions.	`com.hitachi.software.xeads.func.store.Key`
11	`Value`	This interface represents values that are associated with keys and stored in the API methods that can be used in user functions.	`com.hitachi.software.xeads.func.store.Value`
12	`UserLogger`	This interface is for user logs.	`com.hitachi.software.xeads.common.UserLogger`
13	`EADsStoreException`	This exception class is returned when processing related to a data operation fails.	`com.hitachi.software.xeads.func.store.EADsStoreException`
14	`InternalServerException`	This is a subclass of `EADsStoreException` that is returned when an internal error occurs on the EADS server.	`com.hitachi.software.xeads.func.store.InternalServerException`
15	`UserOperationException`	This is a subclass of `EADsStoreException` that is returned when an error occurs due to an illegal user operation.	`com.hitachi.software.xeads.func.store.UserOperationException`

No.	Interface name, class name, or enumeration	Description	Package name
16	Enumeration `CacheType`	This enumeration represents the cache types.	`com.hitachi.software.xeads.common.CacheType`

18.2.1 Function interface

(1) Description

This interface must be implemented in the user functions used on the EADS servers.

When a user function is initialized, an instance is created. This is a single instance and remains until EADS server termination processing is performed.

(2) Interface name

```
com.hitachi.software.xeads.func.Function
```

(3) List of methods

The following table lists and describes the methods provided by the `Function` interface:

Method name	Description
`init()`	Implements user function initialization processing.
`execute()`	Implements user function processing.
`destroy()`	Implements user function termination processing.

(4) init()

(a) Description

This method implements user function initialization processing.

This method is called when a user function is initialized during startup of an EADS server.

(b) Format

```
public void init(FunctionContext context)
```

(c) Parameters

`context`

Acquires information needed for execution of user functions.

For details about the information needed for execution of user functions, see *18.2.2 FunctionContext interface*.

(5) execute()

(a) Description

This method implements user function processing.

This method is called when a user function execution request is issued from the EADS client.

(b) Format

```
public Object execute(FunctionContext context)
```

(c) Parameters

`context`

Acquires information needed for execution of user functions.
For details about the information needed for execution of user functions, see *18.2.2 FunctionContext interface*.

(d) Return value

This method returns the results of the user function processing.

The contents of `getObject()` of the `Value` interface are returned to the caller when a `Value` instance is specified for the return value.

(e) Notes

For a user function that is executed in multiple threads, implement this method to be thread-safe.

(6) destroy()

(a) Description

This method implements user function termination processing.

The method is called when a user function is terminated during termination of an EADS server.

(b) Format

```
public void destroy(FunctionContext context)
```

(c) Parameters

`context`

Acquires information needed for execution of user functions.
For details about the information needed for execution of user functions, see *18.2.2 FunctionContext interface*.

18.2.2 FunctionContext interface

(1) Description

This interface passes to the `Function` interface information needed to execute a user function.

One instance of the `FunctionContext` interface is created per request.

(2) Interface name

```
com.hitachi.software.xeads.func.FunctionContext
```

(3) List of methods

The following table lists and describes the methods provided by the `FunctionContext` interface:

Method name	Description
[Deprecated] `getServerName()`	[Deprecated] Acquires the name of the EADS server (management directory name) that is to execute the user function.
[Deprecated] `getCacheName()`	[Deprecated] Acquires the name of cache used to execute the user function.
[Deprecated] `getGroupName()`	[Deprecated] Acquires the name of the group specified on the EADS client when the user function is executed.
[Deprecated] `getArgument()`	[Deprecated] Acquires the argument specified on the EADS client when the user function is executed.
`getStore()`	Acquires an instance for manipulating data in the cache that was used for calling the user function.
`getStore()` (cache name specification)	Acquires an instance for manipulating data in the cache with a specified cache name.
`getLogger()`	Acquires a logger for outputting user logs.
`getClientInfo()`	Acquires information about the EADS client that executed the user function.
`getInitConfig()`	Acquires information about the function properties (information that was used to initialize the user function).
`getServerInfo()`	Acquires information about the EADS server that executed the user function.
`getClusterInfo()`	Acquires information about the cluster.

(4) [Deprecated] getServerName()

Reference note

This method is deprecated. Instead, use the `getName()` of the `ServerInfo` interface.

(a) Description

This method acquires the name of the EADS server (management directory name) that is to execute the user function.

(b) Format

```
public String getServerName()
```

(c) Return value

This method returns the name of the EADS server that executes the user function (the management directory name).

(5) [Deprecated] getCacheName()

Reference note

This method is deprecated. Instead, use the `getCacheName()` of the `ClientInfo` interface.

(a) Description

This method acquires the name of the cache used to execute the user function.

(b) Format

```
public String getCacheName()
```

(c) Return value

This method returns the name of the cache used to execute the user function.

If this method is called within `init()` or `destroy()` of the `Function` interface, `null` is returned.

(6) [Deprecated] getGroupName()

Reference note

This method is deprecated. Instead, use the `getKeyOrGroupName()` of the `ClientInfo` interface.

(a) Description

This method acquires the name of the group specified on the EADS client when the user function is executed.

(b) Format

```
public String getGroupName()
```

(c) Return value

This method returns the group name specified on the EADS client when the user function is executed.

It returns `null` in the following cases:

- No group was specified on the EADS client.
- This method was called within `init()` or `destroy()` of the `Function` interface.

(7) [Deprecated] getArgument()

Reference note

This method is deprecated. Instead, use the getFunctionArgument() of the ClientInfo interface.

(a) Description

This method acquires the argument specified on the EADS client when the user function is executed.

(b) Format

```
public Object getArgument()
```

(c) Return value

This method returns the argument specified on the EADS client when the user function is executed.

It returns null in the following cases:

- null was specified in the argument on the EADS client.
- This method was called within init() or destroy() of the Function interface.

(8) getStore()

(a) Description

This method acquires an instance for manipulating data in the cache that was used for calling the user function.

(b) Format

```
public Store getStore()
```

(c) Return value

This method returns an instance for manipulating data in the cache that was used for calling the user function.

For details about the instances for manipulating data, see *18.2.8 Store interface*.

If this method is called within init() or destroy() of the Function interface, null is returned.

(9) getStore() (cache name specification)

(a) Description

This method acquires an instance for manipulating data in the cache with a specified cache name.

(b) Format

```
public Store getStore(String cacheName)
                throws EADsStoreException
```

(c) Parameters

`cacheName`

Specifies the name of the cache subject to processing.

For details about the data that can be specified, see *15.2.2(4) Data types that can be specified as cache names*.

If the name of a cache that has not been created is specified, an error results.

(d) Return value

This method returns an instance for manipulating data in the cache with the specified cache name.

For details about the instances for manipulating data, see *18.2.8 Store interface*.

If this method is called within `init()` or `destroy()` of the `Function` interface, `null` is returned.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `EADsStoreException` (unexpected error)

(10) getLogger()

(a) Description

This method acquires a logger for outputting user logs.

(b) Format

```
public UserLogger getLogger()
```

(c) Return value

This method returns a logger that outputs user logs.

For details about the interface used for outputting user logs, see *18.2.12 UserLogger interface*.

(11) getClientInfo()

(a) Description

This method acquires information about the EADS client that executed the user function.

(b) Format

```
public ClientInfo getClientInfo()
```

(c) Return value

This method returns information about the EADS client that executed the user function. For details about the EADS client information, see *18.2.4 ClientInfo interface*.

If this method is called within `init()` or `destroy()` of the `Function` interface, `null` is returned.

(12) getInitConfig()

(a) Description

This method acquires information about the function properties that were used to initialize the user function.

(b) Format

```
public InitConfig getInitConfig()
```

(c) Return value

This method returns information about the function properties that were used to initialize the user function.

For details about the function properties that were used to initialize the user function, see *18.2.3 InitConfig interface*.

(13) getServerInfo()

(a) Description

This method acquires information about the EADS server that executed the user function.

(b) Format

```
public ServerInfo getServerInfo()
```

(c) Return value

This method returns information about the EADS server that executed the user function.

For information about EADS servers, see *18.2.5 ServerInfo interface*.

(14) getClusterInfo()

(a) Description

This method acquires information about the cluster.

(b) Format

```
public ClusterInfo getClusterInfo()
```

(c) Return value

This method returns information about the cluster.

For information about the cluster, see *18.2.6 ClusterInfo interface*.

If this method is called within `init()` or `destroy()` of the `Function` interface, `null` is returned.

18.2.3 InitConfig interface

(1) Description

This interface acquires information about the function properties used to initialize a user function.

One instance of the `InitConfig` interface is created for each user function when a user function is initialized.

(2) Interface name

```
com.hitachi.software.xeads.func.InitConfig
```

(3) List of methods

The following table lists and describes the methods provided by the `InitConfig` interface:

Method name	Description
`getFunctionProperty()`	Acquires the value specified for the function properties used to initialize the user function.
`getFunctionPropertyNames()`	Acquires a list of parameters in the function properties used to initialize the user function.
`getFunctionName()`	Acquires the name of the user function that is currently executing.

(4) getFunctionProperty()

(a) Description

This method acquires the value specified for the function properties used to initialize the user function.

The method acquires the verified value for the function properties, not the specified value itself.

(b) Format

```
public String getFunctionProperty(java.lang.String propName)
```

(c) Parameters

`propName`

Specifies the parameter name for the function properties.

(d) Return value

This method returns the value corresponding to the parameter name.

If no specified value corresponds to the parameter name, it returns `null`.

(e) Exceptions

- `NullPointerException` (parameter name is `null`)

(5) getFunctionPropertyNames()

(a) Description

This method acquires a list of parameters in the function properties used to initialize the user function.

(b) Format

```
public java.util.Set<String> getFunctionPropertyNames()
```

(c) Return value

This method returns a list of parameter names in the function properties used to initialize the user function.

(6) getFunctionName()

(a) Description

This method acquires the name of the user function that is currently executing.

(b) Format

```
public String getFunctionName()
```

(c) Return value

This method returns the name of the user function that is currently executing.

18.2.4 ClientInfo interface

(1) Description

This interface acquires information about the EADS client that executed the API method.

(2) Interface name

```
com.hitachi.software.xeads.common.ClientInfo
```

(3) List of methods

The following table lists and describes the methods provided by the `ClientInfo` interface:

Method name	Description
`getIp()`	Acquires the IP address of the EADS client that executed the API method.
`getPid()`	Acquires the PID assigned by the EADS client that executed the API method.
`getCacheName()`	Acquires the cache name specified on the EADS client.
`getKeyOrGroupName()`	Acquires the key or group name specified on the EADS client.

Method name	Description
getFunctionArgument()	Acquires the arguments of the user function specified on the EADS server.

(4) getIp()

(a) Description

This method acquires the IP address of the EADS client that executed the API method.

(b) Format

```
public byte[] getIp()
```

(c) Return value

This method returns the IP address of the EADS client that executed the API method.

(5) getPid()

(a) Description

This method acquires the PID assigned by the EADS client that executed the API method.

The acquired PID is the same as the value in the pid column in the log output by the EADS client that executed the API method.

(b) Format

```
public int getPid()
```

(c) Return value

This method returns the PID assigned by the EADS client that executed the API method.

(6) getCacheName()

(a) Description

This method acquires the cache name specified on the EADS client.

(b) Format

```
public String getCacheName()
```

(c) Return value

This method returns the cache name specified on the EADS client.

(7) getKeyOrGroupName()

(a) Description

This method acquires the key or group name specified on the EADS client.

(b) Format

```
public String getKeyOrGroupName()
```

(c) Return value

This method returns the key or group name specified on the EADS client.

If no key or group was specified on the EADS client, `null` is returned.

(8) getFunctionArgument()

(a) Description

This method acquires the arguments of the user function specified on the EADS server.

(b) Format

```
public Object getFunctionArgument()
```

(c) Return value

This method returns the arguments of the user function specified on the EADS server.

If no user function arguments were specified or `null` was specified on the EADS client, `null` is returned.

18.2.5 ServerInfo interface

(1) Description

This interface acquires information about an EADS server.

(2) Interface name

```
com.hitachi.software.xeads.common.ServerInfo
```

(3) List of methods

The following table lists and describes the methods provided by the `ServerInfo` interface:

Method name	Description
`getName()`	Acquires the name of the EADS server (management directory name) that is executing the user function.

Method name	Description
getAddress()	Acquires the IP address and port number of the EADS server that is executing the user function.
getId()	Acquires the EADS server ID of the EADS server that is executing the user function.
[Deprecated] getCacheNames()	[Deprecated] Acquires a list of cache names of the caches that have been created on the EADS server that is executing the user function. The acquired cache names are listed in ascending order based on their ASCII code values.
[Deprecated] getCacheType()	[Deprecated] Acquires information about the cache type from the cache with a specified cache name.
getEHeapSize()	Acquires the explicit heap size allocated for caches by the EADS server that is executing the user function.
getEHeapUsageSize()	Acquires the usage amount of the explicit heap allocated for caches by the EADS server that is executing the user function.
[Deprecated] getCacheDataFileSpecifiedSize()	[Deprecated] Acquires the size of one cache data file (value defined in the cache property file).
[Deprecated] getCacheDataFileRemainingSize()	[Deprecated] Acquires the size of the space available for storing persistent data in the cache data file currently under import processing.
[Deprecated] getCacheDataFileSpecifiedNumber()	[Deprecated] Acquires the number of cache data files (value defined in the cache property file).
[Deprecated] getCacheDataFileUnusedNumber()	[Deprecated] Acquires the number of unused cache data files that are currently available.

(4) getName()

(a) Description

This method acquires the name of the EADS server (management directory name) that is executing the user function.

(b) Format

```
public String getName()
```

(c) Return value

This method returns the name of the EADS server (management directory name) that is executing the user function.

(5) getAddress()

(a) Description

This method acquires the IP address and port number of the EADS server that is executing the user function.

(b) Format

```
public InetSocketAddress getAddress()
```

(c) Return value

This method returns a `java.net.InetSocketAddress` instance indicating the IP address and port number of a server.

(6) getId()

(a) Description

This method acquires the EADS server ID of the EADS server that is executing the user function.

(b) Format

```
public int getId()
```

(c) Return value

This method returns the EADS server ID of the EADS server that is executing the user function.

(7) [Deprecated] getCacheNames()

> **Reference note**
>
> This method is deprecated. Instead, use the `getCacheNames()` of the `ClusterInfo` interface.

(a) Description

This method acquires a list of cache names of the caches that have been created on the EADS server that is executing the user function.

The acquired cache names are listed in ascending order based on their ASCII code values.

(b) Format

```
public java.util.Set<String> getCacheNames()
```

(c) Return value

This method returns a list of cache names of the caches that have been created on the EADS server that is executing the user function.

If this method is called within `init()` or `destroy()` of the `Function` interface, `null` is returned.

(8) [Deprecated] getCacheType()

> **Reference note**
>
> This method is deprecated. Instead, use the `getType()` of the `ClusterInfo` interface.

(a) Description

This method acquires information about the cache type from the cache with a specified cache name.

(b) Format

```
public CacheType getCacheType(String cacheName)
                        throws EADsStoreException
```

(c) Parameters

`cacheName`

Specifies a cache name.

For details about the data that can be specified, see *15.2.2(4) Data types that can be specified as cache names*.

If the name of a cache that has not been created is specified, an error results.

(d) Return value

This method returns information about the cache type.

If this method is called within `init()` or `destroy()` of the `Function` interface, `null` is returned.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `EADsStoreException` (unexpected error)

(9) getEHeapSize()

(a) Description

This method acquires the explicit heap size allocated for caches by the EADS server that is executing the user function.

(b) Format

```
public long getEHeapSize()
                  throws EADsStoreException
```

(c) Return value

This method returns the explicit heap size (in bytes) allocated for caches by the EADS server.

If this method is called within `init()` or `destroy()` of the `Function` interface, `0` is returned.

(d) Exceptions

- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(10) getEHeapUsageSize()

(a) Description

This method acquires the usage amount of the explicit heap allocated for caches by the EADS server that is executing the user function.

(b) Format

```
public long getEHeapSize()
                  throws EADsStoreException
```

(c) Return value

This method returns the usage amount (in bytes) of the explicit heap allocated for caches by the EADS server.

If this method is called within `init()` or `destroy()` of the `Function` interface, `0` is returned.

(d) Exceptions

- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(11) [Deprecated] getCacheDataFileSpecifiedSize()

Reference note

This method is deprecated. Instead, use the `getCacheDataFileSize()` of the `CacheInfo` interface.

(a) Description

This method acquires the size (in bytes) of one cache data file (value defined in the cache property file).

(b) Format

```
public long getCacheDataFileSpecifiedSize(String cacheName)
                                   throws EADsStoreException
```

(c) Parameters

`cacheName`

Specifies a cache name.

For details about the data that can be specified, see *15.2.2(4) Data types that can be specified as cache names*.

If the name of a cache that has not been created is specified, an error results.

(d) Return value

This method returns the size (in bytes) of one cache data file (value defined in the cache property file).

If this method is called within `init()` or `destroy()` of the `Function` interface, `null` is returned.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `EADsStoreException` (unexpected error)

(12) [Deprecated] getCacheDataFileRemainingSize()

> **Reference note**
>
> This method is deprecated. Instead, use the `getRemainingAreaSizeOfWritingCacheDataFile()` of the `CacheInfo` interface.

(a) Description

This method acquires the size (in bytes) of the space available for storing persistent data in the cache data file currently under import processing.

(b) Format

```
public long getCacheDataFileRemainingSize(String cacheName, String key)
                                            throws EADsStoreException
```

(c) Parameters

`cacheName`

Specifies a cache name.

For details about the data that can be specified, see *15.2.2(4) Data types that can be specified as cache names*.

If the name of a cache that has not been created is specified, an error results.

`key`

Specifies a key to be associated with the value. The specification must be a group name.

For details about the data that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

Note that if the specified key does not belong to the EADS server that is executing the user function, an error results.

(d) Return value

This method returns the size (in bytes) of the space available for storing persistent data in the cache data file currently under import processing.

If this method is called within `init()` or `destroy()` of the `Function` interface, `0` is returned.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(f) Notes

- The available file size returned by this method is the information at the time the method executes. This value might change due to cache operations or compaction.

- This method acquires information about the EADS server that is executing this method. The method cannot acquire information about other EADS servers.
- If this method's return value indicates sufficient space in the cache data files but a space shortage has occurred on the EADS server to which data is to be copied, that target EADS server will be isolated when data is actually stored. Determine whether data can actually be stored taking into account the available space on the target EADS server in addition to this method's return value.

(13) [Deprecated] getCacheDataFileSpecifiedNumber()

Reference note

This method is deprecated. Instead, use the `getCacheDataFilesNumber()` of the `CacheInfo` interface.

(a) Description

This method acquires the number of cache data files (value defined in the cache property file).

(b) Format

```
public int getCacheDataFileSpecifiedNumber(String cacheName)
                                      throws EADsStoreException
```

(c) Parameters

`cacheName`

Specifies a cache name.

For details about the data that can be specified, see *15.2.2(4) Data types that can be specified as cache names*.

If the name of a cache that has not been created is specified, an error results.

(d) Return value

This method returns the number of cache data files (value defined in the cache property file).

If this method is called within `init()` or `destroy()` of the `Function` interface, `0` is returned.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `EADsStoreException` (unexpected error)

(14) [Deprecated] getCacheDataFileUnusedNumber()

Reference note

This method is deprecated. Instead, use the `getRemainingCacheDataFilesNumber()` of the `CacheInfo` interface.

(a) Description

This method acquires the number of unused cache data files that are currently available.

(b) Format

```
public int getCacheDataFileUnusedNumber(String cacheName, String key)
                                  throws EADsStoreException
```

(c) Parameters

cacheName

Specifies a cache name.

For details about the data that can be specified, see *15.2.2(4) Data types that can be specified as cache names*.

If the name of a cache that has not been created is specified, an error results.

key

Specifies a key to be associated with the value. The specification must be a group name.

For details about the data that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

Note that if the specified key does not belong to the EADS server that is executing the user function, an error results.

(d) Return value

This method returns the number of unused cache data files that are currently available. This value does not include the number of files reserved by the EADS server.

If the cache data file currently under import processing is the last available cache data file, the method returns 0.

If this method is called within init() or destroy() of the Function interface, 0 is returned.

(e) Exceptions

- UserOperationException (illegal user operation)
- InternalServerException (EADS server internal error)
- EADsStoreException (unexpected error)

(f) Notes

- The number of unused files returned by this method is the information at the time the method executes. This value might change due to cache operations or compaction.
- This method acquires information about the EADS server that is executing this method. The method cannot acquire information about other EADS servers.
- If this method's return value indicates sufficient space in the cache data files but a space shortage has occurred on the EADS server to which data is to be copied, that target EADS server will be isolated when data is actually stored. Determine whether data can actually be stored taking into account the available space on the target EADS server in addition to this method's return value.

18.2.6 ClusterInfo interface

(1) Description

This interface acquires information about the cluster.

(2) Interface name

```
com.hitachi.software.xeads.common.ClusterInfo
```

(3) List of methods

The following table lists and describes the methods provided by the `ClusterInfo` interface:

Method name	Description
`getReplicationFactor()`	Acquires the data multiplicity.
`getCacheNames()`	Acquires a list of cache names of the caches that have been created on the EADS server that is executing the user function. The acquired cache names are listed in ascending order based on their ASCII code values.
`getCacheInfo()`	Acquires information about a specified cache.
`getPosition()`	Acquires the position (hash value) that corresponds to a specified key or group name.
`getRangeId()` (position specification)	Acquires the range ID of the range that corresponds to a specified position.
`getRangeId()` (key or group name specification)	Acquires the range ID of the range that corresponds to a specified key or group name.
`getLocalRangeId()`	Acquires a list of range IDs of all ranges that are to be processed at the source or target of copy processing by the EADS server that is executing the user function.
`isLocalRange()`	Acquires a value indicating whether a specified range is the source or target of copy processing by the EADS server that is executing the user function.
`isLocalMasterRange()`	Acquires a value indicating whether a specified range is the source of copy processing by the EADS server that is executing the user function.

(4) getReplicationFactor()

(a) Description

This method acquires the data multiplicity.

(b) Format

```
public int getReplicationFactor()
```

(c) Return value

This method returns the data multiplicity

(5) getCacheNames()

(a) Description

This method acquires a list of cache names of the caches that have been created on the EADS server that is executing the user function.

The cache names are listed in ascending order based on their ASCII code values.

(b) Format

```
public String[] getCacheNames()
```

(c) Return value

This method returns a list of cache names of the caches that have been created on the EADS server that is executing the user function.

(6) getCacheInfo()

(a) Description

This method acquires information about a specified cache.

(b) Format

```
public CacheInfo getCacheInfo(String cacheName)
                       throws EADsStoreException
```

(c) Parameters

`cacheName`

Specifies a cache name.

For details about the data that can be specified, see *15.2.2(4) Data types that can be specified as cache names*.

(d) Return value

This method returns an instance that contains information about the cache with the specified cache name.

(e) Exceptions

- `UserOperationException` (illegal user operation)

(7) getPosition()

(a) Description

This method acquires the position (hash value) that corresponds to a specified key or group name.

(b) Format

```
public int getPosition(String keyOrGroupName)
                throws EADsStoreException
```

(c) Parameters

`keyOrGroupName`

Specifies a key or a group name.

For details about the data that can be specified, see *15.2.2(1) Data types that can be specified as keys* or *15.2.2(2) Data that can be specified as group names*.

(d) Return value

This method returns the position (hash value) that corresponds to the specified key or group name.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(8) getRangeId() (position specification)

(a) Description

This method acquires the range ID of the range that corresponds to a specified position.

(b) Format

```
public int getRangeId(int position)
               throws EADsStoreException
```

(c) Parameters

`position`

Specifies a position.

(d) Return value

This method returns the range ID of the range that corresponds to the specified position.

(e) Exceptions

- `EADsStoreException` (unexpected error)

(9) getRangeId() (key or group name specification)

(a) Description

This method acquires the range ID of the range that corresponds to a specified key or group name.

(b) Format

```
public int getRangeId(String keyOrGroupName)
                  throws EADsStoreException
```

(c) Parameters

`keyOrGroupName`

Specifies a key or a group name.

For details about the data that can be specified, see *15.2.2(1) Data types that can be specified as keys* or *15.2.2(2) Data that can be specified as group names*.

(d) Return value

This method returns the range ID of the range that corresponds to the specified key or group name.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(10) getLocalRangeId()

(a) Description

This method acquires a list of range IDs of all ranges that are to be processed as the source or target of copy processing by the EADS server that is executing the user function.

(b) Format

```
public int[] getLocalRangeId()
                           throws EADsStoreException
```

(c) Return value

This method returns a list of range IDs of all ranges that are to be processed as the source or target of copy processing by the EADS server that is executing the user function.

(d) Exceptions

- `EADsStoreException` (unexpected error)

(11) isLocalRange()

(a) Description

This method acquires a value indicating whether a specified range is the source or target of copy processing by the EADS server that is executing the user function.

(b) Format

```
public boolean isLocalRange(int rangeId)
                     throws EADsStoreException
```

(c) Parameters

`rangeId`

Specifies the range ID of a range.

(d) Return value

`true`

The specified range is the source or target of copy processing by the EADS server that is executing the user function.

`false`

The specified range is neither the source nor the target of copy processing by the EADS server that is executing the user function.

(e) Exceptions

- `EADsStoreException` (unexpected error)

(12) isLocalMasterRange()

(a) Description

This method acquires a value indicating whether a specified range is the source of copy processing by the EADS server that is executing the user function.

(b) Format

```
public boolean isLocalMasterRange(int rangeId)
                           throws EADsStoreException
```

(c) Parameters

`rangeId`

Specifies the range ID of a range.

(d) Return value

`true`

The specified range is the source of copy processing by the EADS server that is executing the user function.

`false`

The specified range is not the source of copy processing by the EADS server that is executing the user function.

(e) Exceptions

- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

18.2.7 CacheInfo interface

(1) Description

This interface acquires information about caches.

(2) Interface name

```
com.hitachi.software.xeads.common.CacheInfo
```

(3) List of methods

The following table lists and describes the methods provided by the CacheInfo interface:

Method name	Description
getType()	Acquires the cache types.
getCacheDataFileSize()	Acquires the size (in bytes) of a cache data file defined for each cache.
getCacheDataFilesNumber()	Acquires the number of cache data files defined for each cache.
getRemainingAreaSizeOfWritingCacheDataFile()	Acquires the remaining space (in bytes) available in the cache data file currently being imported by the EADS server that is executing the user function.
getRemainingCacheDataFilesNumber()	Acquires the number of unused cache data files currently available to the EADS server that is executing the user function.

(4) getType()

(a) Description

This method acquires the cache types.

(b) Format

```
public CacheType getType()
```

(c) Return value

This method returns the cache types.

For details about the enumeration CacheType, see *18.2.16 Enumeration CacheType*.

(5) getCacheDataFileSize()

(a) Description

This method acquires the size (in bytes) of a cache data file defined for each cache.

(b) Format

```
public long getCacheDataFileSize()
                          throws EADsStoreException
```

(c) Return value

This method returns the size of a cache data file defined for each cache.

(6) getCacheDataFilesNumber()

(a) Description

This method acquires the number of cache data files defined for each cache.

(b) Format

```
public int getCacheDataFilesNumber()
                            throws EADsStoreException
```

(c) Return value

This returns the number of cache data files defined for each cache.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `EADsStoreException` (unexpected error)

(7) getRemainingAreaSizeOfWritingCacheDataFile()

(a) Description

This method acquires the remaining space (in bytes) available in the cache data file currently being imported by the EADS server that is executing the user function.

(b) Format

```
public long getRemainingAreaSizeOfWritingCacheDataFile(int rangeId)
                                                throws EADsStoreException
```

(c) Parameters

`rangeId`

Specifies the range ID of a range that belongs to the EADS server that is executing the user function.

(d) Return value

This method returns the remaining space (in bytes) available for storing persistent data in the cache data file currently being imported.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(f) Notes

Even when this method's return value indicates sufficient space in the cache data file, the available space on other EADS servers that handle the same range might be different because the data arrangement differs from one EADS server to another.

You can expect to reduce such a difference in the available space in the cache data files by sufficiently compacting the cache data files. However, achieving exactly same amount of free space on all the EADS servers cannot be assured.

(8) getRemainingCacheDataFilesNumber()

(a) Description

This method acquires the number of unused cache data files currently available to the EADS server that is executing the user function.

(b) Format

```
public int getRemainingCacheDataFilesNumber(int rangeId)
                                     throws EADsStoreException
```

(c) Parameters

`rangeId`

Specifies the range ID of a range that belongs to the EADS server that is executing the user function.

(d) Return value

This method returns the number of unused cache data files that are currently available. This value does not include the number of files reserved by the EADS server.

If the cache data file currently under import processing is the last available cache data file, the method returns `0`.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(f) Notes

Even when this method's return value indicates that there are a sufficient number of cache data files, the number of unused files on other EADS servers that handle the same range might be different because the data arrangement differs from one EADS server to another.

You can expect to achieve roughly the same number of unused files as that of the other EADS servers by sufficiently compacting the cache data files.

18.2.8 Store interface

(1) Description

This interface manipulates data during execution of a user function.

(2) Interface name

```
com.hitachi.software.xeads.func.store.Store
```

(3) List of methods

The following table lists and describes the methods provided by the `Store` interface:

Method name	Description
`createKey()`	Creates an instance indicating a key that can be manipulated in the cache.
`createValue()`	Serializes a specified value and creates a value instance that can be manipulated in the cache.
`createGroup()`	Creates an instance indicating a group that can be manipulated in the cache.
[Deprecated] `getGroup()`	[Deprecated] Acquires an instance needed to manipulate a group specified on the EADS client when the user function is executed.
[Deprecated] `getGroup()` (group name specification)	[Deprecated] Acquires an instance needed to manipulate a specified group.
`containsKey()` (Key interface specification)	Acquires a value indicating whether the value associated with a specified key is stored in the cache.
[Deprecated] `containsKey()` (character string specification)	[Deprecated] Acquires a value indicating whether the value associated with a specified key is stored in the cache.
`put()`	Stores a value by associating it with a key.
`create()`	Stores a value by associating it with a key only when a new key is stored.
`update()`	Updates a stored value to a specified value.
`replace()`	Compares the value associated with a specified key to the value (`comparativeValue`) specified as a condition, and updates the value if the values match.
`get()`	Acquires the value associated with a specified key.
`remove()`	Deletes a specified key and the value associated with that key.
`getLastUpdateTime()`	Acquires the last time the value associated with a specified key was updated.
`getKeyCount()`	Acquires the total number of keys stored in the cache.
`getGroupCount()`	Acquires the number of groups in the highest hierarchy of all groups to which the keys stored in the cache belong.
`getGroupNames()`	Acquires a list of group names in the highest hierarchy of all groups to which the keys stored in the cache belong. The group names are listed in ascending order based on their ASCII code values.

Method name	Description
[Deprecated] getGroupNameSet()	[Deprecated] Acquires a list of group names in the highest hierarchy in the cache in ascending order based on their ASCII code values.
getEHeapUsageSize() (Key interface specification)	Acquires the size (in bytes) of the explicit heap being used to store values.
getEHeapUsageSize() (Group interface specification)	Acquires the size (in bytes) of the explicit heap being used to store values for the keys that belong to a specified group.
getDiskUsageSize() (Key interface specification)	Acquires the amount (in bytes) of disk space being used to store values.
getDiskUsageSize() (Group interface specification)	Acquires the amount (in bytes) of disk space being used to store values for keys that belong to a specified group.
calcEHeapUsageSize()	Calculates the size (in bytes) of the explicit heap used if a specified value is stored.
calcDiskUsageSize()	Calculates the amount (in bytes) of disk space used if a value is stored.

(4) createKey()

(a) Description

This method creates an instance indicating a key that can be manipulated in the cache.

(b) Format

```
public Key createKey(String key)
              throws EADsStoreException
```

(c) Parameters

key

Specifies the key that is indicated in the created instance.

For details about the data that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

If the specified key is not subject to processing by the Store interface, an error results.

(d) Return value

The method returns a Key instance.

(e) Exceptions

- UserOperationException (illegal user operation)
- InternalServerException (EADS server internal error)
- EADsStoreException (unexpected error)

(5) createValue()

(a) Description

This method serializes a specified value and creates a value instance that can be manipulated in the cache.

(b) Format

```
public Value createValue(Object value)
                  throws EADsStoreException
```

(c) Parameters

`value`

Specifies the value that is indicated in the created instance.
For details about the data that can be specified, see *15.2.2(3) Data types that can be specified as values*.
This parameter is invalid in the following cases:

- `null` is specified.
- The specified object cannot be serialized.

(d) Return value

The method returns an instance in which the `Value` interface is implemented.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `EADsStoreException` (unexpected error)

(6) createGroup()

(a) Description

This method creates an instance indicating a group that can be manipulated in the cache.

(b) Format

```
public Group createGroup(String groupName)
                  throws EADsStoreException
```

(c) Parameters

`groupName`

Specifies the group name of the group that is indicated in the created instance.
For details about the data that can be specified, see *15.2.2(2) Data that can be specified as group names*.
Note that if the specified group name is not subject to processing by the `Store` interface, an error results.

(d) Return value

This method returns the instance needed for accessing the specified group.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(7) [Deprecated] getGroup()

Reference note

This method is deprecated. Instead, use the getKeyOrGroupName() of the ClientInfo interface and createGroup() of the Store interface.

(a) Description

This method acquires an instance needed to manipulate a group specified on the EADS client when the user function is executed.

(b) Format

```
public Group getGroup()
```

(c) Return value

This method returns the instance needed to manipulate the group specified on the EADS client when the user function is executed.

If no group was specified on the EADS client, the method returns null.

(8) [Deprecated] getGroup() (group name specification)

Reference note

This method is deprecated. Instead, use the createGroup() of the Store interface.

(a) Description

This method acquires an instance needed to manipulate a specified group.

(b) Format

```
public Group getGroup(String groupName)
                 throws EADsStoreException
```

(c) Parameters

groupName

Specifies a group name.

For details about the data that can be specified, see *15.2.2(2) Data that can be specified as group names*.

This parameter is invalid in the following case:

- The specified group name is not subject to processing by the Store interface.

(d) Return value

This method returns an instance needed to manipulate the specified group.

(e) Exceptions

- UserOperationException (illegal user operation)
- InternalServerException (EADS server internal error)
- EADsStoreException(unexpected error)

(9) containsKey() (Key interface specification)

(a) Description

This method acquires a value indicating whether the value associated with a specified key is stored in the cache.

(b) Format

```
public boolean containsKey(Key key)
                        throws EADsStoreException
```

(c) Parameters

key

Specifies the key of the Key interface that is associated with the value.

This parameter is invalid in the following cases:

- null is specified.
- The specified object of the Key interface is invalid.
- The specified key is not subject to processing by the Store interface.

(d) Return value

This method returns a value indicating whether the value associated with the specified key is stored in the cache.

true

The value is stored in the cache.

false

The value is not stored in the cache.

(e) Exceptions

- UserOperationException (illegal user operation)
- InternalServerException (EADS server internal error)
- EADsStoreException (unexpected error)

(10) [Deprecated] containsKey() (character string specification)

Reference note

This method is deprecated. Instead, use the containsKey() (Key interface specification) of the Store interface.

(a) Description

This method acquires a value indicating whether the value associated with a specified key is stored in the cache.

(b) Format

```
public boolean containsKey(String key)
                    throws EADsStoreException
```

(c) Parameters

`key`

Specifies a key to be checked.

For details about the data that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

If the specified key is not subject to processing by the `Store` interface, an error results.

(d) Return value

This method returns a value indicating whether the value associated with the specified key is stored in the cache.

`true`

The value is stored in the cache.

`false`

The value is not stored in the cache.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(11) put()

(a) Description

This method stores a value by associating it with a key.

If a value is already stored, the existing value will be updated to the specified value.

(b) Format

```
public void put(Key key, Value value)
         throws EADsStoreException
```

(c) Parameters

`key`

Specifies a key of the `Key` interface that is to be associated with a value.

This parameter is invalid in the following cases:

- `null` is specified.

- The specified object of the `Key` interface is invalid.
- The specified key is not subject to processing by the `Store` interface.

`value`

Specifies a `Value` interface value to be stored.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Value` interface is invalid.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(12) create()

(a) Description

This method stores a value by associating it with a key only when a new key is stored.

If a value has already been stored, the method returns `EADsStoreException`.

(b) Format

```
public void create(Key key, Value value)
            throws EADsStoreException
```

(c) Parameters

`key`

Specifies a key of the `Key` interface that is to be associated with a value.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Key` interface is invalid.
- The specified key is not subject to processing by the `Store` interface.

`value`

Specifies a value to be stored.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Value` interface is invalid.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)

- EADsStoreException (unexpected error)

(13) update()

(a) Description

This method stores a value by associating it with a key only when the specified key is already stored (the value is updated).

If no value is stored, the method returns EADsStoreException.

(b) Format

```
public void update(Key key, Value value)
            throws EADsStoreException
```

(c) Parameters

key

Specifies a key of the Key interface that is to be associated with the value to be stored.

This parameter is invalid in the following cases:

- null is specified.
- The specified object of the Key interface is invalid.
- The specified key is not subject to processing by the Store interface.

value

Specifies a Value interface value to be stored.

This parameter is invalid in the following cases:

- null is specified.
- The specified object of the Value interface is invalid.

(d) Exceptions

- UserOperationException (illegal user operation)
- InternalServerException (EADS server internal error)
- EADsStoreException (unexpected error)

(14) replace()

(a) Description

This method compares the value associated with a specified key to the value (comparativeValue) specified as a condition, and updates the value if the values match.

The method returns EADsStoreException in the following cases:

- A value is stored, but it does not match the value (comparativeValue) specified as the condition.
- No value is stored.

(b) Format

```
public void replace(Key key, Value value, Value comparativeValue)
             throws EADsStoreException
```

(c) Parameters

`key`

Specifies the key of the `Key` interface that is associated with the value to be stored.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Key` interface is invalid.
- The specified key is not subject to processing by the `Store` interface.

`value`

Specifies a `Value` interface value to be stored.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Value` interface is invalid.

`comparativeValue`

Specifies a `Value` interface value to be compared.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Value` interface is invalid.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(15) get()

(a) Description

This method acquires the value associated with a specified key.

(b) Format

```
public Value get(Key key)
          throws EADsStoreException
```

(c) Parameters

`key`

Specifies the key of the `Key` interface that is associated with the value to be acquired.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Key` interface is invalid.
- The specified key is not subject to processing by the `Store` interface.

(d) Return value

This method returns the value associated with the specified key.

If no value is associated with the specified key, the method returns `null`.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(16) remove()

(a) Description

This method deletes a specified key and the value associated with that key.

(b) Format

```
public void remove(Key key)
            throws EADsStoreException
```

(c) Parameters

`key`

Specifies the key of the `Key` interface that is associated with the value to be deleted.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Key` interface is invalid.
- The specified key is not subject to processing by the `Store` interface.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(17) getLastUpdateTime()

(a) Description

This method acquires the last time the value associated with a specified key was updated.

(b) Format

```
public long getLastUpdateTime(Key key)
                       throws EADsStoreException
```

(c) Parameters

key

Specifies the key of the Key interface that is associated with the value.

This parameter is invalid in the following cases:

- null is specified.
- The specified object of the Key interface is invalid.
- The specified key is not subject to processing by the Store interface.

(d) Return value

This method returns the last data update time (absolute time in milliseconds from 1970-01-01 at 00:00:00 (UTC)).

(e) Exceptions

- UserOperationException (illegal user operation)
- InternalServerException (EADS server internal error)
- EADsStoreException (unexpected error)

(18) getKeyCount()

(a) Description

This method acquires the total number of keys stored in the cache.

Because keys and values have a one-to-one correspondence, the acquired number of keys corresponds to the number of values stored in the cache.

(b) Format

```
public int getKeyCount()
                throws EADsStoreException
```

(c) Return value

This method returns the total number of keys stored in the cache.

(d) Exceptions

- InternalServerException (EADS server internal error)
- EADsStoreException (unexpected error)

(19) getGroupCount()

(a) Description

This method acquires the number of groups in the highest hierarchy of all groups to which the keys stored in the cache belong.

(b) Format

```
public int getGroupCount()
                    throws EADsStoreException
```

(c) Return value

This method returns the number of groups in the highest hierarchy of all groups to which the keys stored in the cache belong.

(d) Exceptions

- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(20) getGroupNames()

(a) Description

This method acquires a list of group names in the highest hierarchy of all groups to which the keys stored in the cache belong.

The group names are listed in ascending order based on their ASCII code values.

(b) Format

```
public String[] getGroupNames()
                          throws EADsStoreException
```

(c) Return value

This method returns a list of group names in the highest hierarchy of all the groups to which the keys stored in the cache belong.

(d) Exceptions

- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(21) [Deprecated] getGroupNameSet()

Reference note

This method is deprecated. Instead, use `getGroupNames()` of the `Store` interface.

(a) Description

This method acquires a list of group names in the highest hierarchy in the cache in ascending order based on their ASCII code values.

(b) Format

```
public java.util.Set<String> getGroupNameSet()
                            throws EADsStoreException
```

(c) Return value

This method returns a list of group names in the highest hierarchy in the cache (in ascending order based on their ASCII code values).

(d) Exceptions

- `EADsStoreException`(unexpected error)

(22) getEHeapUsageSize() (Key interface specification)

(a) Description

This method acquires the size (in bytes) of the explicit heap being used to store values.

(b) Format

```
public long getEHeapUsageSize(Key key)
                       throws EADsStoreException
```

(c) Parameters

`key`

Specifies a key of the `Key` interface that is associated with the values.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Key` interface is invalid.
- The specified key is not subject to processing by the `Store` interface.

(d) Return value

This method returns the size (in bytes) of the explicit heap being used to store values.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(23) getEHeapUsageSize() (Group interface specification)

(a) Description

This method acquires the size (in bytes) of the explicit heap being used to store values for the keys that belong to a specified group.

(b) Format

```
public long getEHeapUsageSize(Group group)
                       throws EADsStoreException
```

(c) Parameters

`group`

Specifies the group name of the `Group` interface to which the key belongs.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Group` interface is invalid.
- The specified key is not subject to processing by the `Store` interface.

(d) Return value

This method returns the size (in bytes) of the explicit heap being used to store values for the keys that belong to the specified group.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(24) getDiskUsageSize() (Key interface specification)

(a) Description

This method acquires the amount (in bytes) of disk space being used to store values.

(b) Format

```
public long getDiskUsageSize(Key key)
                      throws EADsStoreException
```

(c) Parameters

`key`

Specifies a key of the `Key` interface that is associated with the values.

This parameter is invalid in the following cases:

- `null` is specified.

- The specified object of the `Key` interface is invalid.
- The specified key is not subject to processing by the `Store` interface.

(d) Return value

This method returns the amount (in bytes) of disk space being used to store values.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(25) getDiskUsageSize() (Group interface specification)

(a) Description

This method acquires the amount (in bytes) of disk space being used to store values for keys that belong to a specified group.

(b) Format

```
public long getEHeapUsageSize(Group group)
                       throws EADsStoreException
```

(c) Parameters

`group`

Specifies a group name of the `Group` interface to which keys belong.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Group` interface is invalid.
- The specified group is not subject to processing by the `Store` interface.

(d) Return value

This method returns the amount (in bytes) of disk space being used to store values for keys that belong to the specified group.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(26) calcEHeapUsageSize()

(a) Description

This method calculates the size (in bytes) of the explicit heap used if a specified value is stored.

(b) Format

```
public long calcEHeapUsageSize(Value value)
                        throws EADsStoreException
```

(c) Parameters

`value`

Specifies a `Value` interface value that was stored.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Value` interface is invalid.

(d) Return value

This method returns the size (in bytes) of the explicit heap used if the specified value is stored.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(27) calcDiskUsageSize()

(a) Description

This method calculates the amount (in bytes) of disk space used if a specified value is stored.

(b) Format

```
public long calcDiskUsageSize(Key key, Value value)
                              throws EADsStoreException
```

(c) Parameters

`key`

Specifies the key of the `Key` interface that is associated with a value.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Value` interface is invalid.

value

Specifies a Value interface value that was stored.

This parameter is invalid in the following cases:

- null is specified.
- The specified object of the Value interface is invalid.

(d) Return value

This method returns the amount of disk space used if the specified value is stored.

(e) Exceptions

- UserOperationException (illegal user operation)
- InternalServerException (EADS server internal error)
- EADsStoreException (unexpected error)

18.2.9 Group interface

(1) Description

This interface manipulates the group belonging to the EADS server that is executing a user function.

The method cannot manipulate a key that does not belong to the group specified.

(2) Interface name

```
com.hitachi.software.xeads.func.store.Group
```

(3) List of methods

The following table lists and describes the methods provided by the Group interface:

Method name	Description
toString()	Acquires the group name.
equals()	Evaluates whether a specified object is the Group instance indicating this group.
getPosition()	Acquires the position (hash group) that corresponds to the group.
createKey()	Creates a key that is to belong to this group.
createGroup()	Creates a hierarchy group immediately below the specified group.
firstKey()	Acquires the first key in order of ASCII codes of all the keys that are stored and belong to the specified group.
lastKey()	Acquires the last key in order of ASCII codes of all the keys that are stored and belong to the specified group.
keyIterator()	Acquires an iterator that accesses the keys belonging to the specified group in ascending order based on their ASCII code values.

Method name	Description
	The start position of the iterator is the first key in the group.
keyIterator() (Key interface specification)	Acquires an iterator that accesses the keys belonging to the specified group in ascending order based on their ASCII code values. The start position of the iterator is the key immediately following a specified key in ascending order based on its ASCII code value of all the keys that belong to the group.
descendingKeyIterator()	Acquires an iterator that accesses the keys belonging to the specified group in descending order based on their ASCII code values. The start position of the iterator is the last key in the group.
descendingKeyIterator() (Key interface specification)	Acquires an iterator that accesses the keys belonging to the specified group in descending order based on their ASCII code values. The start position of the iterator is the key immediately following a specified key in descending order based on its ASCII code value of all the keys that belong to the group.
[Deprecated] higherKeyIterator()	[Deprecated] Acquires an iterator that accesses the keys belonging to the specified group in ascending order based on their ASCII code values. The start position of the iterator is the key that immediately follows a specified key in ascending order based on its ASCII code value.
[Deprecated] lowerDescendingKeyIterator()	[Deprecated] Acquires an iterator that accesses the keys belonging to the specified group in descending order based on their ASCII code values. The start position of the iterator is the key that immediately follows a specified key in descending order based on its ASCII code value.
[Deprecated] put() (character string specification)	[Deprecated] Associates a value with a key, and then stores it.
[Deprecated] put() (Key interface specification)	[Deprecated] Associates a value with a key, and then stores it.
[Deprecated] create()	[Deprecated] Associates a value with a key, and then stores it only if a new key is stored.
[Deprecated] update() (character string specification)	[Deprecated] Associates a value with a key, and then stores it only if a specified key has already been stored (updates the value).
[Deprecated] update() (Key interface specification)	[Deprecated] Associates a value with a key, and then stores it only if a specified key has already been stored (updates the value).
[Deprecated] replace() (character string specification)	[Deprecated] Compares the value associated with a specified key with a value (comparativeValue) specified as a condition. Only if the values match, this method associates the value with the key, and then stores it (replaces the value).
[Deprecated] replace() (Key interface specification)	[Deprecated] Compares the value associated with a specified key with a value (comparativeValue) specified as a condition. Only if the values match, this method associates the value with the key, and then stores it (replaces the value).
[Deprecated] get() (character string specification)	[Deprecated] Acquires the value associated with a specified key.
[Deprecated] get() (Key interface specification)	[Deprecated] Acquires the value associated with a specified key.
[Deprecated] remove() (character string specification)	[Deprecated] Deletes a specified key and the value associated with that key.
[Deprecated] remove() (Key interface specification)	[Deprecated] Deletes a specified key and the value associated with that key.
[Deprecated] getLastUpdateTime() (character string specification)	[Deprecated] Acquires the last update time of the value associated with a specified key.
[Deprecated] getLastUpdateTime() (Key interface specification)	[Deprecated] Acquires the last update time of the value associated with a specified key.

Method name	Description
`getKeyCount()`	Acquires the number of keys that belong to this group.
`getGroupCount()`	Acquires the number of groups in the hierarchy immediately below this group.
`getGroupNames()`	Acquires a list of group names in the hierarchy immediately below this group. The group names are listed in ascending order based on their ASCII code values.
`getGroupLayerNames()`	Acquires a list of group hierarchy names in the hierarchy immediately below this group. The group hierarchy names are listed in ascending order based on their ASCII code values.
[Deprecated] `getGroupNameSet()`	[Deprecated] Acquires a list of group names in ascending order based on their ASCII code values.
[Deprecated] `getValueUsageSize()` (character string specification)	[Deprecated] Acquires the size of the memory usage for the value associated with a specified key.
[Deprecated] `getValueUsageSize()` (Key interface specification)	[Deprecated] Acquires the size of the memory usage for the value associated with a specified key.
[Deprecated] `getValueUsageSize()`	[Deprecated] Acquires the total size of the memory usage for all values belonging to the group including the groups in the lower hierarchies.

(4) toString()

(a) Description

This method acquires the group name.

(b) Format

```
public String toString()
```

(c) Return value

This method returns the group name.

(5) equals()

(a) Description

This method evaluates whether a specified object is the `Group` instance indicating this group.

(b) Format

```
public boolean equals(Object obj)
```

(c) Parameters

`obj`

Specifies the object to be compared.

(d) Return value

`true`

The specified object is the `Group` instance indicating this group.

`false`

The specified object is not the `Group` instance indicating this group.

(6) getPosition()

(a) Description

This method acquires the position (hash group) that corresponds to the group.

(b) Format

```
public int getPosition()
                throws EADsStoreException
```

(c) Return value

This method returns the position (hash group) that corresponds to the group.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `EADsStoreException` (unexpected error)

(7) createKey()

(a) Description

This method creates a key that is to belong to this group.

(b) Format

```
public Key createKey(String element)
              throws EADsStoreException
```

(c) Parameters

`element`

Specifies an element name to be concatenated with the group name.

This parameter is invalid in the following cases:

- A key with the element name concatenated does not satisfy the conditions explained in *15.2.2(1) Data types that can be specified as keys*.
- The name contains a colon (`:`).

(d) Return value

This method returns a `Key` instance.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `EADsStoreException` (unexpected error)

(8) createGroup()

(a) Description

This method creates a hierarchy group immediately below the specified group.

(b) Format

```
public Group createGroup(String groupLayerName)
                    throws EADsStoreException
```

(c) Parameters

`groupLayerName`

Specifies a group hierarchy name.

This parameter is invalid in the following cases:

- A group name with this group hierarchy name concatenated does not satisfy the conditions explained in *15.2.2(1) Data types that can be specified as keys*.
- The name contains a colon (`:`).

(d) Return value

This method returns the instance needed for accessing the specified group.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `EADsStoreException` (unexpected error)

(9) firstKey()

(a) Description

This method acquires the first key in order of ASCII codes of all the keys that are stored and belong to the specified group.

(b) Format

```
public Key firstKey()
               throws EADsStoreException
```

(c) Return value

This method returns the first key in order of ASCII codes of all the keys that are stored and belong to the specified group.

If no keys belong to the specified group, the method returns `null`.

(d) Exceptions

- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(10) lastKey()

(a) Description

This method acquires the last key in order of ASCII codes of all the keys that are stored and belong to the specified group.

(b) Format

```
public Key lastKey()
            throws EADsStoreException
```

(c) Return value

This method returns the last key in order of ASCII codes of all the keys that are stored and belong to the specified group.

If no keys belong to the specified group, the method returns `null`.

(d) Exceptions

- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(11) keyIterator()

(a) Description

This method acquires an iterator that accesses the keys belonging to the specified group in ascending order based on their ASCII code values.

The start position of the iterator is the first key in the group.

(b) Format

```
public java.util.Iterator<Key> keyIterator()
                                    throws EADsStoreException
```

(c) Return value

This method returns an iterator that accesses the keys belonging to the group in ascending order based on their ASCII code values.

(d) Exceptions

- `EADsStoreException` (unexpected error)

(e) Notes

Depending on the timing, the value associated with a key obtained by the iterator might have been deleted.

(12) keyIterator()

(a) Description

This method acquires an iterator that accesses the keys belonging to the specified group in ascending order based on their ASCII code values.

The start position of the iterator is the key immediately following a specified key in ascending order based on its ASCII code value of all the keys that belong to the group.

(b) Format

```
public java.util.Iterator<Key> keyIterator(Key previousKey)
                                 throws EADsStoreException
```

(c) Parameters

`previousKey`

Specifies the immediately preceding key that indicates the start position of the iterator.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Key` interface is invalid.

(d) Return value

This method returns an iterator that accesses the keys belonging to the specified group in ascending order based on their ASCII code values.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `EADsStoreException` (unexpected error)

(f) Notes

Depending on the timing, the value associated with a key obtained by the iterator might have been deleted.

(13) descendingKeyIterator()

(a) Description

This method acquires an iterator that accesses the keys belonging to the specified group in descending order based on their ASCII code values.

The start position of the iterator is the last key in the group.

(b) Format

```
public java.util.Iterator<Key> descendingKeyIterator()
                               throws EADsStoreException
```

(c) Return value

This method returns an iterator that accesses the keys belonging to the group in descending order based on their ASCII code values.

(d) Exceptions

- `EADsStoreException` (unexpected error)

(e) Notes

Depending on the timing, the value associated with a key obtained by the iterator might have been deleted.

(14) descendingKeyIterator()

(a) Description

This method acquires an iterator that accesses the keys belonging to the specified group in descending order based on their ASCII code values.

The start position of the iterator is the key immediately following a specified key in descending order based on its ASCII code value of all the keys that belong to the group.

(b) Format

```
public java.util.Iterator<Key> descendingKeyIterator(Key previousKey)
                                              throws EADsStoreException
```

(c) Parameters

`previousKey`

Specifies the immediately preceding key that indicates the start position of the iterator.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Key` interface is invalid.

(d) Return value

This method returns an iterator that accesses the keys belonging to the specified group in descending order based on their ASCII code values.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `EADsStoreException` (unexpected error)

(f) Notes

Depending on the timing, the value associated with a key obtained by the iterator might have been deleted.

(15) [Deprecated] higherKeyIterator()

> **Reference note**
>
> This method is deprecated. Instead, use `keyIterator()` (`Key` interface specification).

(a) Description

This method acquires an iterator that accesses the keys belonging to the specified group in ascending order based on their ASCII code values.

The start position of the iterator is the key that immediately follows a specified key in ascending order based on its ASCII code value.

(b) Format

```
public java.util.Iterator<Key> higherKeyIterator(java.lang.String key)
            throws EADsStoreException
```

(c) Parameters

`key`

Specifies a key.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

(d) Return value

This method returns an iterator that accesses the keys belonging to the group in ascending order based on their ASCII code values.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `EADsStoreException` (unexpected error)

(f) Notes

Depending on the timing, the value associated with a key obtained by the iterator might have been deleted.

(16) [Deprecated] lowerDescendingKeyIterator()

> **Reference note**
>
> This method is deprecated. Instead, use `descendingKeyIterator()` (`Key` interface specification).

(a) Description

This method acquires an iterator that accesses the keys belonging to the specified group in descending order based on their ASCII code values.

The start position of the iterator is the key that immediately follows a specified key in descending order based on its ASCII code value.

(b) Format

```
public java.util.Iterator<Key> lowerDescendingKeyIterator(java.lang.String
key)
                        throws EADsStoreException
```

(c) Parameters

`key`

Specifies a key.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

(d) Return value

This method returns an iterator that accesses the keys belonging to the group in descending order based on their ASCII code values.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `EADsStoreException` (unexpected error)

(f) Notes

Depending on the timing, the value associated with a key obtained by the iterator might have been deleted.

(17) [Deprecated] put() (character string specification)

Reference note

This method is deprecated. Instead, use `put()` of the `Store` interface.

(a) Description

This method associates a value with a key, and then stores it.

If a problem occurs when the value is stored, the method returns `EADsStoreException`.

(b) Format

```
public void put(java.lang.String key,
                   java.lang.Object value)
          throws EADsStoreException
```

(c) Parameters

`key`

Specifies a key to be associated with the value.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`value`

Specifies the value to be stored.

For details about the data types that can be specified, see *15.2.2(3) Data types that can be specified as values*.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(18) [Deprecated] put() (Key interface specification)

Reference note

This method is deprecated. Instead, use `put()` of the `Store` interface.

(a) Description

This method associates a value with a key, and then stores it.

If a problem occurs when the value is stored, the method returns `EADsStoreException`.

(b) Format

```
public void put(Key key,
                java.lang.Object value)
         throws EADsStoreException
```

(c) Parameters

`key`

Specifies a key of the `Key` interface that is to be associated with the value.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Key` interface is invalid.

`value`

Specifies the value to be stored.

For details about the data types that can be specified, see *15.2.2(3) Data types that can be specified as values*.

(d) Exceptions

- `UserOperationException` (illegal user operation)

- InternalServerException (EADS server internal error)
- EADsStoreException (unexpected error)

(19) [Deprecated] create()

Reference note

This method is deprecated. Instead, use create() of the Store interface.

(a) Description

This method associates a value with a key, and then stores it only if a new key is stored.

If a problem occurs when the value is stored, the method returns EADsStoreException.

(b) Format

```
public void create(java.lang.String key,
                   java.lang.Object value)
            throws EADsStoreException
```

(c) Parameters

key

Specifies a key to be associated with the value.
For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

value

Specifies the value to be stored.
For details about the data types that can be specified, see *15.2.2(3) Data types that can be specified as values*.

(d) Exceptions

- UserOperationException (illegal user operation)
- InternalServerException (EADS server internal error)
- EADsStoreException (unexpected error)

(20) [Deprecated] update() (character string specification)

Reference note

This method is deprecated. Instead, use update() of the Store interface.

(a) Description

This method associates a value with a key, and then stores it only if a specified key has already been stored (updates the value).

If a problem occurs when the value is stored, the method returns EADsStoreException.

(b) Format

```
public void update(java.lang.String key,
                   java.lang.Object value)
            throws EADsStoreException
```

(c) Parameters

`key`

Specifies a key to be associated with the value.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`value`

Specifies the value to be stored.

For details about the data types that can be specified, see *15.2.2(3) Data types that can be specified as values*.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(21) [Deprecated] update() (Key interface specification)

Reference note

This method is deprecated. Instead, use `update()` of the `Store` interface.

(a) Description

This method associates a value with a key, and then stores it only if a specified key has already been stored (updates the value).

If a problem occurs when the value is stored, the method returns `EADsStoreException`.

(b) Format

```
public void update(Key key,
                   java.lang.Object value)
            throws EADsStoreException
```

(c) Parameters

`key`

Specifies a key of the `Key` interface that is to be associated with the value.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Key` interface is invalid.

`value`

Specifies the value to be stored.

For details about the data types that can be specified, see *15.2.2(3) Data types that can be specified as values*.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(22) [Deprecated] replace() (character string specification)

Reference note

This method is deprecated. Instead, use `replace()` of the `Store` interface.

(a) Description

This method compares the value associated with a specified key with a value (`comparativeValue`) specified as a condition. Only if the values match, this method associates the value with the key, and then stores it (replaces the value).

If a problem occurs when the value is replaced, the method returns `EADsStoreException`.

(b) Format

```
public void replace(java.lang.String key,
                    java.lang.Object value,
                    java.lang.Object comparativeValue)
             throws EADsStoreException
```

(c) Parameters

`key`

Specifies a key that is associated with the value to be replaced.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`value`

Specifies the value to be stored.

For details about the data types that can be specified, see *15.2.2(3) Data types that can be specified as values*.

`comparativeValue`

Specifies the value to be compared.

For details about the data types that can be specified, see *15.2.2(3) Data types that can be specified as values*.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(23) [Deprecated] replace() (Key interface specification)

Reference note

This method is deprecated. Instead, use replace() of the Store interface.

(a) Description

This method compares the value associated with a specified key with a value (comparativeValue) specified as a condition. Only if the values match, this method associates the value with the key, and then stores it (replaces the value).

If a problem occurs when the value is replaced, the method returns EADsStoreException.

(b) Format

```
public void replace(Key key,
                    java.lang.Object value,
                    java.lang.Object comparativeValue)
             throws EADsStoreException
```

(c) Parameters

key

Specifies a key of the Key interface that is associated with the value to be replaced.

This parameter is invalid in the following cases:

- null is specified.
- The specified object of the Key interface is invalid.

value

Specifies the value to be stored.

For details about the data types that can be specified, see *15.2.2(3) Data types that can be specified as values*.

comparativeValue

Specifies the value to be compared.

For details about the data types that can be specified, see *15.2.2(3) Data types that can be specified as values*.

(d) Exceptions

- UserOperationException (illegal user operation)
- InternalServerException (EADS server internal error)
- EADsStoreException (unexpected error)

(24) [Deprecated] get() (character string specification)

Reference note

This method is deprecated. Instead, use get() of the Store interface.

(a) Description

This method acquires the value associated with a specified key.

If a problem occurs when the value is acquired, the method returns `EADsStoreException`.

(b) Format

```
public java.lang.Object get(java.lang.String key)
                              throws EADsStoreException
```

(c) Parameters

`key`

Specifies a key that is associated with the value to be acquired.
For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

(d) Return value

This method returns the value associated with the key.

If nothing is associated with the specified key, it returns `null`.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(25) [Deprecated] get() (Key interface specification)

Reference note

This method is deprecated. Instead, use `get()` of the `Store` interface.

(a) Description

This method acquires the value associated with a specified key.

If a problem occurs when the value is acquired, the method returns `EADsStoreException`.

(b) Format

```
public java.lang.Object get(Key key)
                              throws EADsStoreException
```

(c) Parameters

`key`

Specifies a key of the `Key` interface that is associated with the value to be acquired.
This parameter is invalid in the following cases:

    - null is specified.
    - The specified object of the Key interface is invalid.

(d) Return value

This method returns the value associated with the key.

If nothing is associated with the specified key, it returns null.

(e) Exceptions

- UserOperationException (illegal user operation)
- InternalServerException (EADS server internal error)
- EADsStoreException (unexpected error)

(26) [Deprecated] remove() (character string specification)

> **Reference note**
>
> This method is deprecated. Instead, use remove() of the Store interface.

(a) Description

This method deletes a specified key and the value associated with that key.

If a problem occurs when the value is deleted, the method returns EADsStoreException.

(b) Format

```
public void remove(java.lang.String key)
            throws EADsStoreException
```

(c) Parameters

key

Specifies a key that is associated with the value to be deleted.
For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

(d) Exceptions

- UserOperationException (illegal user operation)
- InternalServerException (EADS server internal error)
- EADsStoreException (unexpected error)

(27) [Deprecated] remove() (Key interface specification)

Reference note

This method is deprecated. Instead, use remove() of the Store interface.

(a) Description

This method deletes a specified key and the value associated with that key.

If a problem occurs when the value is deleted, the method returns EADsStoreException.

(b) Format

```
public void remove(Key key)
            throws EADsStoreException
```

(c) Parameters

key

Specifies a key of the Key interface that is associated with the value to be deleted.

This parameter is invalid in the following cases:

- null is specified.
- The specified object of the Key interface is invalid.

(d) Exceptions

- UserOperationException (illegal user operation)
- InternalServerException (EADS server internal error)
- EADsStoreException (unexpected error)

(28) [Deprecated] getLastUpdateTime() (character string specification)

Reference note

This method is deprecated. Instead, use getLastUpdateTime() of the Store interface.

(a) Description

This method acquires the last update time of the value associated with a specified key.

(b) Format

```
public long getLastUpdateTime(String key)
                       throws EADsStoreException
```

(c) Parameters

`key`

Specifies a key.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

(d) Return value

This method returns the last data update time (absolute time in milliseconds from 1970-01-01 at 00:00:00).

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(29) [Deprecated] getLastUpdateTime() (Key interface specification)

> **Reference note**
>
> This method is deprecated. Instead, use `getLastUpdateTime()` of the `Store` interface.

(a) Description

This method acquires the last update time of the value associated with a specified key.

(b) Format

```
public long getLastUpdateTime(Key key)
                        throws EADsStoreException
```

(c) Parameters

`key`

Specifies a key of the `Key` interface that is associated with the value.

This parameter is invalid in the following cases:

- `null` is specified.
- The specified object of the `Key` interface is invalid.

(d) Return value

This method returns the last data update time (absolute time in milliseconds from 1970-01-01 at 00:00:00).

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(30) getKeyCount()

(a) Description

This method acquires the number of keys that belong to this group.

(b) Format

```
public int getKeyCount()
                throws EADsStoreException
```

(c) Return value

This method returns the number of keys that belong to this group.

(d) Exceptions

- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(31) getGroupCount()

(a) Description

This method acquires the number of groups in the hierarchy immediately below this group.

(b) Format

```
public int getGroupCount()
                  throws EADsStoreException
```

(c) Return value

This method returns the number of groups in the hierarchy immediately below this group.

(d) Exceptions

- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(32) getGroupNames()

(a) Description

This method acquires a list of group names in the hierarchy immediately below this group.

The group names are listed in ascending order based on their ASCII code values.

(b) Format

```
public String[] getGroupNames()
                       throws EADsStoreException
```

(c) Return value

This method returns a list of group names in the hierarchy immediately below this group.

(d) Exceptions

- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(33) getGroupLayerNames()

(a) Description

This method acquires a list of group hierarchy names in the hierarchy immediately below this group.

The group hierarchy names are listed in ascending order based on their ASCII code values.

(b) Format

```
public String[] getGroupLayerNames()
                               throws EADsStoreException
```

(c) Return value

This method returns a list of group hierarchy names in the hierarchy immediately below this group.

(d) Exceptions

- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(34) [Deprecated] getGroupNameSet()

> **Reference note**
>
> This method is deprecated. Instead, use `getGroupNames()` of the `Group` interface.

(a) Description

This method acquires a list of group names in ascending order based on their ASCII code values.

(b) Format

```
public java.util.Set<String> getGroupNameSet()
                               throws EADsStoreException
```

(c) Return value

This method returns a list of group names in the hierarchy immediately below (in ascending order based on their ASCII code values).

(d) Exceptions

- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(35) [Deprecated] getValueUsageSize() (character string specification)

> **Reference note**
>
> This method is deprecated. Instead, use `getEHeapUsageSize()` (`Key` interface specification) of the `Store` interface.

(a) Description

This method acquires the size of the memory usage for the value associated with a specified key.

(b) Format

```
public long getValueUsageSize(String key)
                              throws EADsStoreException
```

(c) Parameters

`key`

Specifies a key.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

(d) Return value

This method returns the size of the memory usage (bytes) for the value associated with the specified key.

If the cache does not store values, the method returns `0`.

(e) Exceptions

- `UserOperationException` (illegal user operation)
- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

(36) [Deprecated] getValueUsageSize() (Key interface specification)

> **Reference note**
>
> This method is deprecated. Instead, use `getEHeapUsageSize()` (`Key` interface specification) of the `Store` interface.

(a) Description

This method acquires the size of the memory usage for the value associated with a specified key.

(b) Format

```
public long getValueUsageSize(Key key)
                          throws EADsStoreException
```

(c) Parameters

key

Specifies a key of the Key interface that is associated with the value.

This parameter is invalid in the following cases:

- null is specified.
- The specified object of the Key interface is invalid.

(d) Return value

This method returns the size of the memory usage (bytes) for the value associated with the specified key.

If the cache does not store values, the method returns 0.

(e) Exceptions

- UserOperationException (illegal user operation)
- InternalServerException (EADS server internal error)
- EADsStoreException (unexpected error)

(37) [Deprecated] getValueUsageSize()

> **Reference note**
>
> This method is deprecated. Instead, use getEHeapUsageSize() (Group interface specification) of the Store interface.

(a) Description

This method acquires the total size of the memory usage for all values belonging to the group including the groups in the lower hierarchies.

(b) Format

```
public long getValueUsageSize()
                          throws EADsStoreException
```

(c) Return value

This method returns the total size (bytes) of the memory usage for all values belonging to the group.

If the cache does not store values, the method returns 0.

(d) Exceptions

- `InternalServerException` (EADS server internal error)
- `EADsStoreException` (unexpected error)

18.2.10 Key interface

(1) Description

This interface represents keys in the API methods that can be used in user functions.

(2) Interface name

```
com.hitachi.software.xeads.func.store.Key
```

(3) List of methods

The following table lists and describes the methods provided by the `Key` interface:

Method name	Description
`toString()`	Acquires a key.
`equals()`	Evaluates whether a specified object is a `Key` instance indicating this key.
`getPosition()`	Acquires the position (hash value) corresponding to this key.

(4) toString()

(a) Description

This method acquires a key.

(b) Format

```
public String toString()
```

(c) Return value

This method returns a key.

(5) equals()

(a) Description

This method evaluates whether a specified object is a `Key` instance indicating this key.

(b) Format

```
public boolean equals(Object obj)
```

(c) Parameters

`obj`

Specifies the object to be compared.

(d) Return value

`true`

The specified object is a `Key` instance indicating this key.

`false`

The specified object is not a `Key` instance indicating this key.

(6) getPosition()

(a) Description

This method acquires the position (hash value) corresponding this key.

(b) Format

```
public int getPosition()
                throws EADsStoreException
```

(c) Return value

This method returns the position (hash value) corresponding to this key.

(d) Exceptions

- `UserOperationException` (illegal user operation)
- `EADsStoreException` (unexpected error)

18.2.11 Value interface

(1) Description

This interface represents values that are associated with keys and stored in the API methods that can be used in user functions.

(2) Interface name

```
com.hitachi.software.xeads.func.store.Value
```

(3) List of methods

The following table lists and describes the methods provided by the `Value` interface:

Method name	Description
toString()	Acquires a value of the String type.
equals()	Evaluates whether a specified object is a Value instance for this value.
getObject()	Acquires a value of the Object type.
size()	Acquires the size of the serialized value, which is in the internal format when a value is stored in a cache.

(4) toString()

(a) Description

This method acquires a value of the String type.

This is the same character string as toString() of the Object class for the value of the Object type that is acquired by getObject() of the Value interface.

(b) Format

```
public String toString()
```

(c) Return value

This method returns a value of the String type.

(d) Notes

If getObject() of the Value interface fails due to a deserialization error, the method returns the same value as toString() of the Object class of the Value instance.

(5) equals()

(a) Description

This method evaluates whether a specified object is a Value instance for this value.

(b) Format

```
public boolean equals(Object obj)
```

(c) Parameters

obj

Specifies the object to be compared.

(d) Return value

true

The specified object is a Value instance for this value.

false

The specified object is not a Value instance for the same value.

(e) Notes

The method determines that the objects are the same if their byte arrays obtained after serialization are the same.

(6) getObject()

(a) Description

This method acquires a value of the Object type.

(b) Format

```
public Object getObject()
                    throws EADsStoreException
```

(c) Return value

This method returns a value of the Object type.

(d) Exceptions

- UserOperationException (illegal user operation)
- EADsStoreException (unexpected error)

(7) size()

(a) Description

This method acquires the size of the serialized value, which is in the internal format when a value is stored in a cache.

(b) Format

```
public int size()
```

(c) Return value

This method returns the size (in bytes) of the serialized value, which is in the internal format when a value is stored in a cache.

(d) Notes

This method returns the size of only the area that is used by the value when the value is stored.

Because other data might also be using this area when the value is stored, you can use the following API methods if you want to determine the size of the overall area in which the value is stored:

- calcEHeapUsageSize() of the Store interface
- calcDiskUsageSize() of the Store interface

18.2.12 UserLogger interface

(1) Description

This interface is for user logs.

(2) Interface name

```
com.hitachi.software.xeads.common.UserLogger
```

(3) List of methods

The following table lists and describes the methods provided by the `UserLogger` interface:

Method name	Description
`log()` (format 1)	Outputs a message ID and a message text.
`log()` (format 2)	Outputs only a message text.
`putStackTrace()` (format 1)	Outputs a message ID and a stack trace.
`putStackTrace()` (format 2)	Outputs only a stack trace.

(4) log() (format 1)

(a) Description

This method outputs a message ID and a message text.

(b) Format

```
public void log(java.lang.String messageID,
                java.lang.String message)
```

(c) Parameters

`messageID`

Specifies a message ID.
A maximum of 21 single-byte characters can be specified.
Do not specify double-byte characters or control characters, because these characters might corrupt the format.
If `null` is specified, the null character string is output.

`message`

Specifies a message text.
Do not specify control characters, because they might corrupt the format.
If `null` is specified, the null character string is output.

(5) log() (format 2)

(a) Description

This method outputs only a message text. The method outputs the null character string for the message ID.

(b) Format

```
public void log(java.lang.String message)
```

(c) Parameters

`message`

Specifies a message text.
Do not specify control characters, because they might corrupt the format.
If `null` is specified, the null character string is output.

(6) putStackTrace() (format 1)

(a) Description

This method outputs a message ID and a stack trace.

The method outputs a detailed message and cause of a specified exception.

(b) Format

```
public void putStackTrace(java.lang.String messageID,
                          Throwable cause)
```

(c) Parameters

`messageID`

Specifies a message ID.
A maximum of 21 single-byte characters can be specified.
Do not specify double-byte characters or control characters, because these characters might corrupt the format.
If `null` is specified, the null character string is output.

`cause`

Specifies an exception object.
If `null` is specified, the null character string is output.

(7) putStackTrace() (format 2)

(a) Description

This method outputs only a stack trace. The method outputs the null character string for a message ID.

The method outputs a detailed message and cause of a specified exception.

(b) Format

```
public void putStackTrace(Throwable cause)
```

(c) Parameters

`cause`

Specifies an exception object.

If `null` is specified, the null character string is output.

18.2.13 EADsStoreException class

(1) Description

This exception class is returned when processing related to a data operation fails.

Use `getErrorCode()` to obtain an error code to determine the nature of the error.

(2) Inheritance relationship

```
java.lang.Object
└java.lang.Throwable
  └java.lang.Exception
    └com.hitachi.software.xeads.func.store.EADsStoreException
```

(3) Format

```
public class EADsStoreException
extends Exception
```

(4) List of methods

The following table lists and describes the methods provided by the `EADsStoreException` class:

Method name	Description
`getErrorCode()`	Acquires an error code for an exception that has occurred.

(5) getErrorCode()

(a) Description

This method acquires an error code for an exception that has occurred.

(b) Format

```
public int getErrorCode()
```

(c) Return value

This method returns an error code as a return value. The following table lists the error codes and describes the nature and cause of the errors:

Error code	Error code literal	Exception class	Nature of error	Cause of error	Processing status of data updating API method#
1000	EAD_ERROR_UNEXPECTED	EADsStoreException	An unexpected error occurred.	An unexpected error occurred within the program.	U
1010	EAD_ERROR_INVALID_PARAMETER	UserOperationException	A specified parameter is invalid.	An invalid parameter was specified in the API method argument.	N
1050	EAD_ERROR_NOT_SERIALIZABLE	UserOperationException	The serialization processing failed.	An object that cannot be serialized is specified in the API method argument.	N
1060	EAD_ERROR_NOT_DESERIALIZABLE	UserOperationException	The deserialization processing failed.	The object acquired from the EADS server could not be deserialized. Possible causes are as follows: • The `Class-Path` attribute of the `jar` file manifest does not contain the path of the `jar` file containing the object needed for deserialization. • The `jar` file containing the object needed for deserialization is not located under the *management-directory*`/app/lib` directory.	--
1130	EAD_ERROR_CACHE_NOT_CREATED	UserOperationException	No cache with the specified cache name has been created.	No cache with the specified cache name has been created.	--
1140	EAD_ERROR_CACHE_SETTING	UserOperationException	An API method that is not supported with the current cache settings was executed.	The executed API method attempted to acquire unavailable information from the cache.	--
4000	EAD_ERROR_SERVER	InternalServerException	An internal error that cannot be classified by the EADS server occurred.	An internal error occurred on the EADS server. Normally, this error code is not returned. It is returned when the details about the cause of the error cannot be obtained for one of the following reasons: • A deprecated earlier version of the API method was used. • The EADS server's internal error could not be classified due to an unexpected problem.	U
4060	EAD_ERROR_SERVER_REPLACE_METHOD_NOT_MATCHED	InternalServerException	The value could not be stored because the value stored during execution of `replace()` did not	The value specified in the condition in `replace()` did not match a value in the cache.	N

Error code	Error code literal	Exception class	Nature of error	Cause of error	Processing status of data updating API method#
			match `comparativeValue`.		
4070	`EAD_ERROR_SERVER_REPLACE_METHOD_KEY_NOT_EXIST`	`InternalServerException`	`replace()` was executed, but the value could not be stored because the specified key did not exist (the value associated with the key did not exist).	Values could not be compared because the value associated with the key specified in `replace()` did not exist.	N
4080	`EAD_ERROR_SERVER_CREATE_METHOD_KEY_EXIST`	`InternalServerException`	`create()` was executed, but the value could not be stored because the key had already been stored.	The value associated with the key specified in `create()` has already been stored.	N
4090	`EAD_ERROR_SERVER_UPDATE_METHOD_KEY_NOT_EXIST`	`InternalServerException`	`update()` was executed, but the value could not be stored because the stored key did not exist.	The value associated with the key specified in `update()` has not been stored.	N
4200	`EAD_ERROR_SERVER_CACHE`	`InternalServerException`	A cache operation failed.	A cache operation was disabled because a problem occurred on the EADS server that is executing the user function. Stop the operation and check the EADS server's status.	N
4220	`EAD_ERROR_SERVER_CACHE_CLUSTER_NOT_AVAILABLE`	`InternalServerException`	A cache operation failed because the cluster is not available.	A cache operation was disabled because the cluster was not available due to a problem on another EADS server or a network failure. Stop the operation and check the status of all EADS servers that make up the cluster.	N
4300	`EAD_ERROR_SERVER_CACHE_BEFORE_REPLICATION`	`InternalServerException`	An internal error occurred during a cache operation, but redundant copies of data had not been created.	During a cache operation, an internal error occurred on the EADS server that was executing the user function. No other normal EADS servers are affected because redundant copies of data had not been created. After the EADS server is isolated, you can restart the same operation on a normal EADS server.	N
4310	`EAD_ERROR_SERVER_CACHE_AFTER_REPLICATION`	`InternalServerException`	An internal error occurred on the EADS server during a cache operation and the data update operation failed.	During a cache operation, an internal error occurred on the EADS server that was executing the user function. Because redundant copies of data have already been created, after the erroneous EADS server is isolated, you can resume the operation on a normal server from the status in which data had been updated.	U
4800	`EAD_ERROR_SERVER_LIMIT_EXTERNAL_MEMORY`	`InternalServerException`	There is a shortage of memory for storing data.	The processing could not be performed because the memory for	N

Error code	Error code literal	Exception class	Nature of error	Cause of error	Processing status of data updating API method#
				storing data (explicit heap) was insufficient.	
4810	`EAD_ERROR_SERVER_LIMIT_CACHE_FILE`	`InternalServerException`	There is a shortage of capacity in the cache files for storing data.	The request could not be processed because the capacity of cache files for storing data was insufficient.	N
4820	`EAD_ERROR_SERVER_LIMIT_KV_COUNT`	`InternalServerException`	The number of keys that can be stored has reached the upper limit.	The processing could not be performed because the number of keys that can be stored had reached the upper limit.	N
4830	`EAD_ERROR_SERVER_LIMIT_KEY_VALUE_LENGTH`	`InternalServerException`	The size of the specified key, group name, or value is greater than the maximum size permitted in the cluster.	The processing could not be performed because the size of the specified key, group name, or value was greater than the maximum size permitted in the cluster.	N

#

Indicates whether data updating had occurred when an error code was issued during execution of an API method for updating data, such as `put()` or `remove()`.

The meanings of the letters in this column are as follows:

U: Whether the data was updated is unknown. Check whether the processing was completed.

N: The data has not been updated.

--: This error code is not issued when an API method for updating data, such as `put()` or `remove()`, is executed.

(d) Notes

If a deprecated API method is used, the error code might not be classified in detail.

18.2.14 InternalServerException class

(1) Description

This is a subclass of `EADsStoreException` that is returned when an internal error occurs on the EADS server.

(2) Inheritance relationship

```
java.lang.Object
└java.lang.Throwable
  └java.lang.Exception
    └com.hitachi.software.xeads.func.store.EADsStoreException
      └com.hitachi.software.xeads.func.store.InternalServerException
```

(3) Format

```
public class InternalServerException
extends EADsStoreException
```

18.2.15 UserOperationException class

(1) Description

This is a subclass of `EADsStoreException` that is returned when an error occurs due to an illegal user operation.

(2) Inheritance relationship

```
java.lang.Object
└java.lang.Throwable
  └java.lang.Exception
    └com.hitachi.software.xeads.func.store.EADsStoreException
      └com.hitachi.software.xeads.func.store.UserOperationException
```

(3) Format

```
public class UserOperationException
extends EADsStoreException
```

18.2.16 Enumeration CacheType

(1) Description

This enumeration represents the cache types. For details about the cache types, see *2.3.1 Cache types*.

(2) Enumeration name

```
com.hitachi.software.xeads.common.CacheType
```

(3) Format

```
public enum CacheType {
MEMORY,
DISK,
TWOWAY
}
```

(4) Enumeration literals

Literal	Description
MEMORY	Memory cache
DISK	Disk cache
TWOWAY	Two-way cache

19 Creating a Client Application Program (in C)

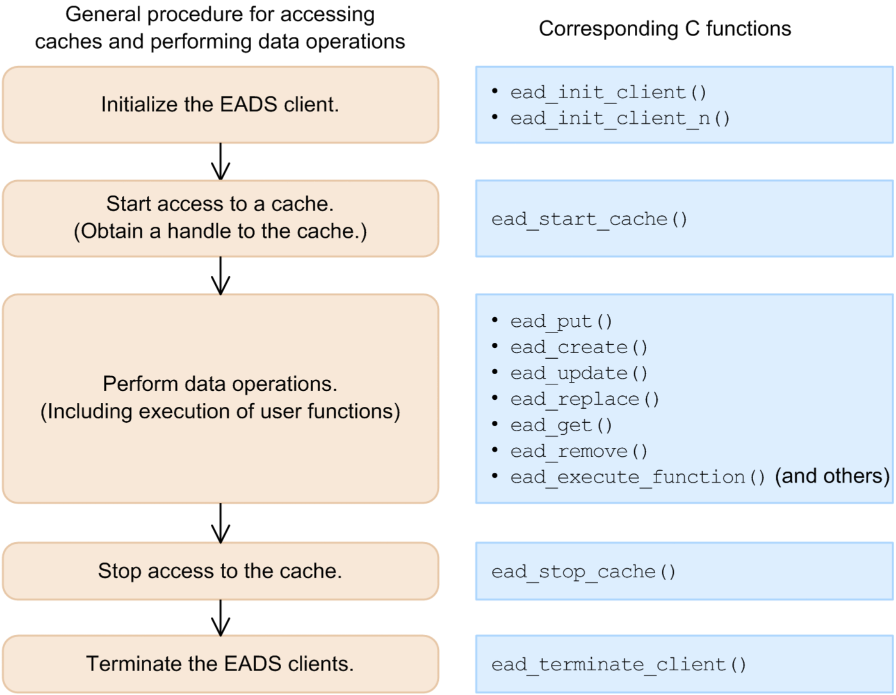
This chapter explains how to create a client application program using C.

19.1 Creating a source program (in C)

This section describes and illustrates, with reference to a sample source program, the flow of cache access and data operations.

19.1.1 Flow of cache access and data operations

The following diagram shows the flow of cache access and data operations.


General procedure for accessing caches and performing data operations
Corresponding C functions
Initialize the EADS client.
• ead_init_client()
• ead_init_client_n()
Start access to a cache. (Obtain a handle to the cache.)
ead_start_cache()
Perform data operations. (Including execution of user functions)
• ead_put()
• ead_create()
• ead_update()
• ead_replace()
• ead_get()
• ead_remove()
• ead_execute_function() (and others)
Stop access to the cache.
ead_stop_cache()
Terminate the EADS clients.
ead_terminate_client()

(1) Example of a source program in C

The following shows an example of a source program in C (to store a key and value):

```
#include <stdio.h>
#include <string.h>

#include <eads.h>

int main(int argc, char **argv) {
    int ret = 0;
    int error_code = 0;
    char CONFPATH[] = "./conf/eads_sample_client.properties";
    char CACHENAME[] = "cache1";
    char KEY[] = "key1";
    char VALUE[] = "value1";
    EAD_CACHE_MANAGER* cmp = NULL;
    EAD_CACHE *cp = NULL;
    ead_value_element value_element;

    value_element.value = (void *)VALUE;
    value_element.value_size = strlen(VALUE) + 1;
    /* Initialize EADS client */
    cmp = ead_init_client(CONFPATH, &error_code);
    printf("ead_init_client() done. (error_code = %d)\n", error_code);
    if (error_code != EAD_OK) {
        return 1;
    }

    /* Start access to the cache */
    cp = ead_start_cache(cmp, CACHENAME, &error_code);
    printf("ead_start_cache() done. (error_code = %d)\n", error_code);
    if (error_code != EAD_OK) {
        /* Need to terminate EADS client */
        goto ERR;
    }

    /* Store key and value */
    ead_put(cp, KEY, &value_element, &error_code);
    printf("ead_put() done. (error_code = %d)\n", error_code);
    if (error_code != EAD_OK) {
        /* Need to terminate EADS client */
        goto ERR;
    }

    /* Stop access to the cache */
    ead_stop_cache(cp, &error_code);
    cp = NULL;
    printf("ead_stop_cache() done. (error_code = %d)\n", error_code);
    if (error_code != EAD_OK) {
        /* Need to terminate EADS client */
        goto ERR;
    }

    /* Terminate EADS client */
    ead_terminate_client(cmp, &error_code);
    cmp = NULL;
    printf("ead_terminate_client() done. (error_code = %d)\n", error_code);
    if (error_code != EAD_OK) {
        /* Need to terminate EADS client */
        goto ERR;
    }

    return 0;

ERR:
    if(cp != NULL) {
        /* Stop access to the cache */
        ead_stop_cache(cp, &error_code);
        cp = NULL;
        printf("ead_stop_cache() done. (error_code = %d)\n", error_code);
    }

    if(cmp != NULL) {
        /* Terminate EADS client */
        ead_terminate_client(cmp, &error_code);
        cmp = NULL;
        printf("ead_terminate_client() done. (error_code = %d)\n",
error_code);
```

```
    }

    return 1;
}
```

(2) Initializing the EADS client

To initialize an EADS client, use `ead_init_client()` or `ead_init_client_n()` and obtain a handle (pointer) to the cache manager for managing the cache.

Some settings are specified in the client properties, such as the connection-target EADS server.

If you want to use multiple cache managers with different settings, such as when connection is established with multiple clusters, edit the client properties and then execute `ead_init_client()` or `ead_init_client_n()` multiple times. To terminate the EADS client when you have executed `ead_init_client()` or `ead_init_client_n()` multiple times, execute `ead_terminate_client()` on each of the acquired handles to the cache manager.

(3) Starting access to the cache

After initialization of the EADS client is complete, start access to the cache.

To start access to the cache, use `ead_start_cache()` to obtain the handle (pointer) for controlling access to the cache. Using this handle will allow you to gain access to the cache.

(4) Storing keys and values

Use `ead_put()` to store a key and value in the cache.

In `ead_put()`, specify the handle to the cache obtained from `ead_start_cache()`. In addition, specify the key and value information (the value and its size) to be stored in the cache. The value information is provided as an `ead_value_element` structure.

(5) Retrieving values

Use `ead_get()` to retrieve values from the cache.

In `ead_get()`, specify the handle to the cache obtained from `ead_start_cache()`. In addition, specify the key associated with the value you want to retrieve.

If the value is successfully retrieved by `ead_get()`, the value information associated with the key is returned as an `ead_value_element` structure.

Note that the memory area that is returned as the return value is not automatically freed. For details about freeing this memory, see *(10) Freeing a memory area returned as a return value*.

The following is an example of source code for retrieving a value.

Example of source code (for retrieving a value)

```
{
    /* Retrieve a value */
    ead_value_element ret_value;
    char KEY[] = "key1";
    ret_value = ead_get(cp, KEY, &error_code);
    printf("ead_get() done. (error_code = %d)\n", error_code);
    if (error_code != EAD_OK) {
        /* Need to terminate EADS client */
        goto ERR;
    }
    /* Free the retrieved value when done using it */
    freeValue(&ret_value);
}
```

(6) Removing keys and values

Use `ead_remove()` to remove a specified key from the cache, as well as the value associated with that key.

In `ead_remove()`, specify the handle to the cache obtained from `ead_start_cache()`. In addition, specify the key associated with the value you want to remove.

(7) Executing user functions

(a) Execution method specifying a key or a group

Use `ead_execute_function()` to execute a user function by specifying a key or a group.

In `ead_execute_function()`, specify the handle to the cache acquired by `ead_start_cache()`.

In addition, specify the name of the key or group to execute a user function and the name of the user function to be executed.

In addition, specify the user function arguments as an `ead_object` structure.

When the user function is executed by `ead_execute_function()`, the execution result of the user function is returned as an `ead_object` structure.

(b) Execution method specifying an EADS server

To execute a user function with an EADS server specified, use `ead_execute_node_function()`.

In `ead_execute_node_function()`, specify the handle to the cache obtained from `ead_start_cache()`.

Specify the EADS server that will be executing the user function as an `ead_node` structure and the arguments to be specified in the user function as an `ead_object` structure.

Also, specify the name of the user function to be executed.

When the user function is executed by `ead_execute_node_function()`, the user function execution results are returned as an `ead_object` structure.

(8) Stopping access to the cache

Use `ead_stop_cache()` to stop access to the cache.

In `ead_stop_cache()`, specify the handle to the cache obtained from `ead_start_cache()`.

(9) Terminating the EADS client

Use `ead_terminate_client()` to terminate the EADS client.

Specify in `ead_terminate_client()` the handle to the cache manager that was acquired by `ead_init_client()` or `ead_init_client_n()`.

(10) Freeing a memory area returned as a return value

The memory area that is returned as the return value of `ead_get()` is not freed automatically. Instead, you must define in the application program a function such as the one below to free the memory.

```
/* Freeing a memory area returned as a return value */
void freeValue(ead_value_element *value) {
    if (value->value != NULL) {
        free(value->value);
        value->value = NULL;
    }
    value->value_size = 0;
}
```

The table below lists the functions for which memory area returned as a return value must be freed by the application program. If a return value to be released is not `NULL`, free the memory area for the functions listed in this table when the return value is no longer needed.

Table 19–1: Functions whose memory area is to be freed when the return value is released

No.	Function name	Return value to be released
1	`ead_get_cache_name()`	`char`-type pointer returned as a cache name
2	`ead_put_all()`	`failure_info` member of the `ead_put_all_results` structure
3	`ead_get()`	`value` member of the `ead_value_element` structure
4	`ead_get_group()`	The following members of the `ead_get_group_results` structure: • `key_value_array` member • `key` member of the `ead_key_value_pair` structure in the `key_value_array` member • `value` member of the `ead_value_element` structure within the `ead_key_value_pair` structure in the `key_value_array` member
5	`ead_get_all()`	The following members of the `ead_get_all_results` structure: • `values` member • `value` member of the `ead_value_element` structure in the `values` member • `failure_info` member
6	`ead_remove_all()`	`failure_info` member of the `ead_remove_all_results` structure
7	`ead_get_group_names()`	`group_names` member of the `ead_group_names` structure
8	`ead_get_group_keys()`	`keys` member of the `ead_keys` structure
9	`ead_get_node_keys()`	
10	`ead_get_group_first_key()`	`char`-type pointer returned as a key
11	`ead_get_node_first_key()`	

No.	Function name	Return value to be released
12	`ead_get_group_next_key()`	
13	`ead_get_node_next_key()`	
14	`ead_execute_function()`	`object` member of the `ead_object` structure that is returned as user function execution results
15	`ead_execute_function_rt()`	
16	`ead_execute_node_function()`	
17	`ead_execute_node_function_rt()`	
18	`ead_get_nodelist()`	`nodes` member of the `ead_nodelist` structure
19	`ead_get_slave_nodelist()`	

19.2 Notes on creating client application programs (in C)

This section provides some points to keep in mind when creating client application programs.

19.2.1 Notes on initializing EADS clients

The following notes apply to initializing an EADS client:

- If you execute multiple EADS clients concurrently on the same machine, specify a different log output destination for each EADS client. If you specify the same log output destination, operation is not guaranteed. For details about specifying the log output destination, see *8.4.2 Specifying the file output destinations*.
- If a null character string is specified as the EADS client name in `ead_init_client_n()`, the processing is the same as for `ead_init_client()`. For details about the relationship between the EADS client name and the log output destination, see *8.4.2 Specifying the file output destinations*.
- To change a client property, perform the following steps:
  1. Terminate the EADS client with `ead_terminate_client()`.
  2. Update the client property file.
  3. Reinitialize the EADS client with `ead_init_client()` or `ead_init_client_n()`.

19.2.2 Notes on starting access to the cache

The following notes apply to starting access to the cache:

- Execute `ead_start_cache()` to start access to the cache. Execute `ead_start_cache()` as the counterpart to `ead_stop_cache()`.
- For the cache name, specify the name you created previously using the command `eztool createcache`.
- If there is no cache with that name, `EAD_ERROR_NET_CLUSTERINFO` is returned.

19.2.3 Notes on manipulating data

(1) Notes about manipulating data

The following notes apply to manipulating data:

- If the specified key is already stored in the cache, `ead_put()` simply overwrites its value. If you do not want the value updated blindly in this way, use one of the following functions instead.
  - `ead_create()`
    Stores the value with the key only when a new key is stored.
  - `ead_update()`
    Stores the value with the specified key only if the key is already stored.
  - `ead_replace()`

Compares the value associated with the specified key to the value information specified in a comparison condition, and stores the value with the key only if the values match.

- When you execute `ead_get()`, `NULL` is returned in the `value` member of the `ead_value_element` structure in any of the following circumstances:
  - The key does not exist in the cache.
  - The value information could not be retrieved.
  - The data size of the value is 0 bytes (it retrieves a byte array of length 0 that was stored using the Java language API).
- When you execute a function such as `ead_get()`, the memory area that is returned as the return value is not freed automatically. Instead, you must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.
- When you execute `ead_replace()`, if the specified key does not exist, it returns `EAD_ERROR_SERVER_REPLACE_METHOD_KEY_NOT_EXIST`.
- When you execute `ead_replace()`, if the value associated with the specified key does not match the value specified in the comparison condition, it returns `EAD_ERROR_SERVER_REPLACE_METHOD_NOT_MATCHED`.
- When you execute `ead_create()`, if the specified key already exists, it returns `EAD_ERROR_SERVER_CREATE_METHOD_KEY_EXIST`.
- When you execute `ead_update()`, if the specified key does not exist, it returns `EAD_ERROR_SERVER_UPDATE_METHOD_KEY_NOT_EXIST`.

(2) Notes about batch data operations

The following notes apply to batch data operations:

- If a specified key has already been stored in cache, `ead_put_all()` updates the value unconditionally.
- Because caches are not locked while operations are ongoing, a target value might be changed by another cache operation during a batch operation.
- When batch operations are performed on a large amount of data, a large amount of memory might be required by EADS clients and EADS servers.
- When batch operations are performed on a large amount of data, it might take a long time to complete the processing. To ensure proper operation, make sure that you design a timeout value that is appropriate for the processing time.
- A memory area returned as a return value during processing, such as by execution of `ead_put_all()`, is not freed automatically. The application program must free such a memory area. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.
- If manipulation of a key fails or the cluster configuration is changed while batch operation with multiple keys specified is underway, the batch operation will terminate and any operation that has not been performed at that point will be cancelled. Similarly, when the cluster configuration is changed during batch operation by a user operation, such as scale-out processing (adding EADS servers) or restoration processing, any operations that have not been performed are cancelled.

  Identify the key resulting in an error and determine the cause of the error from each function's return value. You can identify a key whose manipulation was cancelled based on the error code of `EAD_ERROR_CLIENT_BATCH_CANCEL`.
- When batch operation with multiple keys specified is performed and all key manipulations fail, `EAD_ERROR_BATCH_FAILED_ALL` is returned.

- When batch operation with multiple keys specified is performed but only some key manipulations fail, EAD_ERROR_BATCH_FAILED_PART is returned.

19.2.4 Notes on stopping access to the cache

The following notes apply to stopping access to the cache:

- Execute ead_stop_cache() to stop access to the cache. Execute ead_stop_cache() as the counterpart to ead_start_cache().
- Once you have stopped access to the cache using ead_stop_cache(), you must execute ead_start_cache() if you want to regain access to the cache.

19.2.5 Notes on terminating the EADS client

The following notes apply to terminating the EADS client. Be sure to observe these notes. Failure to observe the notes will cause a segmentation violation.

- Execute ead_terminate_client() after all of the API processing that uses the specified cache manager is finished.
- Always make sure that you execute ead_terminate_client() to terminate the initialized EADS client.

19.3 Compiling the source program (in C)

After you create a source program, you must compile it using the C compiler. Use the `gcc` command for the C compiler. For details on the `gcc` command, see the documentation for the C compiler.

The following headers and libraries are required for compilation:

Include path (-I)

```
/opt/hitachi/xeads/cclient/include
```

Library path (-L)

- 32-bit version
  `/opt/hitachi/xeads/cclient/lib32`
- 64-bit version
  `/opt/hitachi/xeads/cclient/lib64`

Libraries (-l)

- 32-bit version
  `hcc-4.1.1`
  `hntr2-eads-t`
- 64-bit version
  `hccx64-4.1.1`
  `hntr2-eads-te64`
- 32-bit and 64-bit versions shared
  `eads`

20 API Reference (C)

This chapter explains the application programming interface (API) for C supported by EADS.

20.1 Functions provided by the C client library

The C client library provides an EADS API that consists of the functions shown in the table below. Source programs written in the C can use these functions by including the header file provided by EADS.

Note that the functions available in the C client library (except `ead_terminate_client()`) are thread-safe.

Table 20–1: Functions available in the C client library and their header file

No.	Function name	Description	Header file
1	`ead_init_client()`	Performs initial setup of the EADS client according to the client properties.	`eads.h`
2	`ead_init_client_n()`	Initializes the EADS client according to the EADS client name and client properties.	
3	`ead_start_cache()`	Starts access to the cache and obtains a handle (pointer) for accessing the specified cache.	
4	`ead_stop_cache()`	Stops access to a specified cache.	
5	`ead_get_cache_name()`	Acquires the cache name associated with the handle to a specified cache.	
6	`ead_terminate_client()`	Terminates the EADS client.	
7	`ead_put()`	Stores a value by associating it with a key.	
8	`ead_put_array_value()`	Concatenates multiple values and then stores the values by associating them with keys.	
9	`ead_put_all()`	Using a batch operation, this function stores multiple keys and values in cache.	
10	`ead_create()`	Stores a value with a key only when a new key is stored.	
11	`ead_update()`	Stores a value with the specified key only if the key is already stored (updates the value).	
12	`ead_replace()`	Compares the value associated with a specified key to the value information specified in a comparison condition, and stores the value with the key only if the values match (replaces the value).	
13	`ead_get()`	Retrieves the value associated with a specified key.	
14	`ead_get_all()`	Using a batch operation, this function acquires the values associated with a specified list of keys.	

No.	Function name	Description	Header file
15	`ead_get_group()`	Using a batch operation, this function acquires the values associated with keys that belong to a specified group and its lower hierarchy groups.	
16	`ead_remove()`	Deletes a specified key and the value associated with that key.	
17	`ead_remove_all()`	Using a batch operation, this function deletes the values associated with a specified list of keys.	
18	`ead_remove_group()`	Using a batch operation, this function deletes the keys and values that belong to a specified group, including the keys and values that belong to lower hierarchy groups.	
19	`ead_remove_node()`	Using a batch operation, this function deletes keys and values that were copied from a specified EADS server.	
20	`ead_get_group_names()`	Acquires a list of the group names of the groups in the highest hierarchy that are stored on a specified EADS server. The group names are listed in ascending order based on their ASCII code values.	
21	`ead_get_group_keys()`	Acquires a list of keys that belong to a specified group, including keys that belong to groups under the specified group's hierarchy. The keys are listed in ascending order based on their ASCII code values.	
22	`ead_get_node_keys()`	Acquires a list of keys stored on a specified EADS server. The keys are listed in ascending order based on their ASCII code values.	
23	`ead_get_group_count()`	Acquires the number of groups in the highest hierarchy that are stored on a specified EADS server.	
24	`ead_get_group_key_count()`	Acquires the number of keys that belong to a specified group. The number of keys acquired includes the keys that belong to groups under the specified group's hierarchy.	
25	`ead_get_node_key_count()`	Acquires the number of keys that are stored on a specified EADS server.	

No.	Function name	Description	Header file
26	ead_get_group_first_key()	Acquires the first key in ascending order based on its ASCII code value from among all the keys that belong to a specified group. The keys that belong to groups under the specified group's hierarchy are also subject to this acquisition processing.	
27	ead_get_node_first_key()	Acquires the first key in ascending order based on its ASCII code value from among all the keys stored on a specified EADS server.	
28	ead_get_group_next_key()	Acquires the key that immediately follows a specified key in ascending order based on its ASCII code value from among all the keys that belong to a specified group. The keys that belong to the groups under the specified group's hierarchy are also subject to this acquisition processing.	
29	ead_get_node_next_key()	Acquires the key that immediately follows a specified key in ascending order based on its ASCII code value from among all the keys that are stored on a specified EADS server.	
30	ead_execute_function()	Uses a specified key or group to determine the EADS server on which a user function is to be executed and then executes that user function.	
31	ead_execute_function_rt()	Uses a specified key or group to determine the EADS server on which a user function is to be executed and then executes that user function. This function also sets a reception timeout value.	
32	ead_execute_node_function()	Executes a user function with an EADS server specified.	
33	ead_execute_node_function_rt()	Executes a user function with an EADS server specified and sets a reception timeout value.	
34	ead_get_nodelist()	Acquires information about the connection-target EADS servers maintained by the EADS client.	
35	[Deprecated] ead_get_node()	[Deprecated] Acquires information about the original source EADS server that stores a specified key or group.	

No.	Function name	Description	Header file
36	ead_get_slave_nodelist()	Acquires information about the original target EADS servers to which data stored on a specified EADS server is copied.	
37	ead_get_current_master_node()	Acquires information about the source EADS server that currently stores a specified key (or group).	
38	ead_get_original_master_node()	Acquires information about the original source EADS server that stores a specified key (or group).	

20.1.1 ead_init_client() (initializes the EADS client)

(1) Description

This function performs initial setup of the EADS client according to the client properties.

The EADS client name is treated as a null character string.

In addition, it obtains a handle (pointer) to the cache manager that manages the cache.

Each time ead_init_client() is executed, a thread for monitoring the cluster is generated. This thread is terminated when ead_terminate_client() is executed.

(2) Format

```
#include <eads.h>
EAD_CACHE_MANAGER *ead_init_client
(
  const char     *filename,      /* In */
  int            *error_code     /* Out */
);
```

(3) Arguments

filename

Specifies the path of the client property file for the EADS client.

error_code

Specifies the pointer from which to retrieve the error code.
For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

If ead_init_client() terminates normally, it returns a handle (pointer) to the cache manager for managing the cache.

If ead_init_client() terminates abnormally, it returns NULL.

(5) Notes

- If you execute multiple EADS clients concurrently on the same machine, specify a different log output destination for each EADS client. If you specify the same log output destination, operation is not guaranteed. For details about specifying the log output destination, see *8.4.2 Specifying the file output destinations*.
- Always make sure that you execute ead_terminate_client() to terminate an initialized EADS client.

20.1.2 ead_init_client_n() (initializes the EADS client)

(1) Description

This function initializes the EADS client according to the EADS client name and client properties.

In addition, it obtains a handle (pointer) to the cache manager that manages the cache.

Each time ead_init_client_n() is executed, a thread for monitoring the cluster is generated. This thread is terminated when ead_terminate_client() is executed.

(2) Format

```
#include <eads.h>
EAD_CACHE_MANAGER *ead_init_client_n
(
  const char     *client_name,    /* In */
  const char     *file_name,      /* In */
  int            *error_code      /* Out */
);
```

(3) Arguments

client_name

Specifies an EADS client name.

For details about the data that can be specified, see *15.2.2(5) Data that can be specified as EADS client names*.

For the relationships between EADS client names and log file output destinations, see *8.4.2 Specifying the file output destinations*.

file_name

Specifies the path of the client property file for the EADS client.

error_code

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

If ead_init_client_n() terminates normally, it returns a handle (pointer) to the cache manager that manages the cache.

If `ead_init_client_n()` terminates abnormally, it returns `NULL`.

(5) Notes

- If you execute multiple EADS clients concurrently on the same machine, specify a different log output destination for each EADS client. If you specify the same log output destination, valid operation is not guaranteed. For details about specifying the log output destination, see *8.4.2 Specifying the file output destinations*.
- Always make sure that you execute `ead_terminate_client()` to terminate an initialized EADS client.

20.1.3 ead_start_cache() (starts access to the cache)

(1) Description

This function starts access to the cache and obtains a handle (pointer) for accessing the specified cache.

(2) Format

```
#include <eads.h>
EAD_CACHE  *ead_start_cache
(
  const EAD_CACHE_MANAGER *cmp,                /* In */
  const char              *cache_name,         /* In */
  int                     *error_code          /* Out */
);
```

(3) Arguments

`cmp`

Specifies the handle (pointer) to the cache manager that is managing the cache.

Specify a handle obtained from `ead_init_client()` or `ead_init_client_n()`.

`cache_name`

Specifies the name of the cache to be accessed.

For details about the data types that can be specified, see *15.2.2(4) Data types that can be specified as cache names*.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

If `ead_start_cache()` terminates normally, it returns a handle (pointer) to the specified cache.

If `ead_start_cache()` terminates abnormally, it returns `NULL`.

(5) Notes

To terminate access to the cache, execute `ead_stop_cache()`.

20.1.4 ead_stop_cache() (stops access to the cache)

(1) Description

This function stops access to a specified cache.

(2) Format

```
#include <eads.h>
void ead_stop_cache
(
  const EAD_CACHE     *cp,            /* In */
  int                 *error_code     /* Out */
);
```

(3) Arguments

cp

Specifies the pointer to the cache to which you want to stop access.

Specify a handle obtained from ead_start_cache().

Specify the handle (pointer) obtained from ead_start_cache() when access to the cache was started.

error_code

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

20.1.5 ead_get_cache_name() (acquires cache names)

(1) Description

This function acquires the cache name associated with the handle to a specified cache.

It can also acquire the cache name of a cache to which access was terminated by ead_stop_cache().

(2) Format

```
#include <eads.h>
char* ead_get_cache_name
(
  const EAD_CACHE     *cp,            /* In */
  int                 *error_code     /* Out */
);
```

(3) Arguments

cp

Specifies the handle (pointer) to the cache whose cache name is to be acquired.

Specify the handle (pointer) obtained from ead_start_cache() when access to the cache was started.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

This function acquires the cache name associated with the handle to a specified cache.

If a problem occurs when the cache name is acquired, `NULL` is returned.

(5) Notes

The memory area for storing the cache name that is returned as the return value is not freed automatically. You must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.

20.1.6 ead_terminate_client() (terminates the EADS client)

(1) Description

This function deletes the cache manager that is managing the cache, and terminates the EADS client.

`ead_terminate_client()` frees the specified cache manager as well as the cache that was started by it. Therefore, after you have executed `ead_terminate_client()`, you can no longer execute an operation that references them. If you try, the operation is not guaranteed.

(2) Format

```
#include <eads.h>
void ead_terminate_client
(
  const EAD_CACHE_MANAGER *cmp,           /* In */
  int                     *error_code     /* Out */
);
```

(3) Arguments

`cmp`

Specifies the handle (pointer) to the cache manager that is managing the cache.

Specify a handle obtained from `ead_init_client()` or `ead_init_client_n()`.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

20.1.7 ead_put() (store a key and value)

(1) Description

Store a value by associating it with a key.

(2) Format

```
#include <eads.h>
void ead_put
(
  const EAD_CACHE             *cp,            /* In */
  const char                  *key,           /* In */
  const ead_value_element     *value,         /* In */
  int                         *error_code     /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache where the key and value are to be stored.

Specify the handle obtained from `ead_start_cache()`.

`key`

Specifies the key to associate with the value.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`value`

Specifies the value information (`ead_value_element` structure) to store.

For details about the `ead_value_element` structure, see *20.1.39 ead_value_element structure (value information)*.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

20.1.8 ead_put_array_value() (concatenates and stores multiple values)

(1) Description

This function concatenates multiple values and then stores the values by associating them with keys.

You can acquire concatenated values by retrieving the stored values.

(2) Format

```
#include <eads.h>
void ead_put_array_value
(
  const EAD_CACHE             *cp,            /* In */
  const char               * key,          /* In */
  size_t                   array_length,   /* In */
  const ead_value_element  * value_array,  /* In */
  int                      * error_code    /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache where the key and value are to be stored.

Specify the handle obtained from `ead_start_cache()`.

`key`

Specifies the key to associate with the value.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`array_length`

Specifies the number of elements in the value array to be stored.

`value_array`

Specifies a pointer to the top of the value array to be stored.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Notes

- A value of zero cannot be specified in `array_length`. If an invalid value is specified, operation is not guaranteed.
- For the array of `ead_value_element` structures, make sure that you allocate a contiguous memory area large enough for the elements.
- There is no limit to the number of elements in the value array to be stored or their sizes. For the size of concatenated data, specify a value that does not exceed the maximum size of data that can be accepted by the EADS servers.

20.1.9 ead_put_all() (stores keys and values by using a batch operation)

(1) Description

Using a batch operation, this function stores multiple keys and values in cache.

An array of the `ead_key_value_pair` structure is specified in the argument and each key is associated with a value and then is stored in cache.

If the same key is specified multiple times, it is processed multiple times in the order specified.

(2) Format

```
#include <eads.h>
ead_put_all_results ead_put_all
(
  const EAD_CACHE              *cp,              /* In */
  size_t                       array_length,     /* In */
  const ead_key_value_pair     *key_value_array, /* In */
  int                          *error_code       /* Out */
);
```

(3) Arguments

cp

Specifies the handle (pointer) to the cache where the keys and values are to be stored.

Specify the handle (pointer) obtained from ead_start_cache() when access to the cache was started.

array_length

Specifies the number of array elements of the ead_key_value_pair structure to be stored.

For details about the ead_key_value_pair structure and its format, see *20.1.40 ead_key_value_pair structure (key-value pairs)*.

key_value_array

Specifies the pointer to the top of the array of the ead_key_value_pair structure to be stored.

For details about the ead_key_value_pair structure and its format, see *20.1.40 ead_key_value_pair structure (key-value pairs)*.

error_code

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

Using a batch operation, this function returns the results (ead_put_all_results structure) of storing keys and values.

- If the batch storage processing was successful

  The success_operation_number member contains the same number of keys as in the list of keys specified in the argument, and the failure_info member contains NULL.

- If the batch storage processing failed partially or entirely

  The failure_info member contains as many sets of information about the failed processing as indicated in the failure_operation_number member.

- If an error occurred during non-key processing (such as an invalid argument or a shortage of memory area)

  A numeric-type member of the ead_put_all_results structure contains 0, and a pointer-type member contains NULL.

For details about the ead_put_all_results structure and its format, see *20.1.43 ead_put_all_results structure (execution results of ead_put_all())*.

(5) Notes

- If the batch storage processing fails, this function returns the following error codes:
  - If the batch storage processing failed entirely

    EAD_ERROR_BATCH_FAILED_ALL
  - If the batch storage processing failed partially

EAD_ERROR_BATCH_FAILED_PART

- The memory area for storing the results of batch storage of keys and values that is returned as the return value is not freed automatically. You must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.
- A value of zero cannot be specified in `array_length`. If an invalid value is specified, valid operation is not guaranteed.
- Make sure that you allocate a contiguous memory area for the array of the `ead_key_value_pair` structure that is large enough for the elements.
- If the cache operation fails partially, identify which operations have failed from the return value, and then re-execute this function, if necessary.
- Because caches are not locked while operations are ongoing, a target value might be changed by another cache operation during batch operation.
- When batch operations are performed on a large amount of data, a large amount of memory might be required by EADS clients and EADS servers.
- When batch operations are performed on a large amount of data, it might take a long time to complete the processing. To ensure proper operation, make sure that you design a timeout value that is appropriate for the processing time.

20.1.10 ead_create() (stores a new key and value)

(1) Description

This function stores a value with a key only when a new key is stored.

(2) Format

```
#include <eads.h>
void ead_create
(
  const EAD_CACHE            *cp,            /* In */
  const char                 *key,           /* In */
  const ead_value_element    *value,         /* In */
  int                        *error_code     /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache where the key and value are to be stored.

Specify the handle obtained from `ead_start_cache()`.

`key`

Specifies the key to associate with the value.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`value`

Specifies the value information (`ead_value_element` structure) to store.

For details about the `ead_value_element` structure, see *20.1.39 ead_value_element structure (value information)*.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

20.1.11 ead_update() (updates a value)

(1) Description

This function stores a value with a specified key only if the key is already stored (updates the value).

(2) Format

```
#include <eads.h>
void ead_update
(
  const EAD_CACHE              *cp,            /* In */
  const char                   *key,           /* In */
  const ead_value_element      *value,         /* In */
  int                          *error_code     /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache where the key and value are to be stored.

Specify the handle obtained from `ead_start_cache()`.

`key`

Specifies the key to associate with the value.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`value`

Specifies the value information (`ead_value_element` structure) to store.

For details about the `ead_value_element` structure, see *20.1.39 ead_value_element structure (value information)*.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

20.1.12 ead_replace() (replaces a value)

(1) Description

This function compares the value associated with a specified key to the value information specified in a comparison condition, and stores the value with the key only if the values match (replaces the value).

(2) Format

```
#include <eads.h>
void ead_replace
(
  const EAD_CACHE             *cp,                     /* In */
  const char                  *key,                    /* In */
  const ead_value_element     *value,                  /* In */
  const ead_value_element     *comparative_value,      /* In */
  int                         *error_code              /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache where the key and value are to be stored.

Specify the handle obtained from `ead_start_cache()`.

`key`

Specifies the key associated with the value to be replaced.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`value`

Specifies the value information (`ead_value_element` structure) to be stored.

For details about the `ead_value_element` structure, see *20.1.39 ead_value_element structure (value information)*.

`comparative_value`

Specifies the value information (`ead_value_element` structure) to compare.

For details about the `ead_value_element` structure, see *20.1.39 ead_value_element structure (value information)*.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

20.1.13 ead_get() (retrieves a value)

(1) Description

This function retrieves the value associated with a specified key.

(2) Format

```
#include <eads.h>
ead_value_element ead_get
(
  const EAD_CACHE     *cp,            /* In */
  const char          *key,           /* In */
  int                 *error_code     /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache holding the value to be retrieved.

Specify the handle obtained from `ead_start_cache()`.

`key`

Specifies the key associated with the value to be retrieved.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

The function returns the value information (`ead_value_element` structure) associated with the key.

For details about the `ead_value_element` structure, see *20.1.39 ead_value_element structure (value information)*.

`NULL` is set in the `value` member of the `ead_value_element` structure in the following cases:

- The key does not exist in the cache.
- A problem occurred when the value information was acquired.
- The data size of the value is 0 bytes (it retrieves a byte array of length 0 that was stored using the Java language API).

(5) Notes

The memory area of the value information that is returned as the return value is not freed automatically. Instead, you must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.

20.1.14 ead_get_all() (acquires values by using a batch operation)

(1) Description

Using a batch operation, this function acquires the values associated with a specified list of keys.

If the same key is specified multiple times, it is processed multiple times in the order specified in the list of keys.

(2) Format

```
#include <eads.h>
ead_get_all_results ead_get_all
(
  const EAD_CACHE      *cp,           /* In */
  const ead_keys       *keys,         /* In */
  int                  *error_code    /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache where the values to be acquired are stored.

Specify the handle (pointer) obtained from `ead_start_cache()` when access to the cache was started.

`keys`

Specifies a list of keys (`ead_keys` structure) that are associated with the values to be acquired.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

For details about the `ead_keys` structure and its format, see *20.1.41 ead_keys structure (multiple keys)*.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

Using a batch operation, this function returns the results (`ead_get_all_results` structure) of acquiring the values associated with the specified list of keys.

- If the batch acquisition processing was successful

  The `values` member contains the values that correspond to the list of keys specified in the argument.

- If the batch acquisition processing failed partially

  The `failure_info` member contains as many sets of information about failed processing as indicated in the `failure_operation_number` member.

  At the position of a `values` member that corresponds to a key whose acquisition failed, an `ead_value_element` structure whose value is `NULL` is returned.

- If the batch acquisition processing failed entirely

  The `values_length` member contains `0`, and the `values` member contains `NULL`.

- If an error occurred during non-key processing (such as an invalid argument or a shortage of memory area)

  A numeric-type member of the `ead_get_all_results` structure contains `0`, and a pointer-type member contains `NULL`.

For details about the `ead_get_all_results` structure and its format, see *20.1.44 ead_get_all_results structure (execution results of ead_get_all())*.

(5) Notes

- If the batch value acquisition processing fails, this function returns the following error codes:

    - If the batch acquisition processing failed entirely
      `EAD_ERROR_BATCH_FAILED_ALL`
    - If the batch acquisition processing failed partially
      `EAD_ERROR_BATCH_FAILED_PART`
- The memory area for storing the results of batch acquisition of values that is returned as the return value is not freed automatically. You must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.
- If the cache operation fails partially, identify which operations have failed from the return value, and then re-execute this function, if necessary.
- Because caches are not locked while operations are ongoing, a target value might be changed by another cache operation during batch operation.
- When batch operations are performed on a large amount of data, a large amount of memory might be required by EADS clients and EADS servers.
- When batch operations are performed on a large amount of data, it might take a long time to complete the processing. To ensure proper operation, make sure that you design a timeout value that is appropriate for the processing time.

20.1.15 ead_get_group() (acquires values by using a batch operation with group specification)

(1) Description

Using a batch operation, this function acquires the values associated with keys that belong to a specified group and its lower hierarchy groups.

(2) Format

```
#include <eads.h>
ead_get_group_results ead_get_group
(
  const EAD_CACHE      *cp,            /* In */
  const char           *group_name,    /* In */
  int                  *error_code     /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache where the values to be acquired are stored.

Specify the handle (pointer) obtained from `ead_start_cache()` when access to the cache was started.

`group_name`

Specifies the group name of the group for which values are to be acquired.

For details about the data that can be specified, see *15.2.2(2) Data that can be specified as group names*.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

Using a batch operation, this function returns the results (ead_get_group_results structure) of acquiring the values associated with keys that belong to the specified group and its lower hierarchy groups.

- If the batch acquisition processing was partially or entirely successful

  The key_value_array contains as many keys as there are values successfully acquired and the values that correspond to the list of keys specified in the argument.
- If the batch acquisition processing failed entirely

  A numeric-type member of the ead_get_group_results structure contains 0, and a pointer-type member contains NULL.

For details about the ead_get_group_results structure and its format, see *20.1.45 ead_get_group_results structure (execution results of ead_get_group())*.

(5) Notes

- The memory area for storing the results of batch acquisition of values that is returned as the return value is not freed automatically. You must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.
- If the cache operation fails partially, check the cache operation results, and then re-execute the function, if necessary.
- Because caches are not locked while operations are ongoing, a target value might be changed by another cache operation during batch operation.
- When batch operations are performed on a large amount of data, a large amount of memory might be required by EADS clients and EADS servers.
- When batch operations are performed on a large amount of data, it might take a long time to complete the processing. To ensure proper operation, make sure that you design a timeout value that is appropriate for the processing time.

20.1.16 ead_remove() (deletes a value)

(1) Description

This function deletes a specified key and the value associated with that key.

(2) Format

```
#include <eads.h>
void ead_remove
(
  const EAD_CACHE     *cp,            /* In */
  const char          *key,           /* In */
  int                 *error_code     /* Out */
);
```

(3) Arguments

cp

Specifies the handle (pointer) to the cache holding the key and value to be deleted.

Specify the handle obtained from `ead_start_cache()`.

`key`

Specifies the key associated with the value to be deleted.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

20.1.17 ead_remove_all() (deletes values by using a batch operation)

(1) Description

Using a batch operation, this function deletes the values associated with a specified list of keys.

If the same key is specified multiple times, it is processed multiple times in the order specified in the list of keys.

(2) Format

```
#include <eads.h>
ead_remove_all_results ead_remove_all
(
  const EAD_CACHE     *cp,            /* In */
  const ead_keys      *keys,          /* In */
  int                 *error_code     /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache where the values to be deleted are stored.

Specify the handle (pointer) obtained from `ead_start_cache()` when access to the cache was started.

`keys`

Specifies the list of keys (`ead_keys` structure) to be deleted.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

For details about the `ead_keys` structure and its format, see *20.1.41 ead_keys structure (multiple keys)*.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

Using a batch operation, this function returns the results (`ead_remove_all_results` structure) of deleting the values associated with the specified list of keys.

- If the batch deletion processing was entirely successful

The value of the success_operation_number member is the same as the number of keys specified in the argument, and the value of the failure_info member is NULL.

- If the batch deletion processing failed partially or entirely

  The failure_info member contains as many sets of information about failed processing as indicated in the failure_operation_number member.

- If an error occurred during non-key processing (such as an invalid argument or a shortage of memory area)

  A numeric-type member of the ead_remove_all_results structure contains 0, and a pointer-type member contains NULL.

For details about the ead_remove_all_results structure and its format, see *20.1.46 ead_remove_all_results structure (execution results of ead_remove_all())*.

(5) Notes

- If the batch value deletion processing fails, this function returns the following error codes:
  - If the batch deletion processing failed entirely

    EAD_ERROR_BATCH_FAILED_ALL
  - If the batch deletion processing failed partially

    EAD_ERROR_BATCH_FAILED_PART
- The memory area for storing the results of batch deletion of values that is returned as the return value is not freed automatically. You must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.
- If the deletion processing fails, data to be deleted might still remain. Therefore, determine the cause of the error based on the error code and identify which operations have failed. If necessary, re-execute ead_remove_all() (deletes values by using a batch operation).
- Because caches are not locked while operations are ongoing, a target value might be changed by another cache operation during batch operation.
- When batch operations are performed on a large amount of data, a large amount of memory might be required by EADS clients and EADS servers.
- When batch operations are performed on a large amount of data, it might take a long time to complete the processing. To ensure proper operation, make sure that you design a timeout value that is appropriate for the processing time.

20.1.18 ead_remove_group() (deletes values by using a batch operation with group specification)

(1) Description

Using a batch operation, this function deletes the keys and values belonging to a specified group, including the keys and values that belong to the specified group's lower hierarchy groups.

If the deletion processing fails for some reason during batch deletion of keys and values, the function returns an error code that indicates the cause of the failure.

(2) Format

```
#include <eads.h>
void ead_remove_group
(
  const EAD_CACHE     *cp,            /* In */
  const char          *group_name,    /* In */
  int                 *error_code     /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache where the values to be deleted are stored.

Specify the handle (pointer) obtained from `ead_start_cache()` when access to the cache was started.

`group_name`

Specifies the group name of the group to which the keys and values that are to be deleted belong.

For details about the data that can be specified, see *15.2.2(2) Data that can be specified as group names*.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Notes

- If the deletion processing fails, data to be deleted might still remain. Therefore, check the execution results and take appropriate action. If necessary, re-execute `ead_remove_group()` (deletes values by using a batch operation with group specification).
- If the cache operation fails partially, identify which operations failed from the return value, and then re-execute this function, if necessary.
- Because caches are not locked while operations are ongoing, a target value might be changed by another cache operation during batch operation.
- When batch operations are performed on a large amount of data, a large amount of memory might be required by EADS clients and EADS servers.
- When batch operations are performed on a large amount of data, it might take a long time to complete the processing. To ensure proper operation, make sure that you design a timeout value that is appropriate for the processing time.

20.1.19 ead_remove_node() (deletes values by using a batch operation with EADS server specification)

(1) Description

Using a batch operation, this function deletes keys and values that were copied from a specified EADS server.

If the deletion processing fails for some reason during batch deletion of keys and values, the function returns an error code that indicates the cause of the failure.

(2) Format

```
#include <eads.h>
void ead_remove_node
(
  const EAD_CACHE     *cp,              /* In */
  const ead_node      *target_node,     /* In */
  int                 *error_code       /* Out */
);
```

(3) Arguments

cp

Specifies the handle (pointer) to the cache where the values to be deleted are stored.

Specify the handle (pointer) obtained from ead_start_cache() when access to the cache was started.

target_node

Specifies a pointer to the EADS server (ead_node structure) on which the batch deletion processing is to be executed.

You can only specify the pointer of the ead_node structure obtained by using the EADS client library. If you specify any other pointer, correct operation is not guaranteed.

For the format and details of the ead_node structure, see *20.1.50 ead_node structure (object used in a user function with an EADS server specified)*.

An error results if the address information (IP address and port number) managed by the specified ead_node structure does not match the address information of any EADS servers maintained by the EADS client.

error_code

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Notes

- If deletion processing fails, data to be deleted might still remain. Check the execution results and take the appropriate action. If necessary, re-execute ead_remove_node() (deletes values by using a batch operation with EADS server specification).
- Because caches are not locked while operations are ongoing, a target value might be changed by another cache operation during batch operation.
- When batch operations are performed on a large amount of data, a large amount of memory might be required by EADS clients and EADS servers.
- When batch operations are performed on a large amount of data, it might take a long time to complete the processing. To ensure proper operation, make sure that you design a timeout value that is appropriate for the processing time.

20.1.20 ead_get_group_names() (acquires a list of group names in the highest hierarchy)

(1) Description

This function acquires a list of the group names of the groups in the highest hierarchy that are stored on a specified EADS server.

The group names are listed in ascending order based on their ASCII code values.

(2) Format

```
#include <eads.h>
ead_group_names ead_get_group_names
(
  const EAD_CACHE     *cp,            /* In */
  const ead_node      *target_node,   /* In */
  int                 *error_code     /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache in which the list of group names is to be acquired.

Specify the handle (pointer) obtained from `ead_start_cache()` when access to the cache was started.

`target_node`

Specifies a pointer to the EADS server (`ead_node` structure) from which the list of group names is to be acquired.

You can only specify the pointer of the `ead_node` structure obtained by using the EADS client library. If you specify any other pointer, correct operation is not guaranteed.

For the format and details of the `ead_node` structure, see *20.1.50 ead_node structure (object used in a user function with an EADS server specified)*.

An error results if the address information (IP address and port number) managed by the specified `ead_node` structure does not match the address information of any EADS servers maintained by the EADS client.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

This function returns a list of the group names of the groups in the highest hierarchy (`ead_group_names` structure) that are stored on the specified EADS server.

`NULL` is set in the `group_names` member of the `ead_group_names` structure in the following cases:

- The specified EADS server does not contain any keys that belong to the group.
- Acquisition of a list of group names failed due to an error.

For details about the `ead_group_names` structure and its format, see *20.1.42 ead_group_names structure (multiple group names)*.

(5) Notes

- If the group_names member of the ead_group_names structure that is returned as the return value is not NULL, the memory area is not freed automatically. You must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.
- As the number of groups on the specified EADS server increases, the time and the amount of resources required for acquisition processing might increase.

20.1.21 ead_get_group_keys() (acquires a list of keys with group specification)

(1) Description

This function acquires a list of keys that belong to a specified group, including keys that belong to groups under the specified group's hierarchy.

The keys are listed in ascending order based on their ASCII code values.

(2) Format

```
#include <eads.h>
ead_keys ead_get_group_keys
(
  const EAD_CACHE     *cp,              /* In */
  const char          *group_name,      /* In */
  int                 *error_code       /* Out */
);
```

(3) Arguments

cp

Specifies the handle (pointer) to the cache in which the list of keys is to be acquired.

Specify the handle (pointer) obtained from ead_start_cache() when access to the cache was started.

group_name

Specifies a group name.

For details about the data that can be specified, see *15.2.2(2) Data that can be specified as group names*.

error_code

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

This function returns a list of the keys (ead_keys structure) that belong to the specified group in ascending order based on their ASCII code values.

NULL is set in the keys member of the ead_keys structure in the following cases:

- No keys belong to the specified group.

- Acquisition of a list of keys failed due to an error.

For details about the ead_keys structure and its format, see *20.1.41 ead_keys structure (multiple keys)*.

(5) Notes

- If the keys member of the ead_keys structure that is returned as the return value is not NULL, the memory area is not freed automatically. You must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.
- As the number of keys on the specified EADS server increases, the time and the amount of resources required for the acquisition processing might increase.

20.1.22 ead_get_node_keys() (acquires a list of keys with EADS server specification)

(1) Description

This function acquires a list of the keys stored on a specified EADS server.

The keys are listed in ascending order based on their ASCII code values.

(2) Format

```
#include <eads.h>
ead_keys ead_get_node_keys
(
  const EAD_CACHE     *cp,              /* In */
  const ead_node      *target_node,     /* In */
  int                 *error_code       /* Out */
);
```

(3) Arguments

cp

Specifies the handle (pointer) to the cache in which the list of keys is to be acquired.

Specify the handle (pointer) obtained from ead_start_cache() when access to the cache was started.

target_node

Specifies the pointer to the EADS server (ead_node structure) from which the list of keys is to be acquired.

You can only specify the pointer of the ead_node structure obtained by using the EADS client library. If you specify any other pointer, correct operation is not guaranteed.

For the format and details of the ead_node structure, see *20.1.50 ead_node structure (object used in a user function with an EADS server specified)*.

An error results if the address information (IP address and port number) managed by the specified ead_node structure does not match the address information of any EADS servers maintained by the EADS client.

error_code

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

This function returns a list of the keys (ead_keys structure) that are stored on the specified EADS server in ascending order based on their ASCII code values.

NULL is returned in the keys member of the ead_keys structure in the following cases:

- The specified EADS server does not contain the source keys for copy processing.
- Acquisition of a list of keys failed due to an error.

For details about the ead_keys structure and its format, see *20.1.41 ead_keys structure (multiple keys)*.

(5) Notes

- If the keys member of the ead_keys structure that is returned as the return value is not NULL, the memory area is not freed automatically. You must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.
- As the number of keys on the specified EADS server increases, the time and the amount of resources required for the acquisition processing might increase.

20.1.23 ead_get_group_count() (acquires the number of groups in the highest hierarchy)

(1) Description

This function acquires the number of groups in the highest hierarchy that are stored on a specified EADS server.

(2) Format

```
#include <eads.h>
size_t ead_get_group_count
(
  const EAD_CACHE      *cp,              /* In */
  const ead_node       *target_node,     /* In */
  int                  *error_code       /* Out */
);
```

(3) Arguments

cp

Specifies the handle (pointer) to the cache in which the number of groups is to be acquired.

Specify the handle (pointer) obtained from ead_start_cache() when access to the cache was started.

target_node

Specifies a pointer to the EADS server (ead_node structure) from which the number of groups in the highest hierarchy is to be acquired.

You can only specify the pointer of the ead_node structure obtained by using the EADS client library. If you specify any other pointer, correct operation is not guaranteed.

For the format and details of the `ead_node` structure, see *20.1.50 ead_node structure (object used in a user function with an EADS server specified)*.

An error results if the address information (IP address and port number) managed by the specified `ead_node` structure does not match the address information of any EADS servers maintained by the EADS client.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

This function returns the number of groups in the highest hierarchy that are stored on the specified EADS server.

If acquisition of the number of groups fails due to an error, the function returns `0`.

20.1.24 ead_get_group_key_count() (acquires the number of keys with group specification)

(1) Description

This function acquires the number of keys that belong to a specified group. The number of keys acquired includes the keys that belong to groups under the specified group's hierarchy.

(2) Format

```
#include <eads.h>
size_t ead_get_group_key_count
(
  const EAD_CACHE      *cp,             /* In */
  const char           *group_name,     /* In */
  int                  *error_code      /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache in which the number of keys is to be acquired.

Specify the handle (pointer) obtained from `ead_start_cache()` when access to the cache was started.

`group_name`

Specifies a group name.

For details about the data that can be specified, see *15.2.2(2) Data that can be specified as group names*.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

This function returns the number of keys that belong to the specified group.

If acquisition of the number of keys fails due to an error, the function returns 0.

20.1.25 ead_get_node_key_count() (acquires the number of keys with EADS server specification)

(1) Description

This function acquires the number of keys stored on a specified EADS server.

(2) Format

```
#include <eads.h>
size_t ead_get_node_key_count
(
  const EAD_CACHE     *cp,             /* In */
  const ead_node      *target_node,    /* In */
  int                 *error_code      /* Out */
);
```

(3) Arguments

cp

Specifies the handle (pointer) to the cache in which the number of keys is to be acquired.

Specify the handle (pointer) obtained from ead_start_cache() when access to the cache was started.

target_node

Specifies the pointer to the EADS server (ead_node structure) from which the number of keys is to be acquired.

You can only specify the pointer of the ead_node structure obtained by using the EADS client library. If you specify any other pointer, correct operation is not guaranteed.

For the format and details of the ead_node structure, see *20.1.50 ead_node structure (object used in a user function with an EADS server specified)*.

An error results if the address information (IP address and port number) managed by the specified ead_node structure does not match the address information of any EADS servers maintained by the EADS client.

error_code

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

This function returns the number of keys stored on the specified EADS server.

If acquisition of the number of keys fails due to an error, the function returns 0.

20.1.26 ead_get_group_first_key() (acquires the first key with group specification)

(1) Description

This function acquires the first key in ascending order based on its ASCII code value from among all the keys that belong to a specified group. The keys that belong to groups under the specified group's hierarchy are also subject to this acquisition processing.

(2) Format

```
#include <eads.h>
char* ead_get_group_first_key
(
  const EAD_CACHE     *cp,              /* In */
  const char          *group_name,      /* In */
  int                 *error_code       /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache in which the key is to be acquired.

Specify the handle (pointer) obtained from `ead_start_cache()` when access to the cache was started.

`group_name`

Specifies a group name.

For details about the data that can be specified, see *15.2.2(2) Data that can be specified as group names*.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

This function returns the first key in ascending order based on its ASCII code value from among all the keys that belong to the specified group.

The function returns `NULL` in the following cases:

- No keys belong to the specified group.
- Acquisition of the key failed due to an error.

(5) Notes

If acquisition of the key is successful (the return value is not `NULL`), the memory area is not freed automatically. You must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.

20.1.27 ead_get_node_first_key() (acquires the first key with EADS server specification)

(1) Description

This function acquires the first key in ascending order based on its ASCII code value from among all the keys that are stored on a specified EADS server.

(2) Format

```
#include <eads.h>
char* ead_get_node_first_key
(
  const EAD_CACHE     *cp,             /* In */
  const ead_node      *target_node,    /* In */
  int                 *error_code      /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache in which the key is to be acquired.

Specify the handle (pointer) obtained from `ead_start_cache()` when access to the cache was started.

`target_node`

Specifies the pointer to the EADS server (`ead_node` structure) from which the first key is to be acquired.

You can only specify the pointer of the `ead_node` structure obtained by using the EADS client library. If you specify any other pointer, correct operation is not guaranteed.

For the format and details of the `ead_node` structure, see *20.1.50 ead_node structure (object used in a user function with an EADS server specified)*.

An error results if the address information (IP address and port number) managed by the specified `ead_node` structure does not match the address information of any EADS servers maintained by the EADS client.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

This function returns the first key in ascending order based on its ASCII code value from among all the keys that are stored on the specified EADS server.

The function returns `NULL` in the following cases:

- The specified EADS server does not contain the source key for copy processing.
- Acquisition of the key failed due to an error.

(5) Notes

If acquisition of the key is successful (the return value is not NULL), the memory area is not freed automatically. You must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.

20.1.28 ead_get_group_next_key() (acquires the next key with group specification)

(1) Description

This function acquires the key that immediately follows a specified key in ascending order based on its ASCII code value from among all the keys that belong to a specified group. The keys that belong to the groups under the specified group's hierarchy are also subject to this acquisition processing.

If the specified key does not exist on the connection-target EADS server, the function similarly acquires the key that immediately follows the specified key.

(2) Format

```
#include <eads.h>
char* ead_get_group_next_key
(
  const EAD_CACHE      *cp,              /* In */
  const char           *group_name,      /* In */
  const char           *key,             /* In */
  int                  *error_code       /* Out */
);
```

(3) Arguments

cp

Specifies the handle (pointer) to the cache in which the key is to be acquired.

Specify the handle (pointer) obtained from ead_start_cache() when access to the cache was started.

group_name

Specifies a group name.

For details about the data that can be specified, see *15.2.2(2) Data that can be specified as group names*.

key

Specifies the reference key.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

error_code

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

This function returns the key that immediately follows the specified key in ascending order based on its ASCII code value from among all the keys that belong to the specified group.

The function returns `NULL` in the following cases:

- No key follows the specified key.
- Acquisition of the key failed due to an error.

(5) Notes

- If acquisition of the key is successful (the return value is not `NULL`), the memory area is not freed automatically. You must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.
- This function determines the connection-target EADS server based on the specified group.
- The EADS server that stores the keys belonging to the specified group might change due to isolation, restoration, or addition (scale-out) processing on EADS servers other than the connection-target EADS server. Therefore, if you have obtained the first key by executing `ead_get_group_first_key()`, connection might be established with a different EADS server.
- Because groups are not locked on EADS servers, keys that belong to groups might be inserted or deleted by another process after `ead_get_group_first_key()` or `ead_get_group_next_key()` has executed.

20.1.29 ead_get_node_next_key() (acquires the next key with EADS server specification)

(1) Description

This function acquires the key that immediately follows a specified key in ascending order based on its ASCII code value from among all the keys that are stored on a specified EADS server.

If the specified key does not exist on the connection-target EADS server, the function similarly acquires the key that immediately follows the specified key.

(2) Format

```
#include <eads.h>
char* ead_get_node_next_key
(
  const EAD_CACHE     *cp,              /* In */
  const ead_node      *target_node,     /* In */
  const char          *key,             /* In */
  int                 *error_code       /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache in which the key is to be acquired.

Specify the handle (pointer) obtained from `ead_start_cache()` when access to the cache was started.

`target_node`

Specifies the pointer to the EADS server (`ead_node` structure) from which the key is to be acquired.

You can only specify the pointer of the ead_node structure obtained by using the EADS client library. If you specify any other pointer, correct operation is not guaranteed.

For the format and details of the ead_node structure, see *20.1.50 ead_node structure (object used in a user function with an EADS server specified)*.

An error results if the address information (IP address and port number) managed by the specified ead_node structure does not match the address information of any EADS servers maintained by the EADS client.

key

Specifies the reference key.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

error_code

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

This function returns the key that immediately follows the specified key in ascending order based on its ASCII code value from among all the keys that are stored on the specified server.

The function returns NULL in the following cases:

- No key follows the specified key.
- Acquisition of the key failed due to an error.

(5) Notes

- If acquisition of the key is successful (the return value is not NULL), the memory area is not freed automatically. You must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.
- The EADS server that stores the keys belonging to the specified group might change due to isolation, restoration, or addition (scale-out) processing on EADS servers other than the connection-target EADS server.
- Because groups are not locked on EADS servers, keys that belong to groups might be inserted or deleted by another process after ead_get_node_first_key() or ead_get_node_next_key() has executed.

20.1.30 ead_execute_function() (executes a user function with key specification or group specification)

(1) Description

This function uses a specified key or group to determine the EADS server on which a user function is to be executed and then executes that user function.

(2) Format

```
#include <eads.h>
ead_object ead_execute_function
(
  const EAD_CACHE      *cp,                        /* In */
```

```
    const char          *key_or_group_name,       /* In */
    const char          *func_name,               /* In */
    const ead_object    *arg,                     /* In */
    int                 *error_code               /* Out */
);
```

(3) Arguments

cp

Specifies the handle (pointer) to the cache for the user function to be executed.

Specify the handle obtained from ead_start_cache().

key_or_group_name

Specifies a key or a group name.

For details about the data that can be specified, see *15.2.2(1) Data types that can be specified as keys* or *15.2.2(2) Data that can be specified as group names*.

func_name

Specifies the name of the user function.

A user function name can consist of single-byte alphanumeric characters (0 to 9, A to Z, and a to z), underscores (_), periods (.), and dollar signs ($).

There is no limit to the number of characters.

arg

Specifies the arguments (ead_object structure) to pass to the user function.

For details about the ead_object structure, see *20.1.48 ead_object structure (object used in a user function)*.

If there are no arguments to be passed, specify NULL.

error_code

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

If ead_execute_function() terminates normally, it returns the user function execution results (ead_object structure).

For details about the ead_object structure, see *20.1.48 ead_object structure (object used in a user function)*.

The object_size member of the ead_object structure will contain the size of the byte array specified for the user function's return value.

NULL is set in the object member of the ead_object structure in the following cases:

- A problem occurred during execution of the user function.
- The user function execution results could not be acquired due to a problem such as a network failure.
- null is returned as the result of executing the user function.
- A byte array of data size 0 is returned as the result of executing the user function.

(5) Notes

The memory area for storing the user function execution results that are returned as the return value is not freed automatically. Instead, you must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.

20.1.31 ead_execute_function_rt() (executes a user function with key specification or group specification and reception timeout specification)

(1) Description

This function uses a specified key or group to determine the EADS server on which a user function is to be executed and then executes that user function. This function also sets a reception timeout value.

The value of the `eads.client.connection.receive.timeout` parameter in the client properties is not applicable while `ead_execute_function_rt()` is executing.

(2) Format

```
#include <eads.h>
ead_object ead_execute_function_rt
(
  const EAD_CACHE      *cp,                   /* In */
  const char           *key_or_group_name,    /* In */
  const char           *func_name,            /* In */
  const ead_object     *arg,                  /* In */
  int                  recv_timeout,          /* In */
  int                  *error_code            /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache in which the user function is to be executed.

Specify the handle (pointer) obtained from `ead_start_cache()` when access to the cache was started.

`key_or_group_name`

Specifies a key or a group name.

For details about the data that can be specified, see *15.2.2(1) Data types that can be specified as keys* or *15.2.2(2) Data that can be specified as group names*.

`func_name`

Specifies the name of the user function.

A user function name can consist of single-byte alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`), underscores (`_`), periods (`.`), and dollar signs (`$`).

There is no limit to the number of characters.

`arg`

Specifies the arguments (`ead_object` structure) to pass to the user function.

For details about the `ead_object` structure, see *20.1.48 ead_object structure (object used in a user function)*.

If there are no arguments to be passed, specify `NULL`.

`recv_timeout`

Specifies a data reception timeout value (in milliseconds).

For details about the data that can be specified, see *9.3.3(3)(b) eads.client.connection.receive.timeout*.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

If `ead_execute_function_rt()` terminates normally, it returns the user function execution results (`ead_object` structure).

For details about the `ead_object` structure and its format, see *20.1.48 ead_object structure (object used in a user function)*.

The `object_size` member of the `ead_object` structure will contain the size of the byte array specified for the user function's return value.

`NULL` is set in the `object` member of the `ead_object` structure in the following cases:

- A problem occurred during execution of the user function.
- The user function execution results could not be acquired due to a problem such as a network failure.
- `null` is returned as the result of executing the user function.
- A byte array of data size 0 is returned as the result of executing the user function.

(5) Notes

The memory area for storing the user function execution results that is returned as the return value is not freed automatically. You must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.

20.1.32 ead_execute_node_function() (executes a user function with an EADS server specified)

(1) Description

This function executes a user function with an EADS server specified.

(2) Format

```
#include <eads.h>
ead_object ead_execute_node_function
(
  const EAD_CACHE       *cp,              /* In */
  const ead_node        *target_node,    /* In */
```

```
    const char          *func_name,     /* In */
    const ead_object    *arg,           /* In */
    int                 *error_code     /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache for the user function to be executed.

Specify the handle obtained from `ead_start_cache()`.

`target_node`

Specifies a pointer to the EADS server (`ead_node` structure) that will execute the user function.

You can only specify the pointer of the `ead_node` structure obtained by using the EADS client library. If you specify any other pointer, correct operation is not guaranteed.

For the format and details of the `ead_node` structure, see *20.1.50 ead_node structure (object used in a user function with an EADS server specified)*.

An error results if the address information (IP address and port number) managed by the specified `ead_node` structure does not match the address information of any EADS servers maintained by the EADS client.

`func_name`

Specifies the name of the user function.

A user function name can consist of single-byte alphanumeric characters (`0` to `9`, `A` to `Z`, and `a` to `z`), underscores (`_`), periods (`.`), and dollar signs (`$`).

There is no limit to the number of characters.

`arg`

Specifies the arguments (`ead_object` structure) to pass to the user function.

For details about the `ead_object` structure, see *20.1.48 ead_object structure (object used in a user function)*.

If there are no arguments to be passed, specify `NULL`.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

If `ead_execute_node_function()` terminates normally, it returns the user function execution results (`ead_object` structure).

For details about the `ead_object` structure, see *20.1.48 ead_object structure (object used in a user function)*.

The `object_size` member of the `ead_object` structure will contain the size of the byte array specified for the user function's return value.

`NULL` is set in the `object` member of the `ead_object` structure in the following cases:

- A problem occurred during execution of the user function.
- The user function execution results could not be acquired due to a problem such as a network failure.
- `null` is returned as the result of executing the user function.

- A byte array of data size 0 is returned as the result of executing the user function.

(5) Notes

The memory area for storing the user function execution results that are returned as the return value is not freed automatically. Instead, you must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.

20.1.33 ead_execute_node_function_rt() (executes a user function with EADS server and reception timeout specification)

(1) Description

This function executes a user function with an EADS server specified and sets a reception timeout value.

The value of the `eads.client.connection.receive.timeout` parameter in the client properties is not applicable while `ead_execute_node_function_rt()` is executing.

(2) Format

```
#include <eads.h>
ead_object ead_execute_node_function_rt
(
  const EAD_CACHE      *cp,              /* In */
  const ead_node       *target_node,     /* In */
  const char           *func_name,       /* In */
  const ead_object     *arg,             /* In */
  int                  recv_timeout,     /* In */
  int                  *error_code       /* Out */
);
```

(3) Arguments

`cp`

Specifies the handle (pointer) to the cache on which the user function is to be executed.

Specify the handle (pointer) obtained from `ead_start_cache()` when access to the cache was started.

`target_node`

Specifies the pointer to the EADS server (`ead_node` structure) on which the user function is to be executed.

You can only specify the pointer of the `ead_node` structure obtained by using the EADS client library. If you specify any other pointer, correct operation is not guaranteed.

For the format and details of the `ead_node` structure, see *20.1.50 ead_node structure (object used in a user function with an EADS server specified)*.

An error results if the address information (IP address and port number) managed by the specified `ead_node` structure does not match the address information of any EADS servers maintained by the EADS client.

`func_name`

Specifies the name of the user function.

A user function name can consist of single-byte alphanumeric characters (0 to 9, A to Z, and a to z), underscores (_), periods (.), and dollar signs ($).

There is no limit to the number of characters.

arg

Specifies the arguments (ead_object structure) to pass to the user function.

For details about the ead_object structure, see *20.1.48 ead_object structure (object used in a user function)*.

If there are no arguments to be passed, specify NULL.

recv_timeout

Specifies a data reception timeout value (in milliseconds).

For details about the data that can be specified, see *9.3.3(3)(b) eads.client.connection.receive.timeout*.

error_code

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

If ead_execute_node_function_rt() terminates normally, it returns the user function execution results (ead_object structure).

For details about the ead_object structure and its format, see *20.1.48 ead_object structure (object used in a user function)*.

The object_size member of the ead_object structure will contain the size of the byte array specified for the user function's return value.

NULL is set in the object member of the ead_object structure in the following cases:

- A problem occurred during execution of the user function.
- The user function execution results could not be acquired due to a problem such as a network failure.
- null is returned as the result of executing the user function.
- A byte array of data size 0 is returned as the result of executing the user function.

(5) Notes

The memory area for storing the user function execution results that is returned as the return value is not freed automatically. You must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.

20.1.34 ead_get_nodelist() (acquires information about the connection-target EADS servers)

(1) Description

This function acquires information about the connection-target EADS servers maintained by the EADS client.

(2) Format

```
#include <eads.h>
ead_nodelist ead_get_nodelist
(
  const EAD_CACHE_MANAGER *cmp,              /* In */
  int                     *error_code        /* Out */
);
```

(3) Arguments

`cmp`

Specifies the handle (pointer) to the cache manager that is managing the cache.

Specify a handle obtained from `ead_init_client()` or `ead_init_client_n()`.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

This function returns information about the connection-target EADS servers (`ead_nodelist` structure) maintained by the EADS client.

For the format and details of the `ead_nodelist` structure, see *20.1.49 ead_nodelist structure (EADS server information)*.

If the function terminates abnormally, `NULL` is returned in the `nodes` member of the `ead_nodelist` structure.

(5) Notes

- If the `nodes` member of the `ead_nodelist` structure returned as the return value is not `NULL`, the memory area is not freed automatically. Instead, you must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.
- Execution of the application program does not establish communication with the EADS servers. Therefore, the returned information about the connection-target EADS servers might not be the most recent information.
- Whether the EADS servers are connected is not checked when this function executes. For this reason, the acquired information might contain EADS servers that cannot be connected, for example, because the EADS servers are isolated. If you plan to use an acquired EADS server as a connection target, use the `is_enable` member of the `ead_node` structure to check whether a connection can be established with that EADS server.

  For the format and details of the `ead_node` structure, see *20.1.50 ead_node structure (object used in a user function with an EADS server specified)*.

20.1.35 [Deprecated] ead_get_node() (acquires information about the original source EADS server from which a specified key was copied)

Important note

This function is deprecated. Instead, use `ead_get_original_master_node()`.

(1) Description

This function acquires information about the original source EADS server that stores a specified key or group.

By original source EADS server is meant the EADS server that stores the original data of a specified key or group when all EADS servers making up the cluster can be connected successfully.

(2) Format

```
#include <eads.h>
ead_node ead_get_node
(
  const EAD_CACHE_MANAGER *cmp,               /* In */
  const char              *key,               /* In */
  int                     *error_code         /* Out */
);
```

(3) Arguments

`cmp`

Specifies the handle (pointer) to the cache manager that is managing the cache. Specify a handle obtained from `ead_init_client()` or `ead_init_client_n()`.

`key`

Specifies a key that is stored by the target EADS server.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

This function returns information (`ead_node` structure) about the original source EADS server that stores the specified key or group.

If the function terminates abnormally, it sets `0` in each member of the `ead_node` structure.

(5) Notes

- Execution of the application program does not establish communication with the EADS server. Therefore, the returned information about the connection-target EADS server might not be the most recent information.

- If there have been no changes to the cluster configuration, information about the same EADS server will always be acquired, regardless of whether that EADS server can be connected.
- If you plan to use the acquired EADS server as a connection target, use the is_enable member of the ead_node structure to check whether a connection can be established with that EADS server.

  For the format and details of the ead_node structure, see *20.1.50 ead_node structure (object used in a user function with an EADS server specified)*.

20.1.36 ead_get_slave_nodelist() (acquires information about the original target EADS server for data)

(1) Description

This function acquires information about the original target EADS servers to which data stored on a specified EADS server is copied.

By original target EADS server is meant an EADS server to which data stored on a specified EADS server (source EADS server) is copied when all EADS servers making up the cluster can be connected successfully.

(2) Format

```
#include <eads.h>
ead_nodelist ead_get_slave_nodelist
(
  const EAD_CACHE_MANAGER *cmp,              /* In */
  const ead_node          *master_node,      /* In */
  int                     *error_code        /* Out */
);
```

(3) Arguments

cmp

Specifies the handle (pointer) to the cache manager that is managing the cache. Specify a handle obtained from ead_init_client() or ead_init_client_n().

master_node

Specifies a pointer to the EADS server (ead_node structure) from which data is copied.

You can only specify the pointer of the ead_node structure obtained by using the EADS client library. If you specify any other pointer, correct operation is not guaranteed.

For the format and details of the ead_node structure, see *20.1.50 ead_node structure (object used in a user function with an EADS server specified)*.

An error results if the address information (IP address and port number) managed by the specified ead_node structure does not match the address information of any EADS servers maintained by the EADS client.

error_code

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

This function returns information about the original target EADS servers (ead_nodelist structure).

For the format and details of the ead_nodelist structure, see *20.1.49 ead_nodelist structure (EADS server information)*.

If redundant copies of data are not created (multiplicity is 1), zero is returned to the list_size member of the ead_nodelist structure and NULL to the nodes member.

If the function terminates abnormally, NULL is returned in the nodes member of the ead_nodelist structure.

(5) Notes

- If the nodes member of the ead_nodelist structure returned as the return value is not NULL, the memory area is not freed automatically. You must free it in the application program. For details, see *19.1.1(10) Freeing a memory area returned as a return value*.
- Execution of the application program does not establish communication with the EADS servers. Therefore, the returned information about the connection-target EADS servers might not be the most recent information.
- Whether the specified EADS server and the target EADS server for information acquisition are connected is not checked when this function is executed. Therefore, the acquired information might contain EADS servers that cannot be connected, for example, because the EADS servers are isolated.

20.1.37 ead_get_current_master_node() (acquires information about the current source EADS server)

(1) Description

This function acquires information about the source EADS server that currently stores a specified key (or group).

(2) Format

```
#include <eads.h>
ead_node ead_get_current_master_node
(
  const EAD_CACHE_MANAGER     *cmp,           /* In */
  const char                  *key,           /* In */
  int                         *error_code     /* Out */
);
```

(3) Arguments

cmp

Specifies a handle (pointer) to the cache manager that manages the cache.

Specify a handle obtained from ead_init_client() or ead_init_client_n().

key

Specifies a key (or a group).

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

This function returns information about the source EADS server that currently stores the specified key (or group).

If the function terminates abnormally or the source EADS server is isolated or stopped, it sets `0` in each member of the `ead_node` structure.

(5) Notes

Execution of this function does not establish communication with the EADS server. Therefore, the returned information about the connection-target EADS server might not be the most recent information.

20.1.38 ead_get_original_master_node() (acquires information about the original source EADS server)

(1) Description

This function acquires information about the original source EADS server that stores a specified key (or group).

By original source EADS server is meant the EADS server that stores the master copy (source data) of a specified key (or group) when all EADS servers making up the cluster can be connected successfully. This EADS server might be different from the current source EADS server.

If there have been no changes to the cluster configuration, information about the same EADS server will always be acquired, regardless of whether the EADS server can be connected.

(2) Format

```
#include <eads.h>
ead_node ead_get_original_master_node
(
  const EAD_CACHE_MANAGER     *cmp,          /* In */
  const char                  *key,          /* In */
  int                         *error_code    /* Out */
);
```

(3) Arguments

`cmp`

Specifies a handle (pointer) to the cache manager that manages the cache.

Specify a handle obtained from `ead_init_client()` or `ead_init_client_n()`.

`key`

Specifies a key (or a group).

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`error_code`

Specifies the pointer from which to retrieve the error code.

For details about error codes, see *20.2 Error codes in the client library (C)*.

(4) Return value

This function returns information about the original source EADS server that stores the specified key (or group).

If the function terminates abnormally, it sets `0` in each member of the `ead_node` structure.

(5) Notes

- Execution of this function does not establish communication with the EADS server. Therefore, the returned information about the connection-target EADS server might not be the most recent information.
- If you plan to use the acquired EADS server as a connection target, use the `is_enable` member of the `ead_node` structure to check whether a connection can be established with that EADS server. For details about the `is_enable` member of the `ead_node` structure, see *20.1.50(2) Descriptions of members*.

20.1.39 ead_value_element structure (value information)

The `ead_value_element` structure holds the value information (the value and its size).

(1) Format

```
struct ead_value_element {
  size_t     value_size;
  void       *value;
};
```

(2) Descriptions of members

`value_size`

Stores the data size of the value (unit: bytes).

`value`

Stores the value.

Specify the starting address of the data to be stored.

(3) Notes

- If the range specified by the starting address + the data size references an invalid region, the operation is not guaranteed.
- You cannot specify `NULL` for `value`.
- You cannot specify `0` for `value_size`.
- There is no limit to `value_size`. Specify a value that does not exceed the maximum size of data that can be accepted by the EADS servers.

20.1.40 ead_key_value_pair structure (key-value pairs)

The `ead_key_value_pair` structure holds information about key-value pairs.

(1) Format

```
struct ead_key_value_pair {
  char                *key;
  ead_value_element   value;
};
```

(2) Descriptions of members

`key`

Stores the key to be associated with a value.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

`value`

Stores the value information (`ead_value_element` structure).

For details about the `ead_value_element` structure, see *20.1.39 ead_value_element structure (value information)*.

20.1.41 ead_keys structure (multiple keys)

The `ead_keys` structure holds multiple keys.

(1) Format

```
struct ead_keys {
  size_t   size;
  char     **keys;
};
```

(2) Descriptions of members

`size`

Stores the number of keys to be stored.

`keys`

Stores the start address of the array that stores the list of keys.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

20.1.42 ead_group_names structure (multiple group names)

The `ead_group_names` structure holds multiple group names.

(1) Format

```
struct ead_group_names {
  size_t   size;
  char     **group_names;
};
```

(2) Descriptions of members

size

Stores the number of group names that are stored.

group_names

Stores the start address of the array that stores the list of group names.

For details about the data types that can be specified, see *15.2.2(1) Data types that can be specified as keys*.

20.1.43 ead_put_all_results structure (execution results of ead_put_all())

The ead_put_all_results structure holds the execution results of ead_put_all().

(1) Format

```
struct ead_put_all_results {
  size_t                       success_operation_number;
  size_t                       failure_operation_number;
  ead_failure_operation_info   *failure_info;
};
```

(2) Descriptions of members

success_operation_number

Stores the number of keys that were processed successfully when an attempt to perform batch operation failed partially or entirely.

failure_operation_number

Stores the number of elements of the failure_info member when an attempt to perform batch operation failed partially or entirely.

failure_info

Stores a pointer indicating the beginning of the list (ead_failure_operation_info structure) of information about the failed batch operation.

For details about the ead_failure_operation_info structure and its format, see *20.1.47 ead_failure_operation_info structure (information about the failed operation during batch operation)*.

20.1.44 ead_get_all_results structure (execution results of ead_get_all())

The ead_get_all_results structure holds the execution results of ead_get_all().

(1) Format

```
struct ead_get_all_results {
  size_t                      values_length;
  ead_value_element           *values;
  size_t                      success_operation_number;
  size_t                      failure_operation_number;
  ead_failure_operation_info  *failure_info;
};
```

(2) Descriptions of members

`values_length`

Stores the number of acquired values.

`values`

Specifies the start address of the array of the `ead_value_element` structure that stores the acquired value information.

For details about the `ead_value_element` structure, see *20.1.39 ead_value_element structure (value information)*.

`success_operation_number`

Stores the number of keys that were processed successfully when an attempt to perform batch operation failed partially or entirely.

`failure_operation_number`

Stores the number of elements of the `failure_info` member when an attempt to perform batch operation failed partially or entirely.

`failure_info`

Stores a pointer indicating the beginning of the list (`ead_failure_operation_info` structure) of information about the failed batch operation.

For details about the `ead_failure_operation_info` structure and its format, see *20.1.47 ead_failure_operation_info structure (information about the failed operation during batch operation)*.

20.1.45 ead_get_group_results structure (execution results of ead_get_group())

The `ead_get_group_results` structure holds the execution results of `ead_get_group()`.

(1) Format

```
struct ead_get_group_results {
  size_t                 key_value_length;
  ead_key_value_pair     *key_value_array;
};
```

(2) Descriptions of members

key_value_length

Stores the number of elements in the array of the ead_key_value_pair structure that stores acquired keys and values.

key_value_array

Specifies the start address of the array of the ead_key_value_pair structure that stores acquired keys and values.

For details about the ead_key_value_pair structure and its format, see *20.1.40 ead_key_value_pair structure (key-value pairs)*.

20.1.46 ead_remove_all_results structure (execution results of ead_remove_all())

The ead_remove_all_results structure holds the execution results of ead_remove_all().

(1) Format

```
struct ead_remove_all_results {
  size_t                       success_operation_number;
  size_t                       failure_operation_number;
  ead_failure_operation_info   *failure_info;
};
```

(2) Descriptions of members

success_operation_number

Stores the number of keys that were processed successfully when an attempt to perform batch operation failed partially or entirely.

failure_operation_number

Stores the number of elements of the failure_info member when an attempt to perform batch operation failed partially or entirely.

failure_info

Stores a pointer indicating the beginning of the list (ead_failure_operation_info structure) of information about the failed batch operation.

For details about the ead_failure_operation_info structure and its format, see *20.1.47 ead_failure_operation_info structure (information about the failed operation during batch operation)*.

20.1.47 ead_failure_operation_info structure (information about the failed operation during batch operation)

The ead_failure_operation_info structure holds information about a failed operation when an attempt to perform batch operation failed partially or entirely.

(1) Format

```
struct ead_failure_operation_info {
  int index;
  int error_code;
};
```

(2) Descriptions of members

`index`

Stores a position (subscript) in the list of keys specified during execution of the batch operation.

`error_code`

Stores the error code indicating the cause of the error.

20.1.48 ead_object structure (object used in a user function)

The `ead_object` structure holds an object (such as an argument or return value) used in a user function.

(1) Format

```
struct ead_object {
  size_t     object_size;
  void       *object;
};
```

(2) Descriptions of members

`object_size`

Specifies the data size of the object (unit: bytes).

`object`

Stores an object (such as an argument or return value) used in a user function.

Specify the starting address of the data to be stored.

(3) Notes

- If the range specified by the starting address + the data size references an invalid region, the operation is not guaranteed.
- You cannot specify `NULL` for `object`.
- You cannot specify `0` for `object_size`.
- There is no limit to `object_size`. Specify a value that does not exceed the maximum size of data that can be accepted by the EADS servers.

20.1.49 ead_nodelist structure (EADS server information)

The ead_nodelist structure contains information about the EADS servers that are maintained by the EADS client.

(1) Format

```
struct ead_nodelist {
  size_t          list_size;
  struct ead_node *nodes;
};
```

(2) Descriptions of members

list_size

Stores the number of EADS servers.

nodes

Stores information about the EADS servers (ead_node structure).

Specify the start address of the data to be stored.

20.1.50 ead_node structure (object used in a user function with an EADS server specified)

The ead_node structure contains information needed for specifying EADS servers.

(1) Format

```
struct ead_node {
  int                node_id;
  EAD_BOOL           is_enable;
  int                position;
  struct ead_address address;
  char               reserved[8];
};
```

(2) Descriptions of members

node_id

Stores the EADS server ID.

is_enable

Stores the EADS server's status:

EAD_TRUE: Connection can be established.

EAD_FALSE: Connection cannot be established.

position

Stores the position of the EADS server.

address

Stores the address information (ead_address structure) of the EADS server.

For the format and details of the `ead_address` structure, see *20.1.51 ead_address structure (EADS server address information)*.

`reserved[8]`

Reserved area.

20.1.51 ead_address structure (EADS server address information)

The `ead_address` structure contains EADS server address information.

(1) Format

```
struct ead_address {
  unsigned char      ip[4];
  unsigned short     port;
};
```

(2) Descriptions of members

`ip`

Stores the EADS server's IP address.

`port`

Stores the EADS server's port number.

20.2 Error codes in the client library (C)

The error codes described below are returned by the functions in the C client library, in the location specified in the argument `error_code`.

The following table lists each error code along with its meaning and cause.

Table 20–2: Error codes returned by functions in the C client library

Error code	Symbolic constant	Nature of error	Cause	Processing status of data updating API function#	Value of error_code when 0300 is specified in the eads.client.compat parameter in the client properties
0	`EAD_OK`	Processing terminated normally.	Not applicable.	Y	0
1000	`EAD_ERROR_UNEXPECTED`	An unexpected error occurred.	An unexpected error occurred in the program.	U	1000
1010	`EAD_ERROR_ILLEGAL_ARGUMENT`	The specified parameter is invalid.	An invalid parameter was specified in an API function argument.	N	1010
1040	`EAD_ERROR_CACHE_NOT_STARTED`	The operation could not be executed because the cache has not been started.	An attempt was made to manipulate data after the cache had stopped (after execution of `ead_stop_cache()`).	N	1040
1100	`EAD_ERROR_CACHE_NOT_NEED_STOP`	An attempt was made to stop a cache that has already stopped.	Because the cache has already stopped, there is no need to execute `ead_stop_cache().`	--	1100
1110	`EAD_ERROR_INVALID_NODE_ADDRESS`	The specified EADS server's address information does not match any EADS server address information managed by the EADS client.	The address information (IP address and port number) of the EADS server specified in the argument of the API function does not match any EADS server address information maintained by the EADS client.	--	1110
1120	`EAD_ERROR_EXCEED_MAX_CONNECTION_POOL_SIZE`	The number of connections to be pooled for the same connection target has already reached the	The number of concurrent threads issuing requests to the same EADS server has exceeded	N	1120

Error code	Symbolic constant	Nature of error	Cause	Processing status of data updating API function#	Value of error_code when 0300 is specified in the eads.client.compat parameter in the client properties
		maximum value and all of them are in use.	the maximum number of connections.		
2000	EAD_ERROR_INIT	An error occurred during initialization of the EADS client.	An unexpected error occurred while executing ead_init_client() or ead_init_client_n().	--	2000
2010	EAD_ERROR_INIT_PROPERTIES	The client property file could not be read.	The following are possible causes of the error: • The client property file does not exist. • The client property file does not have read permissions. • The specified storage destination path name is a directory, not a file.	--	2010
2020	EAD_ERROR_INIT_INVALID_PROPERTY	There is an invalid property in the client property file.	A property in the client property file is invalid.	--	2020
2030	EAD_ERROR_INIT_LOGGER	An attempt to initialize the logger failed.	The following are possible causes of the error: • The specified directory or log file at the output destination does not have write permissions. • A file with the same name already exists in the specified directory. • The specified path name or file name is invalid. • There is a directory that has the same	--	2030

Error code	Symbolic constant	Nature of error	Cause	Processing status of data updating API function#	Value of error_code when 0300 is specified in the eads.client.compat parameter in the client properties
			name as the log file. • There is not enough memory to start log output.		
2040	EAD_ERROR_INIT_CLUSTERINFO	An attempt to connect to the EADS server specified in the client property file failed.	The following are possible causes of the error: • There is an error in the client property specification of the connection-target EADS server. • Communication could not be established with the connection-target EADS server. Or, a failure occurred on the connection-target EADS server. • The maximum number of simultaneous connections to the EADS server has been exceeded. • The connection-target EADS server is not ready to accept requests.	--	2040
3000	EAD_ERROR_NET	A communication error has occurred with the EADS server.	The following are possible causes of the error: • A network failure occurred during communication. • A problem occurred on the EADS server with which the client was communicating.	U	3000

Error code	Symbolic constant	Nature of error	Cause	Processing status of data updating API function#	Value of error_code when 0300 is specified in the eads.client.compat parameter in the client properties
			• A problem occurred on the host with which the client was communicating.		
3001	EAD_ERROR_NET_SEND_REQUEST	A communication error occurred while a request was being sent to an EADS server.	The following are possible causes of the error: • A network failure occurred during communication. • A problem occurred on the EADS server with which the client was communicating. • A problem occurred on the host with which the client was communicating.	U	3000
3002	EAD_ERROR_NET_RECEIVE_RESPONSE	A communication error occurred while a response was being received from an EADS server.	The following are possible causes of the error: • A network failure occurred during communication. • A problem occurred on the EADS server with which the client was communicating. • A problem occurred on the host with which the client was communicating.	U	3000
3010	EAD_ERROR_NET_TIMEOUT	A timeout occurred during communication with the EADS server.	The following are possible causes of the error: • A problem occurred on the connection-target EADS server. • A problem occurred on the	U	3010

Error code	Symbolic constant	Nature of error	Cause	Processing status of data updating API function#	Value of error_code when 0300 is specified in the eads.client.compat parameter in the client properties
			connection-target host. • A network problem occurred. • The timeout interval setting is incorrect.		
3011	`EAD_ERROR_NET_SEND_TIMEOUT`	A timeout occurred while a request was being sent to an EADS server.	The following are possible causes of the error: • A problem occurred on the connection-target EADS server. • A problem occurred on the connection-target host. • A network problem occurred. • The timeout interval setting is incorrect.	U	3010
3012	`EAD_ERROR_NET_RECEIVE_TIMEOUT`	A timeout occurred while a response was being received from an EADS server.	The following are possible causes of the error: • A problem occurred on the connection-target EADS server. • A problem occurred on the connection-target host. • A network problem occurred. • The timeout interval setting is incorrect.	U	3010
3020	`EAD_ERROR_NET_CONNECTION`	The connection to the EADS server failed.	The following are possible causes of the error: • A problem occurred on the connection-	N	3020

Error code	Symbolic constant	Nature of error	Cause	Processing status of data updating API function#	Value of error_code when 0300 is specified in the eads.client.compat parameter in the client properties
			target EADS server. - The settings for the connection-target EADS server are incorrect. - A network problem occurred. - The timeout interval setting is incorrect.		
3030	EAD_ERROR_NET_PROTOCOL	A protocol failure occurred during communication with the EADS server.	The following are possible causes of the error: - There might be a problem with the connection-target EADS server. - The executed user function returned a value that is neither a byte array nor NULL. - The cache for the connected cluster is in use by Java client libraries.	U	3030
3040	EAD_ERROR_NET_CLUSTERINFO	It was not possible to connect to all of the available EADS servers.	The following are possible causes of the error: - Problems occurred on the connection-target EADS servers. - The settings for the connection-target EADS servers are incorrect. - A network problem occurred. - The specified cache does not exist on the connection-	--	3040

Error code	Symbolic constant	Nature of error	Cause	Processing status of data updating API function#	Value of error_code when 0300 is specified in the eads.client.compat parameter in the client properties
			target EADS server. - The connection-target EADS server has been closed. - The cluster information maintained by the EADS client does not match the cluster information maintained by the restarted connection-target EADS server. - The maximum number of simultaneous connections to the EADS server has been exceeded.		
4000	EAD_ERROR_SERVER	An unexpected internal error occurred on the EADS server.	An unexpected internal error occurred on the connection-target EADS server.	U	4000
4010	EAD_ERROR_SERVER_UNSUPPORTED_REQUEST	The request sent by the EADS client could not be processed by the connection-target EADS server.	Possible causes are as follows: - The connection-target EADS server cannot process the request for a reason such as corrupted data. - The API function used is not supported by the connection-target EADS server.	N	4010
4030	EAD_ERROR_SERVER_UNAVAILABLE	The connection-target EADS server process is temporarily unavailable.	The maximum number of simultaneous connections to the EADS server has been exceeded.	N	4030

Error code	Symbolic constant	Nature of error	Cause	Processing status of data updating API function#	Value of error_code when 0300 is specified in the eads.client.compat parameter in the client properties
4040	EAD_ERROR_SERVER_INCOMPATIBLE_CLUSTERINFO	The cluster information maintained by the connection-target EADS server is not compatible with the cluster information maintained by the EADS client.	The cluster information maintained by the restarted connection-target EADS server does not match the cluster information maintained by the EADS client.	N	4040
4060	EAD_ERROR_SERVER_REPLACE_METHOD_NOT_MATCHED	The value could not be stored because the value existing during execution of ead_replace() did not match the value specified in the condition.	The value specified in the condition in ead_replace() did not match the value in the cache.	N	4060
4070	EAD_ERROR_SERVER_REPLACE_METHOD_KEY_NOT_EXIST	During execution of ead_replace(), processing could not continue because the specified key could not be found (no value for the key could be found).	The values could not be compared because no value could be found for the key specified in ead_replace().	N	4070
4080	EAD_ERROR_SERVER_CREATE_METHOD_KEY_EXIST	During execution of ead_create(), the value could not be stored because the specified key already exists.	A value was already stored for the key specified in ead_create().	N	4080
4090	EAD_ERROR_SERVER_UPDATE_METHOD_KEY_NOT_EXIST	During execution of ead_update(), the value could not be stored because the specified key was not found.	No value was stored for the key specified in ead_update().	N	4090
4100	EAD_ERROR_SERVER_NOT_RUNNING	No EADS server is available for processing requests.	Possible causes are as follows: • The EADS server that processes requests from the EADS client and the EADS servers to which data is to be copied are all isolated or stopped.	N	4100

Error code	Symbolic constant	Nature of error	Cause	Processing status of data updating API function#	Value of error_code when 0300 is specified in the eads.client.compat parameter in the client properties
			- The cluster is not available.		
4110	EAD_ERROR_SERVER_STATUS	The EADS server is in a status in which requests cannot be processed.	The request could not be processed due to the status of the connection-target EADS server.	N	4000
4200	EAD_ERROR_SERVER_CACHE	A cache operation failed.	An operation could not be performed on a cache because a problem occurred on the connection-target EADS server. Stop the operation and check the EADS server's status.	N	4000
4210	EAD_ERROR_SERVER_CACHE_NOT_FOUND	A cache operation failed because the specified cache did not exist.	An operation could not be performed on a cache because the specified cache did not exist. Stop operation on the specified cache and check the EADS server's status.	N	4000
4230	EAD_ERROR_SERVER_CACHE_CLUSTER_UPDATE	A cache operation failed because the cluster configuration was changed during request processing.	An operation could not be performed on a cache because the cluster configuration was changed during request processing. Perform the cache operation again after the cluster configuration change processing has been completed.	N	4000
4300	EAD_ERROR_SERVER_CACHE_BEFORE_REPLICATION	An internal error occurred during a cache operation, but redundant copies of data had not been created.	An internal error occurred during a cache operation on the connection-target EADS server. No other normal EADS servers are affected because redundant copies of data had not been created. You can restart the same	N	4000

Error code	Symbolic constant	Nature of error	Cause	Processing status of data updating API function#	Value of error_code when 0300 is specified in the eads.client.compat parameter in the client properties
			operation after the EADS server is isolated and then the connection target is changed to a normal EADS server.		
4310	EAD_ERROR_SERVER_CACHE_AFTER_REPLICATION	An internal error occurred on the EADS server during cache operation and the data update operation failed.	An internal error occurred on the connection-target EADS server during cache operation. Because redundant copies of data had already been created, once the erroneous connection-target EADS server is isolated and the connection target is changed to a normal server, you can restart the operation from the status in which data had been updated.	U	4000
4700	EAD_ERROR_SERVER_FUNCTION_EXECUTE	An error occurred in the user function on the EADS server.	An error occurred in the user function on the connection-target EADS server. Check the user function's processing.	--	4000
4710	EAD_ERROR_SERVER_FUNCTION_RETURN_SERIALIZE	Serialization processing on the return value of the user function failed on the EADS server.	An object that is not serializable is specified for the return value of the user function executed on the connection-target EADS server.	--	4000
4730	EAD_ERROR_SERVER_FUNCTION_NOT_FOUND	No user function with the specified user function name exists on the EADS server.	No user function with the specified user function name exists on the connection-target EADS server.	--	4000
4800	EAD_ERROR_SERVER_LIMIT_EXTERNAL_MEMORY	There is a shortage of memory for storing data.	The request could not be processed because the memory for storing data	N	4000

Error code	Symbolic constant	Nature of error	Cause	Processing status of data updating API function#	Value of error_code when 0300 is specified in the eads.client.compat parameter in the client properties
			(explicit heap) was insufficient on the connection-target EADS server.		
4810	EAD_ERROR_SERVER_LIMIT_CACHE_FILE	There is a shortage of capacity in the cache files for storing data.	The request could not be processed because the capacity of cache files for storing data was insufficient on the connection-target EADS server.	N	4000
4820	EAD_ERROR_SERVER_LIMIT_KV_COUNT	The number of keys that can be stored on the EADS server has reached the upper limit.	The request could not be processed because the number of keys that can be specified on the connection-target EADS server had reached the upper limit.	N	4000
4830	EAD_ERROR_SERVER_LIMIT_KEY_VALUE_LENGTH	The size of the specified key, group name, or value is greater than the maximum size permitted in the cluster.	The request could not be processed because the size of the specified key, group name, or value was greater than the maximum size permitted in the cluster.	N	4000
4999	EAD_ERROR_SERVER_UNKNOWN	A nonanalyzable internal error occurred on the EADS server.	An internal error occurred on the connection-target EADS server, but the error could not be analyzed because the version of the connection-target EADS server was later than the version of the EADS client libraries.	U	4000
5000	EAD_ERROR_CLIENT	An internal error occurred on an EADS client.	An unexpected error occurred in client libraries.	U	5000
5010	EAD_ERROR_CLIENT_OUT_OF_MEMORY	Memory allocation in the EADS client failed.	Memory allocation failed in client libraries.	U	5010

Error code	Symbolic constant	Nature of error	Cause	Processing status of data updating API function#	Value of error_code when 0300 is specified in the eads.client.compat parameter in the client properties
5020	EAD_ERROR_CLIENT_BATCH_CANCEL	Batch operation was cancelled.	Unperformed operations were cancelled because batch operation could not be continued.	N	5020
6000	EAD_ERROR_BATCH_FAILED_ALL	All of the batch operations failed.	An attempt was made to perform a batch operation on data by using an API function, but all operations failed.	--	6000
6010	EAD_ERROR_BATCH_FAILED_PART	Part of the batch operations failed.	An attempt was made to perform a batch operation on data by using an API function, but some of the operations failed.	--	6010

#

Indicates whether data updating had occurred when the error code was issued during execution of an API function for updating data, such as `ead_put()` or `ead_remove()`.

The meanings of the letters in this column are as follows:

Y: The data had been updated.

U: Whether the data had been updated is unknown. Check whether the processing was completed.

N: The data had not been updated.

--: This error code is not issued when an API function for updating data, such as `ead_put()` or `ead_remove()`, is executed.

21 Useful Lists

This chapter lists and explains the maximum and minimum values.

21.1 List of minimum and maximum values

The following table lists minimum and maximum values in EADS.

Table 21–1: Minimum and maximum values

No.	Item	Minimum value	Maximum value
1	Number of EADS client connections per EADS server	1	1,024
2	Number of EADS servers that make up one cluster	1#1	96
3	Number of caches that can be created per cluster	0	16
4	Number of data items that can be stored on one EADS server#2	0	2,147,483,647
5	Data multiplicity	1	5
6	Length of a key (in characters)	1	1,024
7	Size of a value#3 (in bytes)	1	262,144
8	Length of a group name (in characters)	1	1,022
9	Length of a cache name (in characters)	1	32
10	Number of groups	1	No limit
11	Number of group hierarchies	1	511
12	Size of all cache data files in a range (in megabytes)	16	128
13	Number of cache data files in a range	8	32,768
14	Number of data items that can be specified in one cache data file	--	131,070

Legend:

--: Not applicable.

#1

The number of EADS servers that make up a cluster must be at least the data multiplicity × 2 - 1.

#2

Includes the number of data items copied by data redundancy processing.

#3

For details about the size of a value, see *15.2.2(3) Data types that can be specified as values*.

22 Messages

This chapter lists the messages that are output, and explains their meanings and the corrective actions to take.

22.1 Message output format

XXXXnnnnn-Y

message-text

Description of variable values

Description

Message description that supplements *message-text*

Action

Description of the action that the user is recommended to take

The table below lists and describes each of these elements in more detail. Note that for some messages there might not be a description of variable values or an action to take in response to the message.

Item	Description
XXXXnnnnn	This represents the message ID. It is composed of the following elements: *XXXX* This is `KDEA`, the message prefix for EADS messages. *nnnnn* This represents the message number maintained by EADS. Each message has a unique five-digit number.
Y	This represents the level of the message. The level of the message is indicated by a single letter. The meanings of the letters that indicate the message levels are as follows: **E (Error)** This is a message notifying you that an error has occurred. When this message is output, processing is suspended. **W (Warning)** This is a message notifying you that a warning-level problem has occurred. After this message is output, processing continues. **I (Information)** This is a message notifying you about system operations. After this message is output, processing continues.
Message text	This represents the message text output by EADS.
Description of variable values	The format *xx....xx*: *displayed-information* is used to indicate information displayed in the variable values in the message text (where *xx* are lowercase letters). The following is an example of a description of variable values: Example: *aa....aa*: IP address *bb....bb*: Port number
Description	This provides information that supplements the message text, such as the cause of the problem reported in the message, or the type of EADS operation that produced the message.
Action	This indicates the action that the user is recommended to take.

22.2 KDEA00001 to KDEA01999

This section describes messages `KDEA00001` to `KDEA01999` and explains the corrective actions to take in response to each message.

KDEA00001-I

The server will now start. (server name = *aa....aa*)

aa....aa: EADS server name (management directory name)

Description

The EADS server is starting.

KDEA00002-I

The server will now shut down. (server name = *aa....aa*)

aa....aa: EADS server name (management directory name)

Description

The EADS server is stopping.

KDEA00003-W

The server status was set to "ISOLATED".

Description

The EADS server has become isolated.

Action

The possible causes are as follows:

- The `eztool isolate` command was executed.
- A problem that prevents operation occurred on the EADS server.

See *12.2.1 If one or more EADS servers are isolated*, and take action according to the instructions given in this subsection.

KDEA00004-E

Log initialization failed. (directory path = *aa....aa*)

aa....aa: Path name of the log output destination

Description

An attempt to initialize the log failed.

Action

The possible causes are as follows:

- The specified log output directory is invalid.
- There is a problem in the log output directory or with the log file permissions.

Determine the cause of the error, eliminate it, and then restart the EADS server.

KDEA00005-E

A property file cannot be read. (file path = *aa....aa*)

aa....aa: Path name to the property file

Description

The property file cannot be read.

The possible causes are as follows:

- The property file does not exist.
- The property file cannot be opened.
- The path name points to a directory rather than to a file.

Action

Determine the cause of the error from the standard output, message logs, or exception logs, eliminate it, and then restart the EADS server.

KDEA00006-E

The server failed to start. Startup of the server will now end.

Description

Startup processing failed.

Startup of the EADS server is stopping because a problem occurred during startup processing of the EADS server.

Action

Determine the cause of the error by checking the message that was output immediately before this message, eliminate the cause, and then restart the EADS server.

KDEA00016-E

An attempt to open a port failed. (address = *aa....aa*:*bb....bb*)

aa....aa: IP address

bb....bb: Port number

Description

The communication port cannot be opened.

The possible causes are as follows:

- The specified IP address is invalid.
- The specified port number is already in use.
- The same port number was specified for different parameters.
- The specified port number is a well-known port for which the user does not have permissions.

Action

Check and, if necessary, revise the values specified in the following server property parameters:

- `eads.server.address`
- `eads.server.port`
- `eads.admin.operation.port`

Determine the cause of the error from the exception logs, eliminate it, and then restart the EADS server.

KDEA00020-I

Initialization of the server will now start. (version = *aa....aa*)

aa....aa: EADS server version

Description

Initialization of the EADS server is starting.

KDEA00021-I

Initialization of the server was completed.

Description

Initialization of the EADS server has been completed.

KDEA00028-E

An exception occurred on the server. (exception = *aa....aa*)

aa....aa: Exception (or error) that occurred

Description

An exception (or error) occurred on the EADS server.

The EADS server has stopped.

Action

Determine the cause of the error from the exception logs and the message that was output immediately before this message, eliminate the cause, and then restart the EADS server.

KDEA00029-E

The specified cache was not found. (cache name = *aa....aa*)

aa....aa: Cache name

Description

The specified cache was not found. An operation was performed on a nonexistent cache.

Action

Create the cache, or else check the processing and revise it if necessary.

KDEA00031-E

The thread stopped. (thread name = *aa....aa*)

aa....aa: Thread name

Description

A problem occurred on the EADS server, and the thread stopped.

Action

Contact the customer support center.

KDEA00032-E

An exception occurred while waiting for a connection request. (local = *aa....aa*:*bb....bb*)

aa....aa: IP address of the EADS server (string representation of `java.net.InetAddress`)

bb....bb: Port number of the EADS server

Description

An exception occurred while waiting for a connection request.

Action

The possible causes are as follows:

- A problem occurred on the EADS client with which communication was underway.
- A problem occurred on the EADS server with which communication was underway.
- A problem occurred on the host with which communication was underway.
- A network problem occurred.

Determine the cause of the error from the exception logs and then eliminate it.

KDEA00033-W

The number of received connection requests exceeds the maximum number of simultaneous connections. (local = *aa....aa*:*bb....bb*, remote = *cc....cc*:*dd....dd*, max connections = *ee....ee*)

aa....aa: IP address of the EADS server (string representation of `java.net.InetAddress`)

bb....bb: Port number of the EADS server

cc....cc: IP address of the connection source (string representation of `java.net.InetAddress`)

dd....dd: Port number of the connection source

ee....ee: Maximum number of simultaneous connections

Description

The server received a connection request that exceeded the maximum number of simultaneous connections.

The EADS server returns an error and cuts off communication for connections exceeding the maximum number of simultaneous connections.

Note that this message might also be output in the following cases:

- All EADS servers in the cluster are restarted while the EADS client is running.
- A connection timeout occurs between the EADS client and the EADS server due to a temporary increase in load on the EADS server.

Action

Check and, if necessary, revise the maximum number of simultaneous connections (the server property parameter `eads.server.maxConnections`).

KDEA00036-E

An exception occurred during request handling. (local = *aa....aa*:*bb....bb*, remote = *cc....cc*:*dd....dd*)

aa....aa: IP address of the EADS server (string representation of `java.net.InetAddress`)

bb....bb: Port number of the EADS server

cc....cc: IP address of the connection source (string representation of `java.net.InetAddress`)

dd....dd: Port number of the connection source

Description

An exception occurred while the request was being processed.

Action

The possible causes are as follows:

- A problem occurred on the EADS client with which communication was underway.
- A problem occurred on the host with which communication was underway.
- A network problem occurred.

Determine the cause of the problem, and then eliminate it.

KDEA00039-E

An exception or error occurred on the server. (details = *aa....aa*)

aa....aa: Details

Description

An exception (or error) occurred on the EADS server while a request was being processed. As a result, communication was cut off.

Action

Determine the cause of the error from the exception logs and then eliminate it.

If it remains unresolved, contact the customer support center.

KDEA00041-I

The port will now close. (address = *aa....aa*:*bb....bb*)

aa....aa: IP address (string representation of `java.net.InetAddress`)

bb....bb: Port number

Description

The port is closing.

KDEA00042-I

The library was added to the class path. (file path = *aa....aa*)

aa....aa: Path name of the library file

Description

The library was added to the classpath.

KDEA00045-E

A received request is not supported by the server. (local = *aa....aa*:*bb....bb*, remote = *cc....cc*:*dd....dd*)

aa....aa: IP address of the EADS server

bb....bb: Port number of the EADS server

cc....cc: IP address of the connection source

dd....dd: Port number of the connection source

Description

A received request is not supported by the EADS server.
The request will not be processed.

Action

The possible causes are as follows:

- An invalid EADS client is connected.
- A different version of the EADS client is connected.

Make sure that the correct EADS client is being used.

KDEA00047-E

An error occurred during reception of a request. (local = *aa....aa*:*bb....bb*, remote = *cc....cc*:*dd....dd*)

aa....aa: IP address of the EADS server (string representation of `java.net.InetAddress`)

bb....bb: Port number of the EADS server

cc....cc: IP address of the connection source (string representation of `java.net.InetAddress`)

dd....dd: Port number of the connection source

Description

An error occurred during reception processing of the request.

Action

The possible causes are as follows:

- A problem occurred on the EADS client with which communication was underway.
- A problem occurred on the host with which communication was underway.
- A network problem occurred.

Determine the cause of the error from the exception logs and then eliminate it.

KDEA00048-E

An error occurred during the sending of a response. (local = *aa....aa*:*bb....bb*, remote = *cc....cc*:*dd....dd*)

aa....aa: IP address of the EADS server (string representation of `java.net.InetAddress`) *bb....bb*: Port number of the EADS server

cc....cc: IP address of the connection source (string representation of `java.net.InetAddress`)

dd....dd: Port number of the connection source

Description

An error occurred during send processing of the response.

Action

The possible causes are as follows:

- A problem occurred on the EADS client with which communication was underway.
- A problem occurred on the host with which communication was underway.
- A network problem occurred.

Determine the cause of the error from the exception logs and then eliminate it.

KDEA00049-I

A redirect notification was sent to the client. (local = *aa....aa*:*bb....bb*, remote = *cc....cc*:*dd....dd*, redirect = *ee....ee*:*ff....ff*)

aa....aa: IP address of the EADS server (string representation of `java.net.InetAddress`)

bb....bb: Port number of the EADS server

cc....cc: IP address of the connection source (string representation of `java.net.InetAddress`)

dd....dd: Port number of the connection source

ee....ee: IP address of the redirect destination (string representation of `java.net.InetAddress`)

ff....ff: Port number of the redirect destination

Description

The EADS client is notified of the redirection.

KDEA00050-E

An unexpected initialization error occurred.

Description

An unexpected initialization error occurred.

Action

Contact the customer support center.

KDEA00051-E

The content of the request is invalid.

Description

The content of the request is invalid.

Action

The possible causes are as follows:

- A problem occurred on the EADS client with which communication was underway.
- A problem occurred on the host with which communication was underway.
- A network problem occurred.

Determine the cause of the error, and then eliminate it.

KDEA00052-E

An unexpected exception occurred.

Description

An unexpected exception occurred.

Action

Contact the customer support center.

KDEA00053-E

Loading of the library file failed.

Description

An attempt to load the library file failed.

Action

The possible causes are as follows:

- The library file is invalid.
- There is a problem in the library file or directory permissions.

Determine the cause of the error, and then eliminate it.

KDEA00054-I

A server port was opened. (local = *aa....aa*:*bb....bb*)

aa....aa: IP address of the EADS server (string representation of `java.net.InetAddress`)

bb....bb: Port number of the EADS server

Description

A port was opened.

KDEA00055-I

The server will now start accepting requests. (local = *aa....aa*:*bb....bb*)

aa....aa: IP address of the EADS server (string representation of `java.net.InetAddress`)

bb....bb: Port number of the EADS server

Description

The server will now start accepting requests.

KDEA00056-E

An unexpected exception occurred. (detail = *aa....aa*)

aa....aa: Maintenance information

Description

An unexpected exception occurred.

Action

Contact the customer support center.

KDEA00057-E

An attempt to allocate the data area has failed. (type = *aa....aa*, data size = *bb....bb*)

aa....aa: Type

bb....bb: Size of the memory area whose allocation was attempted

Description

An attempt to allocate memory failed because there is not enough available memory area.

The following table provides the cause by type:

Type	Cause
`store`	There is not enough available memory area for storing values.

Action

Take the following action according to the type:

Type	Action
`store`	Increase the value of the `eads.java.external.heapsize` parameter in the shared properties.

KDEA00058-E

The input data are too long. (type = *aa....aa*, request = *bb....bb*, limit = *cc....cc*)

aa....aa: Type

bb....bb: Required size

cc....cc: Size limit

Description

The size required for the input data exceeds the size limit.

The following table provides the cause by type:

Type	Cause
`store`	The value specified in the `eads.java.external.heapsize` shared property parameter is too small for the size of the value. Alternatively, the value size exceeds the maximum value that can be specified.
`dist`	The value of the `eads.replication.external.heapsize` parameter in the shared properties is too small for the size of the history of update operations.

Action

Make sure that the value size does not exceed the maximum value that can be specified.

If the value size does not exceed the maximum value that can be specified, take the action appropriate for the type as shown in the following table:

Type	Action
`store`	Increase the value of the `eads.java.external.heapsize` parameter in the shared properties.
`dist`	Increase the value of the `eads.replication.external.heapsize` parameter in the shared properties. If you increase this parameter value, also increase the value of the `eads.java.external.heapsize` parameter in the shared properties.

KDEA00060-W

The fragmentation of data occurred. (type = *aa....aa*, count = *bb....bb*)

aa....aa: Type

bb....bb: Number of times data fragmentation occurred

Description

The data stored in the explicit heap was fragmented.

The following table provides the cause by type:

Type	Cause
`store`	Values were fragmented.
`dist`	The history of update operations was fragmented.

The displayed count indicates the number of times fragmentation has occurred since the last time this message was output.

Action

If this message is output frequently, fragmentation might have occurred due to an inadequate explicit heap and as a result performance might have become degraded. Take the following action according to the type:

Type	Action
`store`	Increase the value of the `eads.java.external.heapsize` parameter in the shared properties.
`dist`	Increase the value of the `eads.replication.external.heapsize` parameter in the shared properties. If you increase this parameter value, also increase the value of the `eads.java.external.heapsize` parameter in the shared properties.

KDEA00061-E

The size of key exceeds the maximum. (maximum size (bytes) = *aa....aa*, key size (bytes) = *bb....bb*)

aa....aa: Maximum size (unit: bytes)

bb....bb: key size (unit: bytes)

Description

The specified key size exceeds the maximum size of a key that can be stored in the cluster.

Action

Check and, if necessary, revise the specified key size.

Check and, if necessary, revise the value of the `eads.cache.key.maxsize` parameter in the shared properties.

KDEA00062-E

The size of group name exceeds the maximum. (maximum size (bytes) = *aa....aa*, group name size (bytes) = *bb....bb*)

aa....aa: Maximum size (unit: bytes)

bb....bb: Group name size (unit: bytes)

Description

The specified group name is longer than the maximum size of a group name that can be stored in the cluster (the value of the `eads.cache.key.maxsize` parameter in the shared properties minus 2).

Action

Check and, if necessary, revise the length of the specified group name.

Check and, if necessary, revise the value of the `eads.cache.key.maxsize` parameter in the shared properties.

KDEA01004-I

Initialization of the user function is complete. (function name = *aa....aa*)

aa....aa: Name of the user function

Description

Initialization of the user function is complete.

KDEA01005-W

An exception or error occurred in the init method of the user function. (function name = *aa....aa*)

aa....aa: Name of the user function

Description

An exception (or error) occurred in `init()` in the user function.

Action

Check the user exception logs, eliminate the cause of the error, and then restart the EADS server.

KDEA01009-I

Processing to terminate the user function is complete. (function name = *aa....aa*)

aa....aa: Name of user function

Description

Termination processing of the user function is complete.

KDEA01010-W

Processing to terminate the user function failed. (function name = *aa....aa*) *aa....aa*: Name of user function

Description

An attempt to terminate processing of the user function failed (an exception (or error) occurred in `destroy()` in the user function).

Action

Check the user exception logs and eliminate the cause of the error.

KDEA01011-E

An exception or error occurred in the user function. (function name = *aa....aa*)

aa....aa: Name of user function

Description

An exception (or error) occurred in the user function (an exception (or error) occurred during the execution of `execute()` in the user function).

Action

Check the user exception logs and eliminate the cause of the error.

KDEA01012-W

Generation of a user function instance failed. (function name = *aa....aa*)

aa....aa: Name of user function

Description

An attempt to generate a user function instance failed.

Action

The possible causes are as follows:

- An exception (or error) occurred while the user function instance was being generated.
- You do not have access permissions for the default constructor of the user function.
- The user function is in a format in which its instance cannot be generated.

Check the user exception logs, eliminate the cause of the error, and then restart the EADS server.

KDEA01016-E

An exception or error occurred in the user function call processing. (function name = *aa....aa*, details = *bb....bb*)

aa....aa: User function name to be called

bb....bb: Details

Description

An exception (or error) occurred in the processing of the user function call.

Action

Determine the cause of the error from the details, and then eliminate the problem.

KDEA01017-E

The specified user function was not found. (function name = *aa....aa*)

aa....aa: Name of the user function

Description

The specified user function was not found.

Action

The possible causes are as follows:

- No user function with the specified name exists on the EADS server.
- An attempt was made to execute an invalid user function.

Make sure the user function name is specified correctly.

If the user function name is specified correctly, make sure the specified user function is registered as a valid user function on the EADS server.

KDEA01018-E

An attempt to serialize the user function execution results failed. (function name = *aa....aa*, details = *bb....bb*)

aa....aa: User function name to be called

bb....bb: Details

Description

The object specified in the return value as the user function execution results could not be serialized.

Action

Check whether the specified value is a serializable object.

If it is an object that is not serializable, change it to a serializable object.

KDEA01019-E

An attempt to deserialize the argument failed. (function name = *aa....aa*, details = *bb....bb*)

aa....aa: Name of user function to be called

bb....bb: Details

Description

An argument sent from the EADS client could not be deserialized.

Action

Check the classes in the user function library, and check whether the argument object specified in the EADS client can be deserialized.

If the class is missing, add the class to the user function library or shared library, and then restart the EADS server.

KDEA01021-E

The specified cache was not found. (function name = *aa....aa*, cache name = *bb....bb*)

aa....aa: User function name to be called *bb....bb*: Cache name

Description

An operation was performed on a nonexistent cache.

Action

Either create a cache or check and, if necessary, revise the specified cache name.

KDEA01022-I

Initialization of the user function library will now start. (file path = *aa....aa*)

aa....aa: File path

Description

Initialization of the user function library is starting.

KDEA01023-I

Initialization of the user function library is complete. (file path = *aa....aa*, success = *bb....bb*, failure = *cc....cc*)

aa....aa: File path

bb....bb: Number of user functions that have been initialized successfully

cc....cc: Number of user functions whose initialization failed

Description

Initialization of the user function library has been completed.

KDEA01024-I

No user function library was found.

Description

The user functions will not be initialized because there is no user function library.

KDEA01025-I

"user_function.jar" will not be initialized because a file called "user-function.jar" also exists.

Description

A user function whose file name is `user_function.jar` cannot be initialized because the same user function already exists.

KDEA01026-I

A user function cannot be loaded because a user function with the same fully-qualified class name already exists. (function name = *aa....aa*)

aa....aa: Name of the user function

Description

The target user function cannot be imported because a user function with the same fully qualified class name already exists.

KDEA01027-E

An exception or error occurred in the initialization processing of the user function. (details = *aa....aa*)

aa....aa: Details

Description

An exception (or an error) occurred during user function initialization processing.

Action

Identify the cause of the error from the detailed information and exception logs, eliminate the problem, and then re-start the EADS server.

KDEA01901-I

WRITING (cache = *aa....aa*, range = *bb....bb*, ecf = *cc....cc*, remain = *dd....dd*)

aa....aa: Cache name

bb....bb: Range ID

cc....cc: Path of the cache data files

dd....dd: Number of unused cache data files

Description

Import processing has started on the cache data files.

KDEA01902-I

COMPACTION (cache = *aa....aa*, range = *bb....bb*, ecf = *cc....cc*, size (total = *dd....dd*, relocated = *ee....ee*, valid = *ff....ff* -> *gg....gg*), count (total = *hh....hh*, relocated = *ii....ii*, valid = *kk....kk* -> *mm....mm*), time = *nn....nn*, remain = *pp....pp*)

aa....aa: Cache name

bb....bb: Range ID

cc....cc: Path of the cache data files subject to compaction (absolute path)

dd....dd: Size of space already in use in the files (bytes)

ee....ee: Amount of data moved (bytes)

ff....ff: Amount of effective data before compaction (bytes)

gg....gg: Amount of effective data after compaction (bytes)

hh....hh: Total number of data items including invalid data

ii....ii: Number of data items moved

kk....kk: Number of effective data items before compaction

mm....mm: Number of effective data items after compaction

nn....nn: Processing time (milliseconds)

pp....pp: Number of unused cache data files

Description

Compaction processing was performed on cache data files.

KDEA01903-I

FILE INFO (cache = *aa....aa*, range = *bb....bb*, ecf = *cc....cc*, size = *dd....dd*, allocated size = *ee....ee*)

aa....aa: Cache name

bb....bb: Range ID

cc....cc: Path of the cache data files

dd....dd: File size

ee....ee: Actual size allocated

Description

The actual size of the area allocated to the cache data files differs from the file sizes.

KDEA01904-I

CORRECTION (cache = *aa....aa*, range = *bb....bb*, ecf = *cc....cc*)

aa....aa: Cache name

bb....bb: Range ID

cc....cc: Path of the cache files

Description

Cache files were modified.

22.3 KDEA02000 to KDEA02999

This section describes messages `KDEA02000` to `KDEA02999` and explains what actions to take in response to each message.

KDEA02001-I

Initialization of the client library will now start. (version = *aa....aa*)

aa....aa: EADS client version

Description

Initialization of the client library is starting.

KDEA02002-I

Initialization of the client library completed.

Description

Initialization of the client library is complete.

KDEA02003-E

Initialization of the client library failed.

Description

An attempt to initialize the client library failed.

Action

See *20.2 Error codes in the client library (C)*, and then eliminate the problem.

KDEA02008-I

Cache preparations finished. (cache name = *aa....aa*)

aa....aa: Cache name

Description

The cache is ready to use.

KDEA02009-I

The cache will no longer be used. (cache name = *aa....aa*)

aa....aa: Cache name

Description

Use of the cache is finished.

KDEA02011-I

The client library will now terminate.

Description

The client library is stopping.

KDEA02022-E

The connections to all servers making up the cluster failed.

Description

Connection establishment with all EADS servers constituting the cluster failed.

Action

The possible causes are as follows:

- The cluster was terminated.
- A problem occurred on the EADS server with which communication was underway.
- A problem occurred on the host with which communication was underway.
- A network problem occurred.
- The timeout settings are not appropriate.

Determine the cause of the error, and then eliminate it.

KDEA02023-I

The connection to the cluster was restored.

Description

The connection with the cluster has been restored.

KDEA02024-E

Initialization of the logger failed. (details = *aa....aa*)

aa....aa: Error details

Description

An attempt to initialize the logger failed.

Action

Check the error detail information and eliminate the problem.

KDEA02911-I

start RootAP(address = *aa....aa*, pid = *bb....bb*, no = *cc....cc*)

aa....aa: IP address

bb....bb: PID (number assigned to each application program by the EADS client)

cc....cc: Sequence number (hexadecimal)

Description

Communication with the EADS server is starting.

KDEA02912-I

```
end RootAP(address = aa....aa, pid = bb....bb, no = cc....cc)
```

aa....aa: IP address

bb....bb: PID (number assigned to each application program by the EADS client)

cc....cc: Sequence number (hexadecimal)

Description

Communication with the EADS server is stopping.

22.4 KDEA03000 to KDEA03999

This section describes messages `KDEA03000` to `KDEA03999` and explains what actions to take in response to each message.

KDEA03001-I

Initialization of the client library will now start. (version = *aa....aa*)

aa....aa: EADS client version

Description

Initialization of the client library is starting.

KDEA03002-I

The client library process will now end.

Description

The client library is stopping.

KDEA03003-I

Cache preparations finished. (cache name = *aa....aa*)

aa....aa: Cache name

Description

The cache is ready to use.

KDEA03004-I

Use of the cache will now end. (cache name = *aa....aa*)

aa....aa: Cache name

Description

Use of the cache is finished.

KDEA03006-E

Initialization of the client library failed. (error code = *aa....aa*)

aa....aa: Error code

Description

An attempt to initialize the client library failed.

Action

See *18.1.5 CacheException class* under *(5)(c) Return value*, and then eliminate the problem.

KDEA03019-I

Initialization of the client library completed.

Description

Initialization of the client library is complete.

KDEA03021-E

The connections to all servers making up the cluster failed.

Description

Connection establishment with all EADS servers constituting the cluster failed.

Action

The possible causes are as follows:

- The cluster was terminated.
- A problem occurred on the EADS server with which communication was underway.
- A problem occurred on the host with which communication was underway.
- A network problem occurred.
- The timeout settings are not appropriate.

Determine the cause of the error, and then eliminate it.

KDEA03022-I

The connection to the cluster was restored.

Description

The connection with the cluster has been restored.

KDEA03023-E

Initialization of the logger failed. (details = *aa....aa*)

aa....aa: Error details

Description

An attempt to initialize the logger failed.

Action

Check the error detail information and eliminate the problem.

KDEA03911-I

start RootAP(address = *aa....aa*, pid = *bb....bb*, no = *cc....cc*)

aa....aa: IP address

bb....bb: PID (number assigned to each application program by the EADS client)

cc....cc: Sequence number (hexadecimal)

Description

Communication with the EADS server is starting.

KDEA03912-I

```
end RootAP(address = aa....aa, pid = bb....bb, no = cc....cc)
```

aa....aa: IP address

bb....bb: PID (number assigned to each application program by the EADS client)

cc....cc: Sequence number (hexadecimal)

Description

Communication with the EADS server is stopping.

22.5 KDEA04000 to KDEA05999

This section describes messages KDEA04000 to KDEA05999 and explains what actions to take in response to each message.

KDEA04504-E

An invalid property was detected. (definition file name = *aa....aa*, property name = *bb....bb*, property value = *cc....cc*)

aa....aa: Invalid property file name

bb....bb: Invalid parameter name

cc....cc: Invalid parameter specification value (or blank if there is no value)

Description

An invalid parameter was detected.

Action

Check and, if necessary, revise the parameter indicated in the message.

KDEA04506-E

A data discrepancy in the definition file was detected. (node ID = *aa....aa*, definition file name = *bb....bb*)

aa....aa: EADS server ID of the EADS server whose property file does not match the other EADS servers' property files

bb....bb: Name of the property file whose contents do not match those of the other EADS servers' property files

Description

The contents of the property file do not match the contents of the other EADS servers' property files.

Action

Check if the contents of the property file match the contents of the other EADS servers' property files.

If this error occurs again, the EADS server might be accepting heartbeats from another cluster. If this is the case, make sure that the specified destination IP address or port number (`eads.failureDetector.heartbeat.address` and `eads.failureDetector.heartbeat.port` parameters in the shared properties) is not duplicated in another cluster.

KDEA04508-E

A boot timeout occurred. (node-list = *aa....aa*)

aa....aa: List of IP addresses and port numbers of EADS servers that did not finish starting within the specified time period

Description

Not all the EADS servers that make up the cluster finished startup processing within the specified period of time.

Action

Make sure the EADS servers indicated in the message have been started.

This error might occur when the cluster consists of many EADS servers. If this is the case, change the maximum wait time for all EADS servers making up the cluster to start (`eads.admin.boot.timeout` parameter in the cluster properties). An appropriate value is *number of EADS servers making up the cluster* × 5 seconds.

Also consider starting the EADS servers beginning with the EADS server with the smallest EADS server ID.

KDEA04528-E

A data discrepancy between the cluster-definition file and the server-definition file was detected.

Description

There is a discrepancy between the server property file and the cluster property file.

Action

Check whether the EADS server specified in the server property parameters `eads.server.address` and `eads.server.port` is specified in the cluster property parameters `eads.node.`*EADS-server-ID*`.address` and `eads.node.`*EADS-server-ID*`.port`.

KDEA04531-E

A server position was duplicated. (duplicated node ID1 = *aa....aa*, duplicated node ID2 = *bb....bb*)

aa....aa: EADS server ID of the first EADS server whose position is duplicated

bb....bb: EADS server ID of the second EADS server whose position is duplicated

Description

Two EADS servers making up the cluster have the same position.

Action

Make sure there are no duplicates for the cluster property parameter `eads.node.`*EADS-server-ID*`.position`.

KDEA04540-E

A client connection address was duplicated. (duplicated node ID1 = *aa....aa*, duplicated node ID2 = *bb....bb*)

aa....aa: EADS server ID of the first EADS server whose combination of IP address and port number is duplicated

bb....bb: EADS server ID of the second EADS server whose combination of IP address and port number is duplicated

Description

Two EADS servers have the same definition in the cluster property file.

Action

Check the `eads.node.`*EADS-server-ID*`.address` and `eads.node.`*EADS-server-ID*`.port` parameters in the cluster properties for duplications.

KDEA04541-E

The cluster-definition file contains only part of the server position specification.

Description

The cluster property file contains position specifications for only some of the EADS servers.

Action

If you specify the `eads.node.`*EADS-server-ID*`.position` parameter in the cluster properties, you must specify a position for all the EADS servers. If you want to omit some EADS server positions, do not specify any EADS positions.

KDEA04661-E

Participation in the cluster failed. (error code = *aa....aa*)

aa....aa: Error code

Description

Cluster participation processing failed.

Action

Take the following actions:

1. If necessary, check the message that was output immediately before this message, determined the cause of the error, and then eliminate it.
2. Execute the `eztool status` command to check the cluster status.

   If an EADS server's status is shown as `lock`, it is still locked. In this case, execute the `eztool unlock` command.
3. Execute the `ezstart` or `ezserver` command to perform startup, restoration, or scale-out processing again.

   Note that if a cache failed to resume, but the cache has not been deleted by the `eztool deletecache` command, the EADS servers cannot be restored. In such a case, execute the `eztool deletecache` command to delete the cache that failed to resume.

KDEA04664-E

The relation between the number of servers and the number of replications was invalid. (number of servers = *aa....aa*, number of replications = *bb....bb*)

aa....aa: Number of EADS servers

bb....bb: Data multiplicity

Description

The number of EADS servers making up the cluster is not at least the *number of redundant copies of data* × 2 - 1.

Action

Check the cluster properties and either increase the number of EADS servers so that the number of EADS servers making up the cluster is at least the *number of redundant copies of data* × 2 - 1 or reduce the number of redundant copies of data plus the original (value of the `eads.replication.factor` parameter in the shared properties).

KDEA04665-E

A server cannot be added when the number of replications is 1.

Description

If the data multiplicity (the number of redundant copies of data plus the original) is 1, EADS servers cannot be added.

Action

Check and, if necessary, revise the number of redundant copies of data plus the original (value of the `eads.replication.factor` parameter in the shared properties).

If you have started the cluster with the data multiplicity set to 1, export all needed data, stop all EADS servers, then edit the cluster properties according to the most recent parameter settings. Restart all EADS servers.

KDEA04666-E

The number of servers in the cluster has already reached the maximum.

Description

The number of EADS servers in the cluster has already reached the upper limit.

Action

No more EADS server can be added because 96 EADS servers, which is the maximum number of EADS servers that can make up a cluster, are already participating in the cluster.

Evaluate using another cluster.

KDEA04667-E

In the current state of the cluster, the server cannot participate in the cluster in the specified start type. (start type = *aa....aa*, cluster state = *bb....bb*)

aa....aa: Start method

bb....bb: Cluster status

Description

The EADS server cannot use the specified start method to participate in a cluster that is in the current cluster status.

Action

The possible causes are as follows:

- The option specified in the `ezstart` or `ezserver` command is invalid.
- The EADS server was restarted or restored while it was being isolated.
- The cluster status is not available (not `AVAILABLE`).
- An attempt was made to add an EADS server when at least one EADS server was isolated.

Check the EADS server start options for any error or missing options. Also check if the cluster status satisfies the conditions for the specified start method, and then restart the EADS server.

KDEA04668-E

The server could not participate in the cluster with the conditions specified in the options. (specified options = *aa....aa*, details = *bb....bb*)

aa....aa: Options specified when the EADS server was started

bb....bb: Details

Description

The EADS server could not participate in the cluster with the conditions specified in the options.

Action

The possible causes are as follows:

- There is no EADS server with the specified EADS server ID.
- An EADS server already exists at the specified EADS server location (hash value).
- The range managed by the EADS server with the specified EADS server ID is 1.

Check and, if necessary, revise the option value, and then add the EADS server again.

KDEA04669-E

An attempt to lock the server failed.

Description

An attempt to acquire a lock failed.

Action

Execute the `eztool status -v` command to check the current processing, wait until the processing is finished, and then restart the EADS server. If the processing has already finished but the lock has not been released, execute the `eztool unlock` command to release the lock, and then restart the EADS server.

KDEA04691-I

The transmission of cache data will now start. (cache name = *aa....aa*)

aa....aa: Cache name

Description

Transmission of cache data is beginning.

KDEA04692-I

The cache data was successfully transferred. (cache name = *aa....aa*)

aa....aa: Cache name

Description

Transmission of cache data was successful.

KDEA04693-E

The transmission of cache data failed. (cache name = *aa....aa*, error code = *bb....bb*)

aa....aa: Cache name

bb....bb: Error code

Description

Transmission of cache data failed.

Action

The possible causes are as follows:

- An error occurred on an EADS server from which data was sent.

- A communication error occurred.
- An unexpected error occurred.

Determine the cause of the problem, and then eliminate it.

KDEA04697-E

Transmission of the data failed because a required server was not found.

Description

Transmission of data failed because an active EADS server required for the data transmission was not available.

There is no EADS server that stores required data, or the number of active EADS servers is equal to or less than the number of data copies plus the original.

Action

Take the following actions:

1. Execute the `eztool export` command to export required data to a file.
2. Terminate all active EADS servers.
3. Start all EADS servers.

KDEA04698-E

An error occurred on the server that sent data.(error code = *aa....aa*)

aa....aa: Error code

Description

An error occurred on the EADS server that sent data.

Action

The possible causes are as follows:

- The transmission timed out.
- A communication error occurred.
- An unexpected error occurred.
- Restoration processing or scale-out processing could not be performed.
- An option specified in the `ezstart` or `ezserver` command was invalid.

Determine the cause of the error, eliminate it, and then restart the EADS server.

KDEA04703-E

The master switchover failed because a cache operation failed.

Description

Switchover to the source EADS server from which data is to be copied failed because a data operation failed.

Action

Check the messages that were output immediately before this one.

KDEA04722-E

A specified port is already being used. (self address = *aa....aa*, property name = *bb....bb*, port number = *cc....cc*)

aa....aa: IP address of the local EADS server

bb....bb: Name of the parameter that represents the port number

cc....cc: Specified port number

Description

The specified port number is already being used.

Action

Specify a port number that is not in use by another process.

KDEA04723-E

An unknown message was received. (source address = *aa....aa*, source port = *bb....bb*, header = *cc....cc*)

aa....aa: IP address of the message delivery source

bb....bb: Port number of the message delivery source

cc....cc: Header of the received message (hexadecimal)

Description

An invalid message was received. Another system is sending a message to a communication port that is in use.

Action

Check and, if necessary, revise the destination IP address or port number so that there is no interference in the delivery of messages from another system.

KDEA04725-E

A message of an unsupported protocol version was received. (source address = *aa....aa*, source port = *bb....bb*, protocol version = *cc....cc*)

aa....aa: IP address of the message delivery source

bb....bb: Port number of the message delivery source

cc....cc: Protocol version of the received message

Description

A message with an unsupported protocol version was received.

Systems using different protocol versions have been mixed.

Action

Check if the protocol version matches the installed EADS version.

KDEA04726-W

A connection timeout occurred during failure detection processing. (destination address = *aa....aa*, destination port = *bb....bb*) *aa....aa*: IP address of the connection-target EADS server

bb....bb: Port number of the connection-target EADS server

Description

During the check for live servers, a connection could not be established before the timeout period.

Action

If timeouts occur frequently, check the network devices for a failure.

Alternatively, check and, if necessary, revise the value of the `eads.failureDetector.connection.timeout` parameter in the server properties.

KDEA04727-W

A read timeout occurred during failure detection processing. (destination address = *aa....aa*, destination port = *bb....bb*)

aa....aa: IP address of the connection-target EADS server

bb....bb: Port number of the connection-target EADS server

Description

After establishing a connection during the check for live servers, nothing could be received before the timeout period.

Action

If timeouts occur frequently, check the network devices for a failure.

Alternatively, check and, if necessary, revise the `eads.failureDetector.read.timeout` parameter in the server properties.

KDEA04728-E

An error occurred during establishment of a connection for failure detection processing. (error message = *aa....aa*)

aa....aa: Error message from the exception that was thrown from the JavaVM

Description

While a connection was being established for the check for live servers, a socket could not be created because an exception was thrown from the JavaVM.

Action

Check whether the file descriptor was set according to estimates.

If the error continues to occur, contact the customer support center.

KDEA04752-I

Processing to isolate a server was successful. (server ID = *aa....aa*)

aa....aa: EADS server ID of the EADS server that was isolated

Description

Isolation processing has been completed.

KDEA04755-E

Processing to isolate a server failed because a cache operation failed.

Description

Isolation processing failed due to a data operation error.

Action

Check the messages that were output immediately before this one.

KDEA04781-I

A server will now be added to the cluster. (server ID = *aa....aa*)

aa....aa: EADS server ID of the EADS server that will be added to the cluster

Description

An EADS server is being added to the cluster.

KDEA04783-I

Processing to isolate a server will now start. (server ID = *aa....aa*)

aa....aa: EADS server ID of the EADS server that will be isolated

Description

The `eztool isolate` command was executed. The EADS server will now be isolated.

KDEA04785-I

A command lock will now be acquired.

Description

A command lock will now be acquired.

KDEA04786-I

A command lock could not be acquired.

Description

An attempt to acquire a command lock failed because another command already has a lock.

Action

Re-execute the command after the currently executing command has terminated.

KDEA04787-I

Acquisition of a command lock failed. (error code = *aa....aa*)

aa....aa: Error code

Description

An attempt to acquire a command lock failed.

Action

The possible causes are as follows:

- Another cluster was running.
- The EADS server was terminated.

Determine the cause of the error, eliminate it, and then re-execute the command.

KDEA04789-I

A command lock will now be released.

Description

A command lock will now be released.

KDEA04790-I

An attempt to release a lock failed. (error code = *aa....aa*)

aa....aa: Error code

Description

An attempt to release a command lock failed.

Action

The possible causes are as follows:

- Another cluster was running.
- The EADS server was terminated.

Determine the cause of the error, eliminate it, and then re-execute the `eztool unlock` command to release the lock.

KDEA04792-I

Now waiting for executing requests to complete...

Description

The system is waiting for the current request processing to be completed.

KDEA04793-I

All requests are complete.

Description

The processing of all requests has been completed.

The system will now start processing the requests that arrived while the system was on standby.

KDEA04794-I

The wait for the lock to end will now start.

Description

A wait for completion of locking has started.

KDEA04795-I

> The wait for the lock to end ended successfully.

Description

The wait for completion of locking has ended.

KDEA04799-E

> Processing to isolate a server will now start. (server ID = *aa....aa*)

aa....aa: EADS server ID of the EADS server that will be isolated

Description

A nonoperational EADS server was detected during cluster monitoring. The EADS server will be isolated.

Action

Verify that the `KDEA04752-I` message has been output indicating that the isolation processing has been completed. After that, execute one of the following commands to restore the isolated EADS server:

- `ezstart -r` command
- `ezserver -r` command

KDEA04800-E

> This server will be stopped because a gap in the data could not be filled.

Description

This EADS server will be terminated because consensus processing can no longer be continued in the cluster.

Action

Check if the size of the area for storing the history of update operations has been set as estimated.

If a sufficient size is not specified, change the value of the `eads.replication.external.heapsize` parameter in the shared properties.

Restore the EADS server by executing one of the following commands:

- `ezstart -r` command
- `ezserver -r` command

Alternatively, restart all EADS servers in the cluster.

KDEA04801-E

> This server will be stopped because a gap in the data could not be filled. (cache name = *aa....aa*, range ID = *bb....bb*)

aa....aa: Cache name

bb....bb: Range ID

Description

This EADS server will be stopped because consensus processing can no longer continue in the cluster.

Action

Check if the size of the area for storing the history of update operations has been set as estimated.

If a sufficient size is not specified, change the value of the `eads.replication.external.heapsize` parameter in the shared properties.

Execute one of the following commands to restore the stopped EADS server:

- `ezstart -r` command
- `ezserver -r` command

Alternatively, restart all EADS servers in the cluster.

KDEA04805-E

Only some of the data are available because a server is isolated. (server ID = *aa....aa*, aborted positions = *bb....bb*)

aa....aa: EADS server ID of the EADS server that was isolated

bb....bb: List of the start positions on the consistent hashing in the range of data that cannot be accessed

Description

Some data cannot be accessed because an EADS server has been isolated.

Action

Data cannot be accessed, but the data is still in the cache. Take the following actions:

1. Execute the `eztool export` command to export the required data to a file.
2. Terminate all active EADS servers.
3. Start all EADS servers.

The cluster cannot be restored to normal status by restoring the EADS server.

KDEA04807-E

The system could not continue building a consensus on distribution among servers in the range.

Description

Consensus processing can no longer continue.

Action

Restart all EADS servers in the cluster.

KDEA04808-E

Only some of the data are available because a server is isolated. Some data were lost. (isolated server ID = *aa....aa*, lost data start positions = *bb....bb*, aborted data start positions = *cc....cc*)

aa....aa: EADS server ID of the EADS server that was isolated

bb....bb: List of the start positions on the consistent hashing in the range of missing data

cc....cc: List of the start positions on the consistent hashing in the range of data that cannot be accessed

Description

Some data cannot be accessed because an EADS server has been isolated.

Alternatively, some data has been lost.

Action

Some existing data cannot be accessed, but it is still in cache. Take the following actions:

1. Execute the `eztool export` command to export the required data to a file.
2. Terminate all active EADS servers.
3. Start all EADS servers.

The cluster cannot be restored to normal status by restoring the EADS server.

KDEA04809-W

> The number of running servers is the minimum required for the cluster to be available. (isolated server ID = *aa....aa*, start positions of data to be lost = *bb....bb*, start positions of data to be aborted = *cc....cc*)

aa....aa: EADS server ID of the EADS server that was isolated

bb....bb: If an EADS server has been isolated, a list of the start positions on the consistent hashing in the range of data that might have been lost

cc....cc: If an EADS server has been isolated, a list of the start positions on the consistent hashing in the range of data that might not be accessible

Description

The number of running EADS servers has reached the minimum required for cluster operation.

If an EADS server has been isolated, some data might be inaccessible or lost.

Action

Restore the isolated EADS server by executing one of the following commands:

- `ezstart -r` command
- `ezserver -r` command

KDEA04810-W

> The number of running servers is the minimum required for the cluster to be available. (isolated server ID = *aa....aa*, start positions = *bb....bb*)

aa....aa: EADS server ID of the EADS server that was isolated

bb....bb: If an EADS server has been isolated, a list of the start positions on the consistent hashing in the range of data that might not be accessible

Description

The number of running EADS servers has reached the minimum required for cluster operation.

If an EADS server has been isolated, some data might not be accessible.

Action

Restore the isolated EADS server by executing one of the following commands:

- `ezstart -r` command
- `ezserver -r` command

KDEA04812-E

An unexpected exception occurred. (error code = *aa....aa*)

aa....aa: Error code

Description

An unexpected exception occurred.

Action

Contact the customer support center.

KDEA04813-E

A sequence number is too large and exceeded the upper limit.

Description

An overflow occurred in the internal sequence numbers due to an extended period of operation.

Action

Contact the customer support center.

KDEA04814-W

Server-to-server communication timed out. (timeout value = *aa....aa*, destination addresses = *bb....bb*)

aa....aa: Timeout period

bb....bb: List of IP addresses of the destination EADS servers

Description

Communication between EADS servers timed out.

The possible causes are as follows:

- A failure occurred at a target EADS server, host, or network.
- The timeout period is not appropriate.

Action

Eliminate the cause of the timeout.

Alternatively, increase the value of the `eads.replication.consensus.timeout` parameter in the server properties. However, there is no need to change the parameter value in the following cases:

- A communication error occurred.
- The EADS server was isolated for a reason such as a process error.
- The EADS server was isolated by the `eztool isolate` command.

KDEA04815-E

The cluster status has become NOT_AVAILABLE.

Description

The cluster has been placed in `NOT_AVAILABLE` status.

Either split-brain has occurred or at least half of the EADS servers in the cluster have failed.

Action

If split-brain has occurred, recover the network.

If at least half of the EADS servers in the cluster have failed, restart all EADS servers in the cluster.

KDEA04816-I

The cluster has recovered from the NOT_AVAILABLE state.

Description

The cluster has recovered from `NOT_AVAILABLE` status.

The cluster has recovered from split-brain.

KDEA04817-E

A gap was found that the server could not resolve.

Description

An attempt to synchronize delayed EADS server processing failed because the logs required for synchronization were not available.

Action

The possible causes are as follows:

- An overflow occurred in the consensus message send queue.
- An EADS server was isolated.
- A communication error occurred.
- An unexpected error occurred.

Determine the cause of the error, eliminate it, and then restore the EADS server by executing one of the following commands:

- `ezstart -r` command
- `ezserver -r` command

KDEA04821-W

Processing to delete part of the history in order to resolve a memory shortage in the history storage area has started.

Description

Deletion processing on the existing history of update operations has started because a space shortage occurred on the area for storing the history of update operations.

Action

Check if the size of the area for storing the history of update operations has been set as estimated.

If a sufficient size is not specified, change the value of the `eads.replication.external.heapsize` parameter in the shared properties.

KDEA04822-W

During the building of a consensus on distribution among the servers in the cluster, processing to delete part of the history in order to resolve a memory shortage in the history storage area failed. (requested size = *aa....aa*) *aa....aa*: Size of the area requested for storing the history of update operations

Description

Consensus processing was stopped because an attempt was made to delete the existing history of update operations due to a shortage of space for storing the history of update operations, but the required size of area could not be allocated.

Action

Check the network devices for a failure, wait a while, and then re-execute the command.

If the error occurs again, check if the size of the area for storing the history of update operations has been set as estimated.

If a sufficient size is not specified, change the value of the `eads.replication.external.heapsize` parameter in the shared properties.

KDEA04823-E

During the building of a consensus on distribution among the servers in the cluster, processing to delete part of the history in order to resolve a memory shortage in the history storage area failed. (requested size = *aa....aa*)

aa....aa: Size of the area requested for storing the history of update operations

Description

All processing terminated abnormally because an attempt was made to delete the existing history of update operations due to a shortage of space for storing the history of update operations, but the required size of area could not be allocated.

Action

Check if the size of the area for storing the history of update operations has been set as estimated.

If a sufficient size is not specified, change the value of the `eads.replication.external.heapsize` parameter in the shared properties.

KDEA04824-W

During processing to delete part of the history in order to resolve a memory shortage in the history storage area, processing to build a consensus stopped because the memory shortage could not be resolved. (requested size = *aa....aa*, cache name = *bb....bb*, range ID = *cc....cc*)

aa....aa: Size of the area requested for storing the history of update operations

bb....bb: Cache name

cc....cc: Range ID

Description

Consensus processing was stopped because an attempt was made to delete the existing history of update operations due to a shortage of space for storing the history of update operations, but the required size of area could not be allocated.

Action

Check the network devices for a failure, wait a while, and then re-execute the command.

If the error occurs again, check if the size of the area for storing the history of update operations has been set as estimated.

If a sufficient size is not specified, change the value of the `eads.replication.external.heapsize` parameter in the shared properties.

KDEA04825-E

During processing to delete part of the history in order to resolve a memory shortage in the history storage area, all processing ended abnormally because the memory shortage could not be resolved. (requested size = *aa....aa*, cache name = *bb....bb*, range ID = *cc....cc*)

aa....aa: Size of the area requested for storing the history of update operations

bb....bb: Cache name

cc....cc: Range ID

Description

All processing terminated abnormally because an attempt was made to delete the existing history of update operations due to a shortage of space for storing the history of update operations, but the required size of area could not be allocated.

Action

Check if the size of the area for storing the history of update operations has been set as estimated.

If a sufficient size is not specified, change the value of the `eads.replication.external.heapsize` parameter in the shared properties.

KDEA04841-E

A TCP connection was closed because the queue for sending distribution-consensus messages overflowed. (destination address = *aa....aa*, destination port = *bb....bb*, queuesize = *cc....cc*, queuedatasize = *dd....dd*)

aa....aa: IP address of the connection-target EADS server

bb....bb: Port number of the connection-target EADS server

cc....cc: Number of messages in the send queue

dd....dd: Data size of the messages in the send queue

Description

Connection was closed due to an overflow in the consensus message send queue.

Action

Increase the value of the `eads.replication.sendQueue.length` or `eads.replication.sendQueue.datasize` parameter in the server properties.

KDEA04871-E

Initialization of UDP reception failed because an error occurred during initialization of the receiving socket. (error message = *aa....aa*, multicast address = *bb....bb*, multicast port = *cc....cc*)

aa....aa: Error message from the exception that was thrown from the JavaVM

bb....bb: IP address (multicast address)

cc....cc: Port number

Description

Initialization of UDP reception failed because an error occurred during initialization of the receiving socket.

Action

Determine the cause of the error from the displayed exception error message and eliminate it. Also check if the file descriptor has been set as estimated.

KDEA04872-E

Initialization of UDP transmission failed because an error occurred during initialization of the sending socket. (error message = *aa....aa*, multicast address = *bb....bb*, multicast port = *cc....cc*)

aa....aa: Error message from the exception that was thrown from the JavaVM

bb....bb: IP address (multicast address)

cc....cc: Port number

Description

Initialization of UDP transmission failed because an error occurred during initialization of the sending socket.

Action

Determine the cause of the error from the displayed exception error message and eliminate it. Also check if the file descriptor has been set as estimated.

KDEA04881-E

Initialization of TCP/IP reception failed because an error occurred during initialization of the receiving socket. (error message = *aa....aa*, local address = *bb....bb*, local port = *cc....cc*)

aa....aa: Error message from the exception that was thrown from the JavaVM

bb....bb: Local IP address

cc....cc: Local port number

Description

Initialization of TCP/IP reception failed because an error occurred during initialization of the receiving socket.

Action

Check for any of the following messages:

- `Cannot assign requested address`
  Check if the local IP address is valid in the OS.
- `Address already in use`
  Check if the local port number is in use by another process.
- Other
  Determine the cause of the error from the displayed exception error message and eliminate it. Also check if the file descriptor has been set as estimated.

KDEA04882-E

Initialization of TCP/IP transmission failed because an error occurred during initialization of the sending socket. (error message = *aa....aa*, destination address = *bb....bb*, destination port = *cc....cc*) *aa....aa*: Error message from the exception that was thrown from the JavaVM

bb....bb: Connection-target IP address

cc....cc: Connection-target port number

Description

Initialization of TCP/IP transmission failed because an error occurred during initialization of the sending socket.

Action

Check for any of the following messages:

- `Connection refused`

  Check if the connection-target port number is correct.

  If the connection-target port number is correct, check if the connection-target EADS server has already started.

- `Connection Timedout`

  Check if the connection-target IP address is correct.

  If the connection-target IP address is correct, check if the connection-target OS has already started.

- `java.net.SocketTimeoutException`

  Check if the connection-target IP address is correct.

  If the connection-target IP address is correct, check if the connection-target OS has already started.

- Other

  Determine the cause of the error from the displayed exception error message and eliminate it. Also check if the file descriptor has been set as estimated.

KDEA04883-I

> A different ReceiveBufferSize from the property value will be used. (property name = *aa....aa*, property value = *bb....bb*, ReceiveBufferSize = *cc....cc*)

aa....aa: Parameter name

bb....bb: Specified value of the parameter

cc....cc: Size of consensus message receive buffer that is actually used

Description

A receive buffer size that differs from the specified value was used because the value of the `eads.replication.connection.buffersize` parameter specified in the server properties exceeds the window size supported by the OS.

KDEA04884-I

> A different SendBufferSize from the property value will be used. (property name = *aa....aa*, property value = *bb....bb*, SendBufferSize = *cc....cc*)

aa....aa: Parameter name

bb....bb: Specified value of the parameter

cc....cc: Size of consensus message send buffer that is actually used

Description

A send buffer size that differs from the specified value was used because the value of the `eads.replication.connection.buffersize` parameter specified in the server properties exceeds the window size supported by the OS.

KDEA04887-W

A read timeout occurred in a TCP/IP reception. (local address = *aa....aa*, local port = *bb....bb*, destination address = *cc....cc*, destination port = *dd....dd*)

aa....aa: Local IP address

bb....bb: Local port number

cc....cc: Connection-target IP address

dd....dd: Connection-target port number

Description

A connection was established, but data could not be received within the timeout period.

Action

If timeouts occur frequently, check and, if necessary, revise the value specified in the server property parameter `eads.transfer.timeout`.

KDEA04888-W

A send timeout occurred in a TCP/IP transmission. (local address = *aa....aa*, local port = *bb....bb*, destination address = *cc....cc*, destination port = *dd....dd*)

aa....aa: Local IP address

bb....bb: Local port number

cc....cc: Connection-target IP address

dd....dd: Connection-target port number

Description

A connection was established, but data could not be sent within the timeout period.

Action

If timeouts occur frequently, check and, if necessary, revise the value specified in the server property parameter `eads.transfer.timeout`.

KDEA04891-E

An unknown message was received. (source address = *aa....aa*, source port = *bb....bb*, error message = *cc....cc*)

aa....aa: IP address of the message delivery source

bb....bb: Port number of the message delivery source

cc....cc: Error message

Description

An invalid message was received. Another system is sending a message to a communication port that is in use.

Action

Check and, if necessary, revise the destination IP address or port number so that there is no interference in the delivery of messages from another system

KDEA04892-W

TCP/IP transmission failed because the server is not running.

Description

TCP/IP transmission processing failed because the EADS server terminated.

KDEA04932-E

An error occurred during cache processing. (exception = *aa....aa*, cache name = *bb....bb*)

aa....aa: Exception class name

bb....bb: Cache name (or null if there is no cache name)

Description

An error occurred during cache processing.

Action

Check the messages that were output immediately before this one.

KDEA04962-W

Cache creation was aborted because a consensus on distribution for part of the range is not executable. (cache name = *aa....aa*, management client = *bb....bb*)

aa....aa: Cache name

bb....bb: IP address from which the command was executed

Description

Cache creation was cancelled because consensus processing could not be performed in some part of the range.

Action

Restart all EADS servers in the cluster, and then re-execute the command.

KDEA04965-E

Addition of a server to the cluster failed because the system could not continue building a consensus on distribution among the servers in the cluster. (node id = *aa....aa*)

aa....aa: EADS server ID of the EADS server that was to be added

Description

The processing to add an EADS server to the cluster failed because consensus processing could no longer be continued in the cluster.

Action

Execute the `eztool status` command to check the cluster status. If the cluster status is `AVAILABLE`, the EADS server can be restored.

If the cluster status is `NOT_AVAILABLE` or `PARTIALLY_AVAILABLE`, the EADS server cannot be restored, in which case you must restart all EADS servers in the cluster.

KDEA04966-E

Processing to isolate a server failed because the system could not continue building a consensus on distribution among the servers in the cluster. (node id = *aa....aa*)

aa....aa: EADS server ID of the EADS server that was to be isolated

Description

The processing to isolate an EADS server failed because consensus processing could no longer be continued in the cluster.

Action

Execute the `eztool status` command to check the cluster status. If the cluster status is `AVAILABLE`, the EADS server can be restored.

If the cluster status is `NOT_AVAILABLE` or `PARTIALLY_AVAILABLE`, the EADS server cannot be restored, in which case you must restart all EADS servers in the cluster.

KDEA04967-E

Acquisition of a command lock failed because the system could not continue building a consensus on distribution among the servers in the cluster.

Description

An attempt to acquire a command lock failed because consensus processing could no longer be continued in the cluster.

Action

Execute the `eztool status` command to check the cluster status, and then re-execute the command on another EADS server.

KDEA04968-E

The release of a lock failed because the system could not continue building a consensus on distribution among the servers in the cluster.

Description

An attempt to release a command lock failed because consensus processing could no longer be continued in the cluster.

Action

Re-execute the `eztool unlock` command on another EADS server.

22.6 KDEA06000 to KDEA07999

This section describes messages `KDEA06000` to `KDEA07999` and explains what actions to take in response to each message.

KDEA06001-E

The specified cache already exists. (cache name = *aa....aa*)

aa....aa: Cache name

Description

The specified cache name already exists.

Action

Delete the cache that already exists, or change the cache name.

Ignore the message if no corrective action is required.

KDEA06002-E

The specified cache was not found. (cache name = *aa....aa*)

aa....aa: Cache name

Description

The specified cache was not found.

Action

Create a new cache, or else check the processing and revise it if necessary.

KDEA06004-E

An unexpected exception occurred. (detail = *aa....aa*)

aa....aa: Maintenance information

Description

An unexpected exception occurred.

Action

Contact the customer support center.

KDEA07001-E

An attempt to open a file failed. (file name = *aa....aa*, detail = *bb....bb*)

aa....aa: Store data file name (absolute path) that an attempt was made to open

bb....bb: Details

Description

An attempt to open the store data file failed.

The possible causes are as follows:

- The store data file to be read does not exist.
- A path to non-store data files was specified.
- There are no access permissions to the store data file.

Action

Examine the details, and then check whether there is a problem with the OS environment.

KDEA07002-E

An I/O error occurred. (file name = *aa....aa*, detail = *bb....bb*)

aa....aa: Store data file name (absolute path)

bb....bb: Details

Description

An I/O error occurred.

Action

Examine the details, and then check whether there is a problem with the OS environment.

KDEA07003-E

The data format is invalid. (file name = *aa....aa*, detail = *bb....bb*)

aa....aa: Store data file name (absolute path)

bb....bb: Maintenance information

Description

The data format is invalid.

The possible causes are as follows:

- It is not a store data file.
- The store data file is damaged.

Action

Make sure the correct store data file was specified.

Alternatively, use another store data file.

KDEA07004-E

A checksum error occurred. (file name = *aa....aa*)

aa....aa: Store data file name (absolute path)

Description

A checksum error occurred.

The store data file might be damaged.

Action

Make sure the correct store data file was specified.

Alternatively, use another store data file.

KDEA07005-E

The input data version is too new. (file name = *aa....aa*, input version = *bb....bb*)

aa....aa: Store data file name (absolute path)

bb....bb: Store data file format version

Description

An attempt was made to read the retrieved store data file in a new format version.

Action

Make sure the correct store data file was specified.

Alternatively, use another store data file.

KDEA07006-E

An unexpected exception occurred. (details = *aa....aa*)

aa....aa: Maintenance information

Description

An unexpected exception occurred.

Action

Contact the customer support center.

KDEA07101-E

The specified cache type cannot be used. (cache name = *aa....aa*, cache type = *bb....bb*)

aa....aa: Cache name

bb....bb: Cache type

Description

The specified cache type cannot be used.

The possible causes are as follows:

- The value of the `eads.cache.type` parameter in the cache properties is invalid.
- The value of the `eads.java.external.heapsize` parameter in the shared properties is too small.

Action

Check and, if necessary, revise the value of the `eads.cache.type` parameter in the cache properties.

If the value of the `eads.cache.type` parameter in the cache properties is correct, increase the value of the `eads.java.external.heapsize` parameter in the shared properties.

KDEA07104-E

The specified cache or cache file was not found. (cache name = *aa....aa*)

aa....aa: Cache name

Description

A specified cache or cache file was not found.

Action

Create the cache.

Alternatively, check if the correct cache file is specified and, if necessary, revise the specification.

KDEA07106-E

An attempt to open a file failed. (file name = *aa....aa*, details = *bb....bb*)

aa....aa: Name (absolute path) of the cache file that was to be opened

bb....bb: Details

Description

An attempt to open a cache file failed.

The possible causes are as follows:

- The cache file to be imported does not exist.
- The specified path is not for a cache file.
- There is no access permission for the cache file.

Action

Examine the details, and then check whether there is a problem with the OS environment.

KDEA07107-E

An I/O error occurred. (file name = *aa....aa*, details = *bb....bb*)

aa....aa: Cache file name (absolute path) (if the cache file name cannot be identified, `null`)

bb....bb: Details

Description

An I/O error occurred.

Action

Examine the details, and then check whether there is a problem with the OS environment.

KDEA07108-E

The data format is invalid. (file name = *aa....aa*, details = *bb....bb*)

aa....aa: Cache file name (absolute path)

bb....bb: Maintenance information

Description

The data format is invalid.

The possible causes are as follows:

- The specified file is not a cache file.
- The cache file is damaged.

Action

Check if the correct cache file is specified and, if necessary, revise the specification.

Alternatively, use another cache file.

KDEA07109-E

A checksum error occurred. (file name = *aa....aa*)

aa....aa: Cache file name (absolute path)

Description

A checksum error occurred.

The cache file might be damaged.

Action

Check if the correct cache file is specified and, if necessary, revise the specification.

Alternatively, use another cache file.

KDEA07110-E

The input data version is too new. (file name = *aa....aa*, input version = *bb....bb*)

aa....aa: Cache file name (absolute path)

bb....bb: Cache file format version

Description

An attempt was made to import a cache file that was obtained using a later version.

Action

Check if the correct cache file is specified and, if necessary, revise the specification.

Alternatively, use another cache file.

KDEA07111-E

A data discrepancy between the cache property file and the cache file was detected. (cache property file name = *aa....aa*, parameter = *bb....bb*, cache property value = *cc....cc*, cache file value = *dd....dd*)

aa....aa: Cache property file name

bb....bb: Parameter name

cc....cc: Value in the cache property file

dd....dd: Value in the cache file

Description

There is a discrepancy in values between the cache property file and the cache file.

Once a cache file has been created, none of the following parameters can be changed in the cache properties:

- `eads.cache.type`
- `eads.cache.disk.filesize`
- `eads.cache.disk.blocksize`

Also, once a cache file has been created, the value of the following parameter in the cache properties cannot be reduced:

- `eads.cache.disk.filenum`

Action

Check if the correct cache file is specified and, if necessary, revise the specification.

Alternatively, use another cache file.

If the value in the cache property file is invalid, check the following parameter values in the cache properties, correct values as necessary, and then restart the EADS server:

- `eads.cache.type`
- `eads.cache.disk.filesize`
- `eads.cache.disk.blocksize`
- `eads.cache.disk.filenum`

KDEA07112-E

A data discrepancy between the cluster property file and the cache file was detected. (cache name = *aa....aa*, type = *bb....bb*, cluster property value = *cc....cc*, cache file value = *dd....dd*)

aa....aa: Cache name

bb....bb: Type

cc....cc: Value in the cluster property file (if the parameter is not specified, `null`)

dd....dd: Value in the cache file

Description

There is a discrepancy in values between the cluster property file and the cache file.

Once a cache file has been created, the parameters cannot be changed in the cluster properties.

The following table provides the cause by type:

Type	Cause
`server number`	The number of EADS servers does not match.
`no server ID`	The EADS server with the specified ID does not exist.
`position`	The position of the EADS server does not match.

Action

Check if the correct cache file is specified and, if necessary, revise the specification.

Alternatively, use another cache file.

If the value in the cluster property file is invalid, check the following parameter values in the cluster properties, correct values as necessary, and then restart the EADS server:

- `eads.node.`*EADS-server-ID*`.address`
- `eads.node.`*EADS-server-ID*`.port`
- `eads.node.`*EADS-server-ID*`.position`

KDEA07113-E

A data discrepancy between the cache and the cache file was detected. (cache name = *aa....aa*, error details = *bb....bb*)

aa....aa: Cache name

bb....bb: Error details

Description

The EADS server could not be restored because there was a discrepancy in values between the cache and the cache file.

Action

Check if the correct cache file is specified and, if necessary, revise the specification.

Alternatively, use another cache file.

KDEA07115-E

A cache file cannot be written to. (cache name = *aa....aa*, details = *bb....bb*)

aa....aa: Cache name

bb....bb: Details

Description

An attempt to import data to a cache file failed because a problem occurred during the import processing.

Action

Determine the cause of the error from the details or from the message output immediately before this message, and then eliminate the problem.

KDEA07116-E

A cache file cannot be read. (cache name = *aa....aa*, details = *bb....bb*)

aa....aa: Cache name

bb....bb: Details

Description

An attempt to import a cache file failed.

The possible causes are as follows:

- A problem occurred while the cache property file was being imported.
- A problem occurred while the cache file was being imported
- There are too many or too few cache files.

Action

Determine the cause of the error from the details or from the message that was output immediately before this message, and then eliminate the problem.

If the number of cache files was too large or too small when the `ezstart -r` or `ezserver -r` command was executed, you might be able to restore the cache file by executing the `deleteecf -l` command and then re-executing the `ezstart -r` or `ezserver -r` command.

KDEA07117-E

> A data discrepancy between the shared property file and the cache file was detected. (cache name = *aa....aa*, parameter = *bb....bb*, shared property value = *cc....cc*, cache file value = *dd....dd*)

aa....aa: Cache name

bb....bb: Parameter name

cc....cc: Value in the shared property file

dd....dd: Value in the cache file

Description

There is a discrepancy between a value in the shared property file and the value in the cache file.

Once a cache file has been created, the following parameters cannot be changed in the shared properties:

- `eads.cache.key.maxsize`
- `eads.replication.factor`

Action

Check if the correct cache file is specified and, if necessary, revise the specification.

Alternatively, use another cache file.

If the value in the shared property file is invalid, check the following parameter values in the shared properties, correct the value, and then restart the EADS server:

- `eads.cache.key.maxsize`
- `eads.replication.factor`

KDEA07120-E

> An I/O error occurred. (directory = *aa....aa*, details = *bb....bb*)

aa....aa: Directory name (absolute path)

bb....bb: Details

Description

An I/O error occurred.

Action

Examine the details, and then check whether there is a problem with the OS environment.

Check especially for the following problems:

- Required directories have not been created.
- Directories cannot be created.

KDEA07121-E

> A data discrepancy between the cache property file and the cache was detected. (cache property file name = *aa....aa*, parameter = *bb....bb*, cache property value = *cc....cc*, cache value = *dd....dd*)

aa....aa: Cache property file name

bb....bb: Parameter name

cc....cc: Value in the cache property file

dd....dd: Cache value

Description

There is a discrepancy between a value in the cache property file and a cache value.

Once a cache file has been created, none of the following parameters can be changed in the cache properties:

- `eads.cache.type`
- `eads.cache.disk.filesize`
- `eads.cache.disk.filenum`
- `eads.cache.disk.blocksize`

Action

Check the following parameter values in the cache properties, correct the values, and then restart the EADS server:

- `eads.cache.type`
- `eads.cache.disk.filesize`
- `eads.cache.disk.filenum`
- `eads.cache.disk.blocksize`

KDEA07122-E

The area for writing the cache data files is insufficient. (cache name = *aa....aa*)

aa....aa: Cache name

Description

A space shortage occurred in the area for importing cache data files.

Action

Increase the value of the `eads.cache.disk.filenum` parameter in the cache properties according to the procedure described in *11.4.1 How to change the properties*.

KDEA07123-E

An I/O error occurred. (cache name = *aa....aa*, details = *bb....bb*)

aa....aa: Cache name

bb....bb: Details

Description

An I/O error occurred.

Action

Determine the cause of the error from the exception logs and then eliminate the problem.

KDEA07124-E

An unexpected exception occurred. (cache name = *aa....aa*, details = *bb....bb*)

aa....aa: Cache name

bb....bb: Maintenance information

Description

An unexpected exception occurred.

Action

Contact the customer support center.

KDEA07125-I

A parameter different from the cache file value will be used. (cache property file name = *aa....aa*, parameter = *bb....bb*, cache property value = *cc....cc*, cache file value = *dd....dd*)

aa....aa: Cache property file name

bb....bb: Parameter name

cc....cc: Value in the cache property file

dd....dd: Value in the cache file

Description

A parameter value that differs from the value in the cache file was used.

KDEA07126-W

The loading of the cache property file failed. (cache property file name = *aa....aa*)

aa....aa: Cache property file name

Description

An attempt to load a cache property file failed because an error occurred while the cache property file was being loaded.

Action

Check the messages that were output immediately before this message.

KDEA07127-E

The loading of the cache property file failed. (cache property file name = *aa....aa*)

aa....aa: Cache property file name

Description

An attempt to load a cache property file failed because an error occurred while the cache property file was being loaded.

Action

Check the messages that were output immediately before this message.

KDEA07128-E

The compaction of cache data files failed. (cache name = *aa....aa*, range ID = *bb....bb*, error details = *cc....cc*)

aa....aa: Cache name

bb....bb: Range ID

cc....cc: Error details

Description

An attempt to perform compaction on the cache data files failed.

Action

Cache files might have been damaged.

Delete the cache files with the `eztool deleteecf` command, and then execute one of the following commands to restore the EADS server again:

- `ezstart -r` command
- `ezserver -r` command

KDEA07201-E

Some data of cache might be inconsistent. (details = *aa....aa*)

aa....aa: Details

Description

Some data in cache might be inconsistent.

Because an attempt to resume the cache failed, EADS server restoration processing might have failed.

Action

Check the cache.

If necessary, delete the cache with the `eztool deletecache` command, and then restore the EADS server.

KDEA07202-E

The operation and cache type combination is invalid. (operation = *aa....aa*, cache name = *bb....bb*, cache type = *cc....cc*)

aa....aa: Operation

bb....bb: Cache name

cc....cc: Cache type

Description

The combination of the operation and the cache type is invalid.

An attempt to add an EADS server might have failed because there are caches other than memory caches.

Action

Check the cache.

If necessary, delete the cache with the `eztool deletecache` command, and then add the EADS server.

22.7 KDEA08000 to KDEA09999

This section describes messages `KDEA08000` to `KDEA09999` and explains what actions to take in response to each message.

KDEA08001-I

The command will now start. (subcommand = *aa....aa*, parameter = *bb....bb*)

aa....aa: Subcommand

bb....bb: Options and arguments

Description

The command is starting.

KDEA08002-I

The command will now end.

Description

The command has finished executing.

KDEA08006-E

Log initialization failed. (log directory = *aa....aa*)

aa....aa: Path name of the log output destination

Description

An attempt to initialize the log failed.

Action

Check if the path name of the log file output location specified in the `eads.command.logger.dir` parameter in the command properties is correct.

If an attempt to initialize the log library failed, re-execute the command when the EADS server's workload is low.

KDEA08007-E

The specified option is invalid. (error details = *aa....aa*) For more information, use "eztool -h".

aa....aa: Error details

Description

The specified command option is invalid.

For details about the format of the command, execute the command `eztool -h`.

Action

Check the command options.

KDEA08008-E

The specified option is duplicated. (option = *aa....aa*) *aa....aa*: Command option

Description

The specified command option is duplicated.

Action

Check the command options.

KDEA08009-E

The specified subcommand is invalid. (subcommand = *aa....aa*) For more information, use "eztool -h".

aa....aa: Invalid subcommand

Description

The specified subcommand is invalid.

For details about the format of the command, execute the command `eztool -h`.

Action

Check the subcommand.

KDEA08010-E

The specified parameter is invalid. (parameter = *aa....aa*) For more information, use "eztool -h".

aa....aa: Invalid command argument

Description

The specified command argument is invalid.

For details about the format of the command, execute the command `eztool -h`.

Action

Check the command arguments.

KDEA08011-E

An attempt to connect to a server failed. (server = *aa....aa*)

aa....aa: The host name (or IP address) and port number of the connection-target EADS server

Description

An attempt to connect to the EADS server failed.

Action

Check the server properties of the connection-target EADS server.

Check whether the EADS server to be connected has been started.

If the failure occurred on an EADS server in a cluster, wait until the down EADS server is detected, and then retry the operation. For details about how long it takes to detect a down EADS server, see *9.3.2(1) Sending heartbeats and checking for live servers*.

KDEA08013-E

The server is not in a state in which commands can be executed. (subcommand = *aa....aa*, error details = *bb....bb*) *aa....aa*: Subcommand

bb....bb: Error details (list of EADS servers on which commands cannot be executed, and their statuses)

Description

The EADS server is not in a status in which commands can be executed.

Action

Check the status of the EADS servers by executing the command `eztool status`.

Retry the operation from a status in which commands can be executed.

KDEA08014-E

An attempt to lock the server failed. (server = *aa....aa*, error details = *bb....bb*)

aa....aa: The host name (or IP address) and port number of the EADS server where the lock attempt failed

bb....bb: Error details

Description

A lock attempt failed.

Action

Check whether other commands are running by executing the command `eztool status -v`.

If the lock status of the EADS server is `lock` (locked) even though commands have finished executing, unlock it by executing the command `eztool unlock`.

If the cause of the error is unknown, contact the customer support center.

KDEA08015-E

A server error occurred during the execution of a command. (server = *aa....aa*, error details = *bb....bb*)

aa....aa: The host name (or IP address) and port number of the EADS server where the error occurred

bb....bb: Error details

Description

An error occurred on the EADS server during execution of the command.

Action

Check the error detail information.

Check the messages for the EADS server to find the error that occurred on the EADS server.

If the cause of the error is unknown, contact the customer support center.

KDEA08016-E

An attempt to unlock the server failed. (server = *aa....aa*, error details = *bb....bb*)

aa....aa: The host name (or IP address) and port number of the EADS server on which the unlock attempt failed

bb....bb: Error details

Description

An unlock attempt failed.

Action

Check whether other commands are running by executing the command `eztool status -v`.

If the lock status of the EADS server is `lock` (locked) even though commands have finished executing, unlock it by executing the command `eztool unlock`.

If the cause of the error is unknown, contact the customer support center.

KDEA08018-E

The server connection timed out. (server = *aa....aa*, timeout value = *bb....bb*)

aa....aa: The host name (or IP address) and port number of the EADS server that timed out

bb....bb: Timeout period

Description

The connection to the EADS server timed out.

Action

Check the status of the EADS servers by executing the command `eztool status`.

If the failure occurred on an EADS server in a cluster, wait until the down EADS server is detected, and then retry the operation. For details about how long it takes to detect a down EADS server, see *9.3.2(1) Sending heartbeats and checking for live servers*.

KDEA08019-E

The command timed out. (timeout value = *aa....aa*)

aa....aa: Timeout period

Description

The command timed out.

Action

Check the status of the EADS servers by executing the command `eztool status`.

Check the values listed below, change values as necessary, and then re-execute the command:

- Timeout value specified in a command argument
- `eads.command.common.execution.timeout` parameter in the command properties
- `eads.command.`*`subcommand-name`*`.execution.timeout` parameter

In the case of the `eztool resume` command, data might no longer be consistent.

If you continue operation in such a status, data might become corrupted or lost. To prevent this, take the following steps:

1. Execute the `eztool status -v` command to check the cluster status.
2. Execute the `eztool unlock` command to release any lock.
3. Execute the `eztool listcache` command to check the list of caches.
4. If cache resume processing has failed, execute the `eztool deletecache` command to delete the caches.
5. If EADS servers are isolated, restore them.
6. Perform the cache resume processing again.

KDEA08020-E

An unexpected error occurred during command execution. (error details = *aa....aa*)

aa....aa: Error details

Description

An unexpected error occurred during execution of the command.

Action

Check the error detail information.

If the cause of the error is unknown, contact the customer support center.

KDEA08023-I

The status has already changed. (server = *aa....aa*)

aa....aa: The host name (or IP address) and port number of the EADS server

Description

The status of the EADS server has already changed.

KDEA08024-W

No matching store data files were found. (server = *aa....aa*, warning details = *bb....bb*)

aa....aa: The host name (or IP address) and port number of the EADS server

bb....bb: Warning details

Description

No matching store data files were found.

Action

Check the specified store data file key.

Make sure the store data file exists at the store data file output destination.

This warning is displayed when you perform tasks such as adding an EADS server to a cluster or moving store data files due to a change in the directory structure. Check the warning details, and if there is no problem, this warning can be ignored.

KDEA08025-E

The specified subcommand requires a value. (subcommand = *aa....aa*) For more information, use "eztool -h".

aa....aa: Subcommand for which no argument was specified

Description

The specified subcommand requires an argument.

For details about the format of the command, execute the command `eztool -h`.

Action

Check the arguments to the subcommand.

KDEA08026-W

The specified cache already exists. (cache name = *aa....aa*)

aa....aa: Cache name

Description

The specified cache name already exists.

Action

Check whether the correct cache name was specified.

KDEA08027-W

A connection cannot be established because the cluster is offline. (server = *aa....aa*)

aa....aa: The host name (or IP address) and port number of the EADS server

Description

A connection cannot be established because the EADS server is not joined to a cluster.

Action

Check the status of the cluster by executing the command `eztool status`.

KDEA08029-E

The number of store data file generations has exceeded the limit. (generation count = *aa....aa*, limit = *bb....bb*)

aa....aa: Number of generations

bb....bb: Upper limit on the number of generations

Description

The number of store data file generations has exceeded the limit.

Action

Take one of the following corrective actions:

- Execute the `eztool deleteesd` command to delete unneeded store data files from the cluster.
- Export data to another directory by executing the command `eztool export -d`.
- Save the store data file to a directory of your choice.

KDEA08030-E

No generation-managed store data files were found.

Description

No generation-managed store data files were found.

Action

Check the store data files by executing the command `eztool listesd`.

KDEA08031-W

The specified cache name does not exist. (cache name = *aa....aa*)

aa....aa: Cache name

Description

The specified cache was not found.

Action

Check whether the correct cache name was specified.

KDEA08032-E

Cache creation failed. (cache name = *aa....aa*, error details = *bb....bb*)

aa....aa: Cache name

bb....bb: Error details

Description

An attempt to create a cache failed.

Action

Make sure the limit on the number of caches (16) has not been reached.

Check whether the correct cache name was specified.

Check the error detail information.

KDEA08033-E

Importing failed. (store date file key = *aa....aa*, error details = *bb....bb*)

aa....aa: Store data file key

bb....bb: Error details

Description

An attempt to read data failed.

Action

Check the specified store data file key.

Check the error detail information.

Check the immediately preceding messages for the EADS server.

Check the status of the EADS servers by executing the command `eztool status`.

KDEA08034-E

Data addition failed. (cache name = *aa....aa*, key = *bb....bb*, error details = *cc....cc*)

aa....aa: Cache name

bb....bb: Key

cc....cc: Error details

Description

Data addition failed.

Action

Check the cache name.

Check the error detail information.

Check the status of the EADS servers by executing the command `eztool status`.

KDEA08035-E

Data acquisition failed. (cache name = *aa....aa*, key = *bb....bb*, error details = *cc....cc*)

aa....aa: Cache name

bb....bb: Key

cc....cc: Error details

Description

An attempt to acquire data failed.

Action

Check the cache name.

Check the error detail information.

Check the status of the EADS servers by executing the command `eztool status`.

KDEA08036-E

Data deletion failed. (cache name = *aa....aa*, key = *bb....bb*, error details = *cc....cc*)

aa....aa: Cache name

bb....bb: Key

cc....cc: Error details

Description

An attempt to delete data failed.

Action

Check the cache name.

Check the error detail information.

Check the status of the EADS servers by executing the command `eztool status`.

KDEA08037-E

The cluster property file is incorrect. (error details = *aa....aa*)

aa....aa: Error details

Description

The properties related to the cluster configuration are incorrect.

Action

Make sure the cluster property file exists.

Check the error detail information.

Make sure the following cluster property parameters are correct:

- `eads.node.`*EADS-server-ID*`.address`
- `eads.node.`*EADS-server-ID*`.port`
- `eads.node.`*EADS-server-ID*`.position`

KDEA08046-W

The specified function name does not exist. (function name = *aa....aa*)

aa....aa: Name of the user function

Description

The specified user function name was not found.

Action

Make sure that the user function name is correct.

KDEA08049-E

No matching store data file was found. (store data file key = *aa....aa*)

aa....aa: Store data file key

Description

No matching store data files were found.

Action

Check the specified store data file key.

Make sure the store data file exists at the store data file output destination.

This message is displayed when you perform tasks such as moving store data files due to a change in the directory structure. Check the error details, and if there is no problem, this warning can be ignored.

KDEA08050-E

The specified store data file already exists. (store data file key = *aa....aa*)

aa....aa: Store data file key

Description

The specified store data file name already exists.

Action

Check if the specified store data file key is correct.

KDEA08051-E

Processing to open the cluster failed. (error details = *aa....aa*)

aa....aa: Error details

Description

An attempt to release the cluster from closed status failed.

Action

Check the error detail information.

KDEA08052-E

Processing to isolate the server failed. (error details = *aa....aa*)

aa....aa: Error details

Description

An attempt to isolate an EADS server failed.

Action

Check the error detail information.

Execute the `eztool status` command to check the status of the EADS server on which the command was executed.

- If the EADS server on which the command was executed is not isolated

  Verify that the cluster's status is not partially available (not `PARTIALLY_AVAILABLE`), and then re-execute the command.

- If the EADS server on which the command was executed is isolated and the `eztool isolate --stop` command was executed

  Re-execute the command to shut down the EADS server process.

KDEA08053-E

The wait for completion of execution failed. (error details = *aa....aa*)

aa....aa: Error details

Description

The wait for a command to finish executing failed.

Action

- For the `eztool createcache` or `eztool deletecache` command

  Execute the `eztool listcache` command to check the number of caches.

  Re-execute the command.

- For the `eztool stop` or `eztool isolate --stop` command

  Execute the `eztool status` command to check the status of the EADS server.

  Also check the process status.

  Re-execute the command.

KDEA08054-I

The store data file was imported. (store data file key = *aa....aa*)

aa....aa: Store data file key

Description

A store data file was imported.

KDEA08055-I

The store data file was exported. (store data file key = *aa....aa*)

aa....aa: Store data file key

Description

A store data file was exported.

KDEA08056-E

A property file cannot be read. (file path = *aa....aa*)

aa....aa: Path name to the property file

Description

The property file cannot be read.

The possible causes are as follows:

- The property file cannot be opened.
- The path name points to a directory rather than to a file.

Action

Check and, if necessary, revise the path name to the property file.

Determine the cause of the error, and then eliminate it.

KDEA08057-W

The obtained cluster information might be old.

Description

The obtained cluster information might be outdated.

Only isolated EADS servers might be running.

Action

Execute the `eztool status` command to check the status of the cluster and the EADS servers.

KDEA08058-E

Execution of a function failed. (cache name = *aa....aa*, function name = *bb....bb*, error details = *cc....cc*)

aa....aa: Cache name

bb....bb: Name of the user function

cc....cc: Error details

Description

An attempt to execute a user function failed.

Action

Check the cache name, user function name, and error detail information.

Execute the `eztool status` command to check the EADS server status.

KDEA08059-I

Stopping the compaction of cache data files is already reserved.

Description

A request to stop compaction of cache data files has already been queued.

KDEA08060-E

The resumption of cache files failed. (error details = *aa....aa*)

aa....aa: Error details

Description

An attempt resume a cache failed.

Action

Check the error detail information.

KDEA08061-E

The compaction of cache data files failed. (error details = *aa....aa*)

aa....aa: Error details

Description

An attempt to perform compaction on cache data files failed.

Action

Check the error detail information.

KDEA08062-I

The compaction of cache data files was stopped.

Description

Compaction of cache data files was stopped.

KDEA08063-I

The compaction of cache data files is not executing.

Description

Compaction has not been performed on cache data files.

KDEA08064-I

The cache files were deleted.

Description

Cache files were deleted.

KDEA08065-E

Deletion of cache files failed. (error details = *aa....aa*)

aa....aa: Error details

Description

An attempt to delete cache files failed.

Action

Check the error detail information.

KDEA08066-W

No matching cache file was found. (server = *aa....aa*, warning details = *bb....bb*)

aa....aa: EADS server's host name (or IP address) and port number

bb....bb: Warning details

Description

The corresponding cache file was not found.

Action

Check the cache file storage location specified in the cache property file.

This warning is displayed when the corresponding cache file has been moved or deleted, for example, due to a change to the directory configuration or because the contents of the cache property file have been changed. Check the warning details.

KDEA08067-E

No matching cache file was found.

Description

The corresponding cache file was not found.

Action

Use the `eztool listecf` command to check whether the cache file exists.

Check the contents of the cache property file.

Check the cache file storage location specified in the cache property file.

KDEA08068-E

The importing of cache files failed. (error details = *aa....aa*)

aa....aa: Error details

Description

An attempt to import cache files failed.

Action

Check the error detail information.

KDEA08070-E

Creation of cache files failed. (cache name = *aa....aa*, error details = *bb....bb*)

aa....aa: Cache name

bb....bb: Error details

Description

An attempt to create cache files failed.

Action

Check the error detail information.

KDEA08071-W

Cache files do not exist because the type of the specified cache is MemoryCache.

Description

Cache files do not exist because the specified cache type is memory cache.

KDEA08072-W

The cache already exists. (cache name = *aa....aa*)

aa....aa: Cache name

Description

The cache already exists.

Action

Check the cache.

If necessary, delete the cache with the `eztool deletecache` command, and then re-execute the command.

KDEA08073-I

The resumption of cache started. (cache name = *aa....aa*)

aa....aa: Cache name

Description

Cache resume has started.

KDEA08074-I

The resumption of cache finished. (cache name = *aa....aa*)

aa....aa: Cache name

Description

Cache resume has finished.

KDEA08075-W

The resumption of cache files did not run, because the cache to resume target does not exist.

Description

An attempt to resume cache files was not made because the target cache did not exist.

Action

Check the cache.

If necessary, delete the cache with the `eztool deletecache` command, and then re-execute the command.

KDEA08076-E

Some data might be inconsistent, because resumption of cache failed. Therefore a lock was not released.

Description

Some data might be inconsistent because a cache resume attempt failed. The lock was not released.

Action

Data might no longer be consistent. If you continue operation in such a status, data might become corrupted or lost. Take the following actions:

1. Execute the `eztool status -v` command to check the cluster status.
2. Execute the `eztool unlock` command to release any lock.
3. Execute the `eztool listcache` command to check the list of caches.
4. Execute the `eztool deletecache` command to delete the caches whose resume processing failed.
5. If EADS servers are isolated, restore them.
6. Perform the cache resume processing again.

KDEA08077-W

The compaction of cache data files was not executed. (warning details = *aa....aa*)

aa....aa: Warning details

Description

Compaction of cache data files was not performed.

Action

Check the warning details.

If necessary, reduce the threshold and increase the number of unused files, and then re-execute the command.

KDEA08078-W

The compaction of cache data files could not reach to the specified unused file count. (minimum unused file count = *aa....aa*)

aa....aa: Minimum number of unused files

Description

Compaction of cache data files could not yield the specified number of unused files.

Action

If necessary, reduce the threshold and then re-execute the command.

KDEA08079-I

Exporting was not executed because no memory cache exist on the server.

Description

Data was not exported because there was no memory cache on the EADS server.

KDEA08081-E

Data addition failed. (cache name = *aa....aa*, error details = *bb....bb*)

aa....aa: Cache name

bb....bb: Error details

Description

An attempt to add data failed.

Action

Check the cache name.

Check the error detail information.

Check the EADS server's status.

KDEA08082-E

Data acquisition failed. (cache name = *aa....aa*, error details = *bb....bb*)

aa....aa: Cache name

bb....bb: Error details

Description

An attempt to acquire data failed.

Action

Check the cache name.

Check the error detail information.

Check the EADS server's status.

KDEA08083-E

Data deletion failed. (cache name = *aa....aa*, error details = *bb....bb*)

aa....aa: Cache name

bb....bb: Error details

Description

An attempt to delete data failed.

Action

Check the cache name.

Check the error detail information.

Check the EADS server's status.

KDEA08084-E

The command cannot be executed because the command version and server version are different. (command version = *aa....aa*, server version = *bb....bb*)

aa....aa: Command version

bb....bb: EADS server version

Description

The command cannot be executed because the command's version does not match the EADS server's version.

Action

Execute the command whose version matches the EADS server's version.

KDEA08085-W

No range for the server ID specified in the server ID specification group was found. (server = *aa....aa*, server ID = *bb....bb*)

aa....aa: EADS server name

bb....bb: EADS server ID

Description

The range for the EADS server ID specified in the EADS server ID specified group was not found.

Action

Check whether the EADS server ID conversion rule specified in the `eztool import` or `eztool importecf` command is correct.

Check the cluster configuration and the EADS server ID of each EADS server.

KDEA08086-E

The execution of the command failed because the cluster configuration was changed while the command was being executed.

Description

An attempt to execute the command failed because the cluster configuration was changed during command execution.

Action

Re-execute the command after the processing that changes the cluster configuration (such as scale-out processing) has finished.

KDEA08401-E

Log initialization failed. (log directory = *aa....aa*) *aa....aa*: Path name of the log output destination

Description

An attempt to initialize the log failed.

Action

Check if the path name of the log file output location specified in the `eads.logger.dir` parameter in the server properties is correct.

Note that this message is not output to the log file because it is issued from a script.

KDEA08402-E

> The server directory is invalid. (directory = *aa....aa*)

aa....aa: Management directory

Description

The management directory is invalid.

Action

Make sure that the management directory is correct.

Note that this message is not output to the log file because it is issued from a script.

KDEA08403-E

> Startup of the server failed.

Description

Startup of the EADS server failed.

Action

Make sure that the `ezserver` file is an executable file.

In addition, make sure that the log output destination and log file can be accessed.

Note that this message is not output to the log file because it is issued from a script.

KDEA08404-E

> The directory name contains invalid characters. (directory = *aa....aa*)

aa....aa: Directory name

Description

The directory name is invalid. The directory name contains invalid characters.

Action

Use alphanumeric characters (0 to 9, A to Z, a to z), underscores (_), and forward slashes (/) in the directory name.

In addition, do not specify a relative path name that contains a period for the directory name.

Note that this message is not output to the log file because it is issued from a script.

KDEA08405-E

> The server process was not found. (server = *aa....aa*)

aa....aa: EADS server name (management directory name)

Description

The EADS server process was not found.

Action

Make sure that the EADS server has been started.

Note that this message is not output to the log file because it is issued from a script.

KDEA08406-E

A required program was not found. (directory = *aa....aa*)

aa....aa: EADS server name (management directory name)

Description

A required program was not found.

Action

Make sure that the product has been properly installed.

Make sure that the location of the management directory is correct.

Note that this message is not output to the log file because it is issued from a script.

KDEA08407-E

The specified option is invalid. (error details = *aa....aa*)

aa....aa: Error details

Description

The specified option is invalid.

Action

Check the options.

Note that this message is not output to the log file because it is issued from a script.

KDEA08408-E

The server process already exists. (server = *aa....aa*)

aa....aa: EADS server name (management directory name)

Description

The EADS server is already running.

Action

Check whether the EADS server is running.

Note that this message is not output to the log file because it is issued from a script.

KDEA08409-E

An attempt to lock the server failed.

Description

An attempt to place a lock failed.

Action

Execute the `eztool status -v` command to check whether another command is executing.

If the command has terminated, but the EADS server is still locked (`lock` status), execute the `eztool unlock` command to unlock the EADS server.

If the cause of the error is unknown, contact the customer support center.

Note that this message is not output to the log file because it is issued from a script.

KDEA08410-E

The specified options cannot be specified at the same time. (option names = *aa....aa*, *bb....bb*)

aa....aa: option

bb....bb: option

Description

Specified options cannot be specified together.

Action

Check the options.

Note that this message is not output to the log file because it is issued from a script.

KDEA08411-E

File creation failed. (file name = *aa....aa*, error details = *bb....bb*)

aa....aa: File name

bb....bb: Error details

Description

An attempt to create a file failed.

Action

Check the error detail information.

Note that this message is not output to the log file because it is issued from a script.

KDEA08412-W

An attempt to unlock the server failed.

Description

An attempt to release a lock failed.

Action

Execute the `eztool status -v` command to check whether another command is executing.

If the command has terminated, but the EADS server is still locked (`lock` status), execute the `eztool unlock` command to unlock the EADS server.

Note that this message is not output to the log file because it is issued from a script.

KDEA08413-W

The directory of cache files is invalid. (cache properties name = *aa....aa*, warning details = *bb....bb*)

aa....aa: Cache property file name

bb....bb: Warning details

Description

The directory specified in the cache property file is invalid.

Action

Check the warning details.

If necessary, check whether the directory specified in the cache property file is correct.

Note that this message is not output to the log file because it is issued from a script.

KDEA08414-W

The directory of cache files is invalid. (cache properties name = *aa....aa*, warning details = *bb....bb*)

The parameter is invalid. The default value will be used. (parameter = *aa....aa*, value = *bb....bb*, default value = *cc....cc*)

aa....aa: Parameter name

bb....bb: Specified value

cc....cc: Default value

Description

An invalid value was specified in a parameter. The system will use the default value.

Action

An invalid value is specified in the parameter.

If necessary, check whether the value specified in the parameter is correct.

Note that this message is not output to the log file because it is issued from a script.

KDEA08415-E

The specified option is duplicated. (option = *aa....aa*)

aa....aa: Option name

Description

A specified option is duplicated.

Action

Check whether the option is duplicated.

Note that this message is not output to the log file because it is issued from a script.

KDEA08416-E

The specified parameter is invalid. (parameter = *aa....aa*)

aa....aa: Argument

Description

A specified argument is invalid.

Action

Check whether an invalid argument is specified.

Note that this message is not output to the log file because it is issued from a script.

KDEA08417-W

Deletion of a file failed. (warning details = *aa....aa*)

aa....aa: Warning details

Description

An attempt to delete a file failed.

Action

Check the warning details.

If necessary, manually delete the file whose deletion processing failed.

Note that this message is not output to the log file because it is issued from a script.

KDEA08418-E

An attempt to read the file failed. (file name = *aa....aa*, error details = *bb....bb*)

aa....aa: File name

bb....bb: Error details

Description

An attempt to read a file failed.

Action

Check whether the file exists or check the file permissions.

Note that this message is not output to the log file because it is issued from a script.

KDEA08419-W

The file might not be created. (file name = *aa....aa*, warning details = *bb....bb*)

aa....aa: File name

bb....bb: Warning details

Description

An attempt to create a file might have failed.

Action

Check the warning details.

Check whether the file has been created successfully. Alternatively, re-execute the command.

Note that this message is not output to the log file because it is issued from a script.

KDEA08501-I

The collection of statistics started.

Description

The collection of statistics has started.

KDEA08502-I

The collection of statistics stopped.

Description

Statistics collection has stopped.

KDEA08505-I

Export processing started. (store data file path = *aa....aa*, management client = *bb....bb*)

aa....aa: Store data file output destination path name

bb....bb: IP address from which the command was executed

Description

Exporting of data has started.

KDEA08506-I

Import processing started. (store data directory = *aa....aa*, store data file key = *bb....bb*, management client = *cc....cc*)

aa....aa: Store data file storage destination path name

bb....bb: Store data file key

cc....cc: IP address from which the command was executed

Description

Importing of data has started.

KDEA08507-I

A cache was created. (cache name = *aa....aa*, management client = *bb....bb*)

aa....aa: Created cache name

bb....bb: IP address from which the command was executed

Description

A cache was created.

KDEA08508-I

A cache was deleted. (cache name = *aa....aa*, management client = *bb....bb*)

aa....aa: Deleted cache name

bb....bb: IP address from which the command was executed

Description

A cache was deleted.

KDEA08510-I

Open processing finished. (management client = *aa....aa*)

aa....aa: IP address from which the command was executed

Description

The closed EADS server was opened.

KDEA08512-E

Initialization of the management service failed. (error details = *aa....aa*)

aa....aa: Error details

Description

Initialization of the management service failed.

Action

Check the error details, and then restart the EADS server.

In addition, check the server property parameter `eads.admin.operation.port`.

Check the directories and files to which statistics are output.

KDEA08513-E

Output of statistics failed. (statistics file path = *aa....aa*, error details = *bb....bb*)

aa....aa: Statistics file path name

bb....bb: Error details

Description

An attempt to output the statistics file failed.

Action

Check the output destination for the statistics file.

KDEA08516-E

Exporting failed. (store data file path = *aa....aa*, management client = *bb....bb*, error details = *cc....cc*)

aa....aa: Store data file output destination path name

bb....bb: IP address from which the command was executed

cc....cc: Error details

Description

An attempt to export data failed.

Action

Check the store data file key.

Check the error detail information.

Check the messages that were output immediately before this one.

Check the status of the EADS servers by executing the command `eztool status`.

KDEA08517-E

Importing failed. (store data directory = *aa....aa*, store date file key = *bb....bb*, management client = *cc....cc*, error details = *dd....dd*)

aa....aa: Store data file storage destination path name

bb....bb: Store data file key

cc....cc: IP address from which the command was executed

dd....dd: Error details

Description

An attempt to read data failed.

Action

Check the store data file key.

Check the error detail information.

Check the messages that were output immediately before this one.

Check the status of the EADS servers by executing the command `eztool status`.

KDEA08518-E

Creation of a cache failed. (cache name = *aa....aa*, management client = *bb....bb*, error details = *cc....cc*)

aa....aa: Name of cache whose creation was attempted

bb....bb: IP address from which the command was executed

cc....cc: Error details

Description

An attempt to create a cache failed.

Action

Check the cache name.

Check the error detail information.

Check the messages that were output immediately before this one.

Check the status of the EADS servers by executing the command `eztool status`.

KDEA08519-E

Deletion of a cache failed. (cache name = *aa....aa*, management client = *bb....bb*, error details = *cc....cc*)

aa....aa: Name of the cache whose deletion was attempted *bb....bb*: IP address from which the command was executed

cc....cc: Error details

Description

An attempt to delete a cache failed.

Action

Check the cache name.

Check the error detail information.

Check the messages that were output immediately before this one.

Check the status of the EADS servers by executing the command `eztool status`.

KDEA08530-E

The server is not in a state in which commands can be executed. (subcommand = *aa....aa*, management client = *bb....bb*, error details = *cc....cc*)

aa....aa: Subcommand

bb....bb: IP address from which the command was executed

cc....cc: Error details

Description

The EADS server is not in a state in which commands can be executed.

Action

Check the status of the EADS server, command, and cluster.

KDEA08531-I

The status has already been changed. (subcommand = *aa....aa*, management client = *bb....bb*, details = *cc....cc*)

aa....aa: Subcommand

bb....bb: IP address from which the command was executed

cc....cc: Details

Description

The status of the EADS server has already been changed.

KDEA08532-I

Close processing finished. (management client = *aa....aa*)

aa....aa: IP address from which the command was executed

Description

The EADS server was closed.

KDEA08533-I

Export processing finished. (store data file path = *aa....aa*, management client = *bb....bb*)

aa....aa: Store data file output destination path name

bb....bb: IP address from which the command was executed

Description

Data export has been completed.

KDEA08534-I

Import processing finished. (store data directory = *aa....aa*, store data file key = *bb....bb*, management client = *cc....cc*)

aa....aa: Store data file storage destination path name

bb....bb: Store data file key

cc....cc: IP address from which the command was executed

Description

Data import has been completed.

KDEA08535-E

An unexpected error occurred. (error details = *aa....aa*)

aa....aa: Error details

Description

An unexpected error occurred.

Action

Check the error detail information.

If the cause of the error is unknown, contact the customer support center.

KDEA08537-I

The importing of the store data file started. (store data file path = *aa....aa*)

aa....aa: Store data file storage destination path name

Description

The importing of data has started.

KDEA08538-I

The importing of the store data file finished. (store data file path = *aa....aa*)

aa....aa: Store data file storage destination path name

Description

Data import has been completed.

KDEA08539-E

The importing of the store data file failed. (store data file path = *aa....aa*, error details = *bb....bb*)

aa....aa: Store data file storage destination path name

bb....bb: Error details

Description

An attempt to read data failed.

Action

Check the error detail information.

Make sure the store data file is correct.

KDEA08545-W

The specified cache name already exists. (cache name = *aa....aa*, management client = *bb....bb*)

aa....aa: Cache name

bb....bb: IP address from which the command was executed

Description

The specified cache name already exists.

Action

Make sure the cache name is correct.

KDEA08547-W

The specified cache name does not exist. (cache name = *aa....aa*, management client = *bb....bb*)

aa....aa: Cache name

bb....bb: IP address from which the command was executed

Description

The specified cache does not exist.

Action

Make sure the cache name is correct.

KDEA08548-I

The store data file was deleted. (store data file path = *aa....aa*)

aa....aa: Store data file output destination path name

Description

The store data file was deleted.

KDEA08549-E

Deletion of the store data file failed. (store data file name = *aa....aa*, error details = *bb....bb*) *aa....aa*: Store data file name

bb....bb: Error details

Description

An attempt to delete store data file failed.

Action

Check the messages that were output immediately before this one.

Check the status of the EADS servers by executing the command `eztool status`.

KDEA08569-E

The specified cache name is invalid. (cache name = *aa....aa*, management client = *bb....bb*)

aa....aa: Cache name

bb....bb: IP address from which the command was executed

Description

The specified cache name is invalid.

Action

Check whether the correct cache name was specified.

KDEA08570-E

The number of caches exceeds the limit. (cache count = *aa....aa*, limit = *bb....bb*, management client = *cc....cc*)

aa....aa: Number of caches

bb....bb: Upper limit on the number of caches

cc....cc: IP address from which the command was executed

Description

The number of caches exceeds the upper limit.

Action

Execute the `eztool deletecache` command to delete unneeded caches.

KDEA08571-E

The specified directory name is invalid. (store data directory name = *aa....aa*, management client = *bb....bb*)

aa....aa: Store data file storage destination path name

bb....bb: IP address from which the command was executed

Description

The specified directory name is invalid.

Action

Check if the specified directory name is correct.

KDEA08572-E

The specified store data file name is invalid. (store data file key = *aa....aa*, management client = *bb....bb*)

aa....aa: Store data file key

bb....bb: IP address from which the command was executed

Description

The specified store data file name is invalid.

Action

Check if the specified store data file key is correct.

KDEA08575-E

Acquisition of the store data failed. (management client = *aa....aa*, error details = *bb....bb*)

aa....aa: IP address from which the command was executed

bb....bb: Error details

Description

An attempt to acquire store data failed.

Action

Make sure the store data file exists at the store data file output destination.

Check the output destination and access permissions for the store data file.

KDEA08576-I

Processing to isolate the server finished. (management client = *aa....aa*)

aa....aa: IP address from which the command was executed

Description

The EADS server has been isolated.

KDEA08577-E

Processing to isolate the server failed. (management client = *aa....aa*, error details = *bb....bb*)

aa....aa: IP address from which the command was executed

bb....bb: Error details

Description

An attempt to isolate the EADS server failed.

Action

Check the error detail information.

Check the status of other EADS servers.

Restore other isolated EADS servers, and then re-execute the command.

KDEA08580-I

Processing to close the server started. (management client = *aa....aa*)

aa....aa: IP address from which the command was executed

Description

The processing to close the EADS server has started.

KDEA08582-E

The wait for completion of execution failed. (management client = *aa....aa*, error details = *bb....bb*)

aa....aa: IP address from which the command was executed

bb....bb: Error details

Description

The wait for the command to finish executing failed.

Action

Re-execute the command.

KDEA08585-E

The compaction of cache data files failed. (management client = *aa....aa*, cache name = *bb....bb*, range ID = *cc....cc*, error details = *dd....dd*)

aa....aa: IP address from which the command was executed

bb....bb: Cache name

cc....cc: Range ID

dd....dd: Error details

Description

Compaction of cache data files failed.

Action

Check the error detail information.

KDEA08586-I

Stopping the compaction of cache data files was reserved. (management client = *aa....aa*)

aa....aa: IP address from which the command was executed

Description

A request to stop compaction of cache data files was queued.

KDEA08587-I

The compaction of cache data files was stopped. (management client = *aa....aa*) *aa....aa*: IP address from which the command was executed

Description

Compaction of cache data files was stopped.

KDEA08588-I

The cache files were deleted. (management client = *aa....aa*, cache name = *bb....bb*)

aa....aa: IP address from which the command was executed

bb....bb: Cache name

Description

The cache files were deleted.

KDEA08589-E

Deletion of cache files failed. (management client = *aa....aa*, cache name = *bb....bb*, error details = *cc....cc*)

aa....aa: IP address from which the command was executed

bb....bb: Cache name

cc....cc: Error details

Description

An attempt to delete cache files failed.

Action

Check the error detail information.

KDEA08591-I

The resumption of cache files started. (management client = *aa....aa*)

aa....aa: IP address from which the command was executed

Description

Cache resume processing has started.

KDEA08592-I

The resumption of cache files finished. (management client = *aa....aa*)

aa....aa: IP address from which the command was executed

Description

Cache resume processing has finished.

KDEA08593-E

The resumption of cache files failed. (management client = *aa....aa*, error details = *bb....bb*) *aa....aa*: IP address from which the command was executed

bb....bb: Error details

Description

Cache resume processing failed.

Action

Check the error detail information.

KDEA08594-I

The importing of cache files started. (management client = *aa....aa*)

aa....aa: IP address from which the command was executed

Description

Cache file import processing has started.

KDEA08595-I

The importing of cache files finished. (management client = *aa....aa*)

aa....aa: IP address from which the command was executed

Description

Cache file import processing has finished.

KDEA08596-E

The importing of cache files failed. (management client = *aa....aa*, error details = *bb....bb*)

aa....aa: IP address from which the command was executed

bb....bb: Error details

Description

Cache file import processing failed.

Action

Check the error detail information.

KDEA08597-E

Creation of cache files failed. (cache name = *aa....aa*, management client = *bb....bb*, error details = *cc....cc*)

aa....aa: Cache name

bb....bb: IP address from which the command was executed

cc....cc: Error details

Description

An attempt to create cache files failed.

Action

Check the error detail information.

KDEA08598-I

Exporting was not executed because no memory cache exist on the server. (management client = *aa....aa*)

aa....aa: IP address from which the command was executed

Description

Data export processing was not performed because there was no memory cache on the EADS server.

Action

Check the error detail information.

KDEA08599-E

The command cannot be executed because the command version and server version are different. (command version = *aa....aa*, server version = *bb....bb*)

aa....aa: Command version

bb....bb: EADS server version

Description

The command cannot be executed because the command's version does not match the EADS server's version.

Action

Execute the command whose version matches the EADS server's version.

KDEA08600-W

No range for the server ID specified in the server ID specification group was found. (server ID = *aa....aa*)

aa....aa: EADS server ID

Description

The range for an EADS server ID specified in the EADS server ID specified group was not found.

Action

Check whether the EADS server ID conversion rule specified in the `eztool import` or `eztool importecf` command is correct.

Check the cluster configuration and each EADS server's EADS server ID.

KDEA08601-E

The user does not have the write permission for the file. (file path = *aa....aa*)

aa....aa: File path

Description

The user does not have write permissions for the file.

Action

Add write permissions to the file.

KDEA08602-E

The user does not have the execute permission for the file. (file path = *aa....aa*)

aa....aa: File path

Description

The user does not have execute permissions for the file.

Action

Add execution permissions to the file.

KDEA08603-E

An attempt to output the configuration after scaling failed. (error details = *aa....aa*)

aa....aa: Error details

Description

An attempt to output the cluster configuration after scale-out processing failed.

Action

Check the error detail information.

When this error has occurred, the current cluster configuration might differ from the cluster configuration defined in the property files. If this is the case, either copy another EADS server's property files that have been output successfully, or manually specify in the property files the cluster configuration after scale-out processing.

22.8 KDEA10000 to KDEA11999

This section describes messages `KDEA10000` to `KDEA11999` and explains what actions to take in response to each message.

KDEA10001-W

> The parameter is outside the valid range. The default value will be used. (parameter = *aa....aa*, value = *bb....bb*, default value = *cc....cc*)

aa....aa: Parameter name

bb....bb: Specified value of the parameter

cc....cc: Default value of the parameter

Description

the specified value of the parameter is invalid. The default value will be used.

Action

Check the specified value of the parameter and change it to the correct value.

KDEA10002-E

> The parameter is outside the valid range or was not specified. (parameter = *aa....aa*, value = *bb....bb*)

aa....aa: Parameter name

bb....bb: Invalid specified value of the parameter (or null or `NULL` if no value was specified)

Description

The parameter was not specified, or the specified value is invalid.

Action

Check the specified value of the parameter, change it to the correct value, and then restart the EADS server.

KDEA10005-E

> A received packet was invalid for the protocol. (local = *aa....aa*:*bb....bb*, remote = *cc....cc*:*dd....dd*)

aa....aa: Local IP address

bb....bb: Local port number

cc....cc: Remote IP address

dd....dd: Remote port number

Description

A received packet was invalid for the protocol.

Action

The possible causes are as follows:

- A problem has occurred at the remote system with which communication was underway.

- A network problem occurred.

Determine the cause of the error, and then eliminate it.

KDEA10006-W

A packet incompatible with the communications protocol was received. (local = *aa....aa*:*bb....bb*, remote = *cc....cc*:*dd....dd*)

aa....aa: Local IP address

bb....bb: Local port number

cc....cc: IP address of the connection source

dd....dd: Port number of the connection source

Description

An incompatible packet was received.

Action

The possible causes are as follows:

- A problem has occurred at the remote system with which communication was underway.
- The remote system is not compatible with the local system.
- A network problem occurred.

Determine the cause of the error, and then eliminate it.

KDEA10007-E

Some parameter values conflict. (parameter1 = *aa....aa*, parameter2 = *bb....bb*)

aa....aa: Parameter 1 containing a conflicting value

bb....bb: Parameter 2 containing a conflicting value

Description

The value of parameter 1 conflicts with the value of parameter 2.

There is a problem in the relationship between the parameters, such as in an inequality relationship.

Action

Check the parameter values, correct them, and then restart the EADS server.

Appendix

A. Glossary

cache

An area for storing the data (pairs of keys and values) handled by EADS.

Up to 16 caches can be created in a cluster.

An EADS client manipulates data associated with a particular cache within a cluster.

check for live servers

This functionality detects a down EADS server in a cluster when it does not send heartbeats.

The check for live servers uses TCP.

cluster

A collection of multiple EADS servers. EADS clients recognize a cluster as a single storage destination.

A cluster consists of a set of EADS servers that share the same port numbers and the same multicast address within the same segment.

disk cache

A cache that uses a disk area to store data.

EADS client

A user program that connects to an EADS server by using client libraries provided by EADS.

EADS server

A server process that manages data consisting of keys and values.

heartbeat

A packet, delivered via multicast within a cluster, that indicates normal operation.

A heartbeat is sent using UDP.

memory cache

A cache that uses a memory area to store data.

scale-out processing

Processing that adds new EADS servers to a cluster without stopping the cluster.

two-way cache

A cache that uses a memory area and a disk area to store data.

user function

A program that defines a series of data operations (user processing) to be performed on a cache, such as data totaling and analysis.

User functions are created by the user and placed on an EADS server in advance. They are called and executed by the EADS client. The two methods for executing user functions are as follows:

- By specifying a key or a group
- By specifying an EADS server

Index