HITACHI
Inspire the Next

Hitachi Advanced Database Command Reference

3000-6-503-H0(E)

Notices

■ Relevant program products

P-9W62-C411 Hitachi Advanced Data Binder version 05-01 (for Red Hat(R) Enterprise Linux(R) Server 6 (64-bit x86_64) and Red Hat(R) Enterprise Linux(R) Server 7 (64-bit x86_64))

P-9W62-C311 Hitachi Advanced Data Binder Client version 05-01 (for Red Hat(R) Enterprise Linux(R) Server 6 (64-bit x86_64) and Red Hat(R) Enterprise Linux(R) Server 7 (64-bit x86_64))

P-2462-C114 Hitachi Advanced Data Binder Client version 05-01 (for Windows 7, Windows 8.1, Windows 10, Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, and Windows Server 2016)

This manual can be used for products other than the products shown above. For details, see the *Release Notes*.

Hitachi Advanced Data Binder is the product name of Hitachi Advanced Database in Japan.

■ Trademarks

HITACHI, HA Monitor, HiRDB, Job Management Partner 1 and JP1 are either trademarks or registered trademarks of Hitachi, Ltd. in Japan and other countries.

Access is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

AMD is a trademark of Advanced Micro Devices, Inc.

Excel is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

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

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

Microsoft is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

MSDN is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

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

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

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

UNIX is a trademark of The Open Group.

Visual Studio is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

Windows is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

Windows Server is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

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

1. This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)

2. This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

3. This product includes software written by Tim Hudson (tjh@cryptsoft.com).

4. This product uses OpenSSL Toolkit software in accordance with the OpenSSL License and Original SSLeay License, which are described as follows.

LICENSE ISSUES
==============

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts. Actually both licenses are BSD-style Open Source licenses. In case of any license issues related to OpenSSL please contact openssl-core@openssl.org.

OpenSSL License

```
/* ====================================================================
* Copyright (c) 1998-2011 The OpenSSL Project. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* 1. Redistributions of source code must retain the above copyright
* notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in
* the documentation and/or other materials provided with the
* distribution.
*
* 3. All advertising materials mentioning features or use of this
* software must display the following acknowledgment:
* "This product includes software developed by the OpenSSL Project
* for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
*
* 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
* endorse or promote products derived from this software without
* prior written permission. For written permission, please contact
* openssl-core@openssl.org.
*
* 5. Products derived from this software may not be called "OpenSSL"
* nor may "OpenSSL" appear in their names without prior written
* permission of the OpenSSL Project.
*
* 6. Redistributions of any form whatsoever must retain the following
* acknowledgment:
* "This product includes software developed by the OpenSSL Project
* for use in the OpenSSL Toolkit (http://www.openssl.org/)"
```

*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the copyright
* notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in the
* documentation and/or other materials provided with the distribution.
* 3. All advertising materials mentioning features or use of this software
* must display the following acknowledgement:
* "This product includes cryptographic software written by
* Eric Young (eay@cryptsoft.com)"
* The word 'cryptographic' can be left out if the rouines from the library
* being used are not cryptographic related :-).
* 4. If you include any Windows specific code (or a derivative thereof) from
* the apps directory (application code) you must include an acknowledgement:
* "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
*
* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
* The licence and distribution terms for any publically available version or
* derivative of this code cannot be changed. i.e. this code cannot simply be
* copied and put under another distribution licence
* [including the GNU Public Licence.]
*/
■Double precision SIMD-oriented Fast Mersenne Twister (dSFMT)
Copyright (c) 2007, 2008, 2009 Mutsuo Saito, Makoto Matsumoto
and Hiroshima University.
Copyright (c) 2011, 2002 Mutsuo Saito, Makoto Matsumoto, Hiroshima
University and The University of Tokyo.
All rights reserved.

■ Microsoft product screen shots

Microsoft product screen shots reprinted with permission from Microsoft Corporation.

■ Microsoft product name abbreviations

This manual uses the following abbreviations for Microsoft product names:

Abbreviation			Full name or meaning
Windows	Windows 7	Windows 7 x86	Microsoft(R) Windows(R) 7 Professional (32-bit)
			Microsoft(R) Windows(R) 7 Enterprise (32-bit)
			Microsoft(R) Windows(R) 7 Ultimate (32-bit)
		Windows 7 x64	Microsoft(R) Windows(R) 7 Professional (64-bit)
			Microsoft(R) Windows(R) 7 Enterprise (64-bit)
			Microsoft(R) Windows(R) 7 Ultimate (64-bit)
	Windows 8.1	Windows 8.1 x86	Windows(R) 8.1 Pro (32-bit)
			Windows(R) 8.1 Enterprise (32-bit)

Abbreviation			Full name or meaning
		Windows 8.1 x64	Windows$^{(R)}$ 8.1 Pro (64-bit)
			Windows$^{(R)}$ 8.1 Enterprise (64-bit)
	Windows 10	Windows 10 x86	Windows$^{(R)}$ 10 Pro (32-bit)
			Windows$^{(R)}$ 10 Enterprise (32-bit)
		Windows 10 x64	Windows$^{(R)}$ 10 Pro (64-bit)
			Windows$^{(R)}$ 10 Enterprise (64-bit)
	Windows Server 2008 R2		Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2008 R2 Standard
			Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2008 R2 Enterprise
			Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2008 R2 Datacenter
	Windows Server 2012		Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2012 Standard
			Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2012 Datacenter
	Windows Server 2012 R2		Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2012 R2 Standard
			Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2012 R2 Datacenter
	Windows Server 2016		Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2016 Standard
			Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2016 Datacenter

■ 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

Apr. 2020

■ Copyright

Preface

This manual describes the command syntax used by Hitachi Advanced Database.

Note that, in this manual, and in the information output by the product (messages, command output results, and so on), *HADB* is often used in place of *Hitachi Advanced Database*.

■ Intended readers

This manual is intended for system engineers who design and set up HADB systems, and for system administrators.

Readers of this manual must have:

- A basic knowledge of Linux or Windows system management
- A basic knowledge of RDBMS operation management
- A basic knowledge of SQL

■ Organization of this manual

This manual is organized into the following chapters:

1. List of Commands and Common Rules
   This chapter lists the HADB commands and explains the command specification format and common rules.

2. adbarchivechunk (Archive Chunk)
   This chapter explains the `adbarchivechunk` command's functions and how to use the command.

3. adbaudittrail (Manage the Audit Trail Facility)
   This chapter explains the `adbaudittrail` command's functions and how to use the command.

4. adbcancel (Forcibly Terminate Transactions or Commands)
   This chapter explains the `adbcancel` command's functions and how to use the command.

5. adbchgchunkcomment (Set, Change, and Delete Comments for Chunks)
   This chapter explains the `adbchgchunkcomment` command's functions and how to use the command.

6. adbchgchunkstatus (Change Chunk Status)
   This chapter explains the `adbchgchunkstatus` command's functions and how to use the command.

7. adbchgnodetype (Change the HADB Server's Node Type)
   This chapter explains the `adbchgnodetype` command's functions and how to use the command.

8. adbchgsqltrc (Start or Stop Output of SQL Trace Information)
   This chapter explains the `adbchgsqltrc` command's functions and how to use the command.

9. adbchgsrvmode (Change the HADB Server Operation Mode)
   This chapter explains the `adbchgsrvmode` command's functions and how to use the command.

10. adbclientdefmang (Centralized Management of Client Definitions)
    This chapter explains the `adbclientdefmang` command's functions and how to use the command.

11. adbcolumnize (Manage the Updated-Row Columnizing Facility)

This chapter explains the `adbcolumnize` command's functions and how to use the command.

12. adbconvertaudittrailfile (Convert the Audit Trail File)

This chapter explains the `adbconvertaudittrailfile` command's functions and how to use the command.

13. adbdbstatus (Analyze the Database Status)

This chapter explains the `adbdbstatus` command's functions and how to use the command.

14. adbexport (Export Data)

This chapter explains the `adbexport` command's functions and how to use the command.

15. adbgetcst (Collect Cost Information)

This chapter explains the `adbgetcst` command's functions and how to use the command.

16. adbidxrebuild (Rebuild Indexes)

This chapter explains the `adbidxrebuild` command's functions and how to use the command.

17. adbimport (Import Data)

This chapter explains the `adbimport` command's functions and how to use the command.

18. adbinfoget (Collect Troubleshooting Information)

This chapter explains the `adbinfoget` command's functions and how to use the command.

19. adbinfosweep (Delete Troubleshooting Information)

This chapter explains the `adbinfosweep` command's functions and how to use the command.

20. adbinit (Initialize the Database)

This chapter explains the `adbinit` command's functions and how to use the command.

21. adbinstall (Install HADB Server or Client)

This chapter explains the `adbinstall` command's functions and how to use the command.

22. adbls -d cltgrp (Display Information of Client Groups and Command Groups)

This chapter explains the `adbls -d cltgrp` command's functions and how to use the command.

23. adbls -d cnct (Display the Connection Status)

This chapter explains the `adbls -d cnct` command's functions and how to use the command.

24. adbls -d gbuf (Display Global Buffer Information)

This chapter explains the `adbls -d gbuf` command's functions and how to use the command.

25. adbls -d lbuf (Display Local Work Table Buffer Information)

This chapter explains the `adbls -d lbuf` command's functions and how to use the command.

26. adbls -d lock (Display the Status of Locked Resources)

This chapter explains the `adbls -d lock` command's functions and how to use the command.

27. adbls -d mem (Display the Memory Usage Status)

This chapter explains the `adbls -d mem` command's functions and how to use the command.

28. adbls -d node (Display the HADB Server Status on Each Node)

This chapter explains the `adbls -d node` command's functions and how to use the command.

29. adbls -d shm (Display Shared Memory Information)

This chapter explains the `adbls -d shm` command's functions and how to use the command.

30. adbls -d srv (Display the HADB Server Status)

This chapter explains the `adbls -d srv` command's functions and how to use the command.

31. adbls -d thd (Display the Thread Status)

This chapter explains the `adbls -d thd` command's functions and how to use the command.

32. adbls -d ver (Display the HADB Server's Version Information)

This chapter explains the `adbls -d ver` command's functions and how to use the command.

33. adbmergechunk (Merge Chunks)

This chapter explains the `adbmergechunk` command's functions and how to use the command.

34. adbmodarea (Add and Change DB Areas)

This chapter explains the `adbmodarea` command's functions and how to use the command.

35. adbmodbuff (Change the Buffer)

This chapter explains the `adbmodbuff` command's functions and how to use the command.

36. adbmonitor (Monitor the HADB Server)

This chapter explains the `adbmonitor` command's functions and how to use the command.

37. adbreorgsystemdata (Reorganize the System Table)

This chapter explains the `adbreorgsystemdata` command's functions and how to use the command.

38. adbsql (Execute SQL Statements)

This chapter explains the `adbsql` command's functions and how to use the command.

39. adbstart (Start the HADB Server)

This chapter explains the `adbstart` command's functions and how to use the command.

40. adbstat (Perform Statistical Analysis of the HADB Server)

This chapter explains the `adbstat` command's functions and how to use the command.

41. adbstop (Terminate the HADB Server)

This chapter explains the `adbstop` command's functions and how to use the command.

42. adbsyndict (Manage the Synonym Dictionary)

This chapter explains the `adbsyndict` command's functions and how to use the command.

43. adbunarchivechunk (Unarchive Chunks)

This chapter explains the `adbunarchivechunk` command's functions and how to use the command.

■ Related publications

This manual is part of a related set of manuals. The manuals in the set are listed below (with the manual numbers):

- *Hitachi Advanced Database Setup and Operation Guide* (3000-6-501(E))
- *Hitachi Advanced Database Application Development Guide* (3000-6-502(E))
- *Hitachi Advanced Database SQL Reference* (3000-6-504(E))
- *Hitachi Advanced Database Messages* (3000-6-505(E))
- *HA Monitor Cluster Software Guide (for Linux$^{(R)}$ (x86) Systems)* (3000-9-201(E))

- *Job Management Partner 1 Version 10 Job Management Partner 1/Automatic Job Management System 3 System Design (Work Tasks) Guide* (3021-3-320(E))
- *JP1 Version 11 JP1/Base User's Guide* (3021-3-A01(E))

In references to Hitachi Advanced Database manuals, this manual uses *HADB* in place of *Hitachi Advanced Database*.

Example: *HADB Setup and Operation Guide*

In references to the HA Monitor manual, this manual uses *HA Monitor for Linux$^{(R)}$ (x86)* in place of *HA Monitor Cluster Software Guide (for Linux$^{(R)}$ (x86) Systems)*.

Example: *HA Monitor for Linux$^{(R)}$ (x86)*

In references to the Job Management Partner 1/Automatic Job Management System 3 manual, this manual uses *Job Management Partner 1/Automatic Job Management System 3 System Design (Work Tasks) Guide* in place of *Job Management Partner 1 Version 10 Job Management Partner 1/Automatic Job Management System 3 System Design (Work Tasks) Guide*.

Example: *Job Management Partner 1/Automatic Job Management System 3 System Design (Work Tasks) Guide*

In references to the JP1/Base manual, this manual uses *JP1/Base User's Guide* in place of *JP1 Version 11 JP1/Base User's Guide*.

Example: *JP1/Base User's Guide*

■ Conventions: Abbreviations for product names

This manual uses the following abbreviations for product names:

Abbreviation		Full name or meaning
HADB	HADB server	Hitachi Advanced Database
	HADB client	Hitachi Advanced Database Client
Linux	Linux	Linux$^{(R)}$
	Red Hat Enterprise Linux Server 6	Red Hat$^{(R)}$ Enterprise Linux$^{(R)}$ Server 6 (64-bit x86_64)
	Red Hat Enterprise Linux Server 6 (64-bit x86_64)	
	Red Hat Enterprise Linux Server 7	Red Hat$^{(R)}$ Enterprise Linux$^{(R)}$ Server 7 (64-bit x86_64)
	Red Hat Enterprise Linux Server 7 (64-bit x86_64)	
HDLM		Hitachi Dynamic Link Manager Software
JP1/AJS3		Job Management Partner 1/Automatic Job Management System 3
JP1/Audit		JP1/Audit Management - Manager
Red Hat Enterprise Linux Server 6 (64-bit x86_64)		Red Hat$^{(R)}$ Enterprise Linux$^{(R)}$ Server 6 (64-bit x86_64)
Red Hat Enterprise Linux Server 7 (64-bit x86_64)		Red Hat$^{(R)}$ Enterprise Linux$^{(R)}$ Server 7 (64-bit x86_64)

■ Conventions: Acronyms

This manual also uses the following acronyms:

Acronym	Full name or meaning
APD	Application Parameter Descriptor
API	Application Programming Interface
ARD	Application Row Descriptor
BI	Business Intelligence
BLOB	Binary Large Object
BNF	Backus-Naur Form
BOM	Byte Order Mark
CLI	Call Level Interface
CLOB	Character Large Object
CPU	Central Processing Unit
CSV	Character-Separated Values
DB	Database
DBMS	Database Management System
DMMP	Device Mapper Multipath
DNS	Domain Name System
ER	Entity Relationship
HBA	Host Bus Adapter
ID	Identification number
IEF	Integrity Enhancement Facility
IP	Internet Protocol
IPD	Implementation Parameter Descriptor
IRD	Implementation Row Descriptor
JAR	Java Archive File
JDBC	Java Database Connectivity
JDK	Java Developer's Kit
JNDI	Java Naming and Directory Interface
JRE	Java Runtime Environment
JTA	Java Transaction API
LOB	Large Object
LRU	Least Recently Used
LV	Logical Volume

Acronym	Full name or meaning
LVM	Logical Volume Manager
MSDN	Microsoft Developer Network
NFS	Network File System
NIC	Network Interface Card
NTP	Network Time Protocol
ODBC	Open Database Connectivity
OS	Operating System
PP	Program Product
RAID	Redundant Array of Independent Disks
RDBMS	Relational Database Management System
TLB	Translation Lookaside Buffer
URL	Uniform Resource Locator
VG	Volume Group
WWN	World Wide Name

■ Conventions: Fonts and symbols

The following table explains the fonts used in this manual:

Font	Convention
Bold	**Bold** type indicates text on 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**. • Click the **Cancel** button. • In the **Enter name** entry box, type your name.
Italics	*Italics* are used to 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*`)` *Italics* are also used for emphasis. For example: • Do *not* delete the configuration file.
`Code font`	A `code font` indicates 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 table below shows the symbols used in this manual for explaining commands and operands, such as the operands used in server definitions.

Note that these symbols are used for explanatory purposes only; do not specify them in the actual operand or command.

Symbol	Meaning	Example
|	In syntax explanations, a vertical bar separates multiple items, and has the meaning of OR.	`adb_sql_text_out = {Y|N}` In this example, the vertical bar means that you can specify either `Y` or `N`.
[ ]	In syntax explanations, square brackets indicate that the enclosed item or items are optional.	`adbsql [-V]` In this example, the square brackets mean that you can specify `adbsql`, or you can specify `adbsql -V`.
{ }	In syntax explanations, curly brackets indicate that only one of the enclosed items is to be selected.	`adbcancel {--ALL|-u` *connection-ID*`}` In this example, the curly brackets mean that you can specify either `--ALL` or `-u` *connection-ID*.
...	In syntax explanations, an ellipsis (`...`) indicates that the immediately preceding item can be repeated as many times as necessary.	`adbbuff -n` *DB-area-name*`[,` *DB-area-name*`] ...` In this example, the ellipsis means that you can specify *DB-area-name* as many times as necessary.
{{ }}	In syntax explanations, double curly brackets indicate that the enclosed items can be repeated as a single unit.	`{{adbinitdbarea -n` *data-DB-area-name*`}}` In this example, the double curly brackets mean that you can specify `adbinitdbarea -n` *data-DB-area-name* as many times as necessary.
X (underline)	In syntax explanations, underlined characters indicate a default value.	`adb_import_errmsg_lv = {0|1}` In this example, the underline means that the value `0` is assumed by HADB when the operand is omitted.
~	A swung dash indicates that the text following it explains the properties of the specified value.	`adb_sys_max_users` = *maximum-number-of-concurrent-connections* ~ <integer> ((`1` to `1024`)) <<`10`>> In this example, the text following the swung dash means that you can specify an integer in the range from `1` to `1024`. If the operand is not specified, the value `10` is assumed by HADB.
< >	Single angle brackets explain the data type of the specified value.	
(( ))	Double parentheses indicate the scope of the specified value.	
<< >>	Double angle brackets indicate a default value.	

■ Conventions: Path names

- `$INSTDIR` is used to indicate the server directory path (for installation).
- `$ADBDIR` is used to indicate the server directory path (for operation).
- `$DBDIR` is used to indicate the DB directory path.
- `%ADBCLTDIR%` (for a Windows HADB client) or `$ADBCLTDIR` (for a Linux HADB client) is used to indicate the client directory path.
- `%ADBODBTRCPATH%` is used to indicate the folder path where HADB's ODBC driver trace files are stored.

■ Conventions: Symbols used in mathematical formulas

The following table explains special symbols used by this manual in mathematical formulas:

Symbol	Meaning
↑↑	Round up the result to the next integer. Example: The result of ↑34 ÷ 3↑ is 12.
↓↓	Discard digits following the decimal point. Example: The result of ↓34 ÷ 3↓ is 11.
MAX	Select the largest value as the result. Example: The result of MAX(3 × 6, 4 + 7) is 18.
MIN	Select the smallest value as the result. Example: The result of MIN(3 × 6, 4 + 7) is 11.

■ Conventions: Syntax elements

Syntax element notation	Meaning
<path name>	The following characters can be used in path names: • In Linux Alphanumeric characters, hash mark (#), hyphen (-), forward slash (/), at mark (@), and underscore (_) • In Windows Alphanumeric characters, hash mark (#), hyphen (-), forward slash (/), at mark (@), underscore (_), backslash (\), and colon (:) Note, however, that the characters that can be used might differ depending on the operating system.
<OS path name>	For an OS path name, all characters that can be used in a path name in the operating system can be used. For details about available characters, see the documentation for the operating system you are using.
<character string>	Any character string can be specified.

<integer suffixed by the unit>	Specify the value in a format consisting of a numeric character (in the range from `0` to `9`) followed by a unit (`MB` (megabyte), `GB` (gigabyte), or `TB` (terabyte)). Do not enter a space between the numeric character and the unit.
	• Examples of correct specification
	`1024MB`
	`512GB`
	`32TB`
	• Example of specification that causes an error
	`512 GB`

■ Conventions: KB, MB, GB, TB, PB, and EB

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.
- 1 PB (petabyte) is 1,024^5 bytes.
- 1 EB (exabyte) is 1,024^6 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

1 List of Commands and Common Rules

This chapter lists the HADB commands and explains the command specification format and common rules.

1.1 List of commands

The following two tables list and describe the HADB commands.

- Table 1-1: List of commands (command execution environment)

  This table provides information such as OS users who can execute the command and privileges required for command execution.

- Table 1-2: List of commands (database recovery by using a backup of system data only)

  This table indicates whether the database can be recovered (after execution of the command) by using a backup of system data only.

For details about the types of nodes on which commands can be executed when the multi-node function is used, see 1.2 Nodes on which commands can be executed when the multi-node function is used.

Table 1-1: List of commands (command execution environment)

No.	Function		Command name	Command execution environment				
				OS users who can execute the command	Privileges required for command execution[#1]	Computers that can execute the command	HADB server status	Connection to HADB server
1	Installation and environment setup	Install	`adbinstall`	HADB administrator and HADB client managing OS users	--	HADB server and HADB client (for Linux)	Inactive	No
2		Initialize the database	`adbinit`	HADB administrator	--	HADB server	Inactive	No
3		Add and change DB areas	`adbmodarea`	HADB administrator	--	HADB server	Active	Yes[#2]
4		Change buffer	`adbmodbuff`	HADB administrator	--	HADB server	Active	Yes[#2]
5	Start and terminate the HADB server	Start the HADB server	`adbstart`	HADB administrator	--	HADB server	Inactive	No
6		Terminate the HADB server	`adbstop`	HADB administrator	--	HADB server	Active	No
7	Database maintenance and backup	Change the HADB server operation mode	`adbchgsrvmode`	HADB administrator	--	HADB server	Active	No
8	Table data import and export	Import data to base tables	`adbimport`	HADB administrator and OS users who belong to the HADB administrators group	- `CONNECT` privilege - `IMPORT TABLE` privilege for a table	HADB server	Active	Yes[#2]

No.	Function		Command name	Command execution environment				
				OS users who can execute the command	Privileges required for command execution#1	Computers that can execute the command	HADB server status	Connection to HADB server
					to be imported			
9		Export table data	`adbexport`	HADB administrator and OS users who belong to the HADB administrators group	• `CONNECT` privilege • `EXPORT TABLE` privilege for a table to be exported	HADB server	Active	Yes#2
10	Index maintenance	Rebuild indexes	`adbidxrebuild`	HADB administrator and OS users who belong to the HADB administrators group	• `CONNECT` privilege • `REBUILD INDEX` privilege for a table for which indexes are defined	HADB server	Active	Yes#2
11	Chunk maintenance	Merge chunks	`adbmergechunk`	HADB administrator and OS users who belong to the HADB administrators group	• `CONNECT` privilege • `MERGE CHUNK` privilege for a table for which chunks are defined	HADB server	Active	Yes#2
12		Archive chunks	`adbarchivechunk`	HADB administrator and OS users who belong to the HADB administrators group	• `CONNECT` privilege • `ARCHIVE CHUNK` privilege for a table for which chunks are defined	HADB server	Active	Yes#2
13		Unarchive chunks	`adbunarchivechunk`	HADB administrator and OS users who belong to	• `CONNECT` privilege	HADB server	Active	Yes#2

No.	Function		Command name	Command execution environment				
				OS users who can execute the command	Privileges required for command execution#1	Computers that can execute the command	HADB server status	Connection to HADB server
				the HADB administrators group	- `UNARCHIVE CHUNK` privilege for a table for which chunks are defined			
14		Change chunk status	`adbchgchunkstatus`	HADB administrator and OS users who belong to the HADB administrators group	- `CONNECT` privilege - `CHANGE CHUNK STATUS` privilege for a table for which chunks are defined	HADB server	Active	Yes#2
15		Set, change, and delete comments for chunks	`adbchgchunkcomment`	HADB administrator and OS users who belong to the HADB administrators group	- `CONNECT` privilege - `CHANGE CHUNK COMMENT` privilege for a table for which chunks are defined	HADB server	Active	Yes#2
16	System table (base table) maintenance	Reorganize the system table (base table)	`adbreorgsystemdata`	HADB administrator	--	HADB server	Active	Yes#2
17	HADB server status display	Display information on client groups and command groups	`adbls -d cltgrp`	HADB administrator and OS users who belong to the HADB administrators group	--	HADB server	C	No
18		Display the connection status	`adbls -d cnct`	HADB administrator and OS users who belong to	--	HADB server	C	No

No.	Function		Command name	Command execution environment				
				OS users who can execute the command	Privileges required for command execution#1	Computers that can execute the command	HADB server status	Connection to HADB server
				the HADB administrators group				
19		Display global buffer information	`adbls -d gbuf`	HADB administrator and OS users who belong to the HADB administrators group	--	HADB server	C	No
20		Display local work table buffer information	`adbls -d lbuf`	HADB administrator and OS users who belong to the HADB administrators group	--	HADB server	C	No
21		Display the status of locked resources	`adbls -d lock`	HADB administrator and OS users who belong to the HADB administrators group	--	HADB server	C	No
22		Display the memory usage status	`adbls -d mem`	HADB administrator and OS users who belong to the HADB administrators group	--	HADB server	C	No
23		Display shared memory information	`adbls -d shm`	HADB administrator and OS users who belong to the HADB administrators group	--	HADB server	C	No
24		Display the HADB server status	`adbls -d srv`	HADB administrator and OS users who belong to the HADB administrators group	--	HADB server	Y	No
25		Display the thread status	`adbls -d thd`	HADB administrator and OS users who belong to	--	HADB server	C	No

No.	Function		Command name	Command execution environment				
				OS users who can execute the command	Privileges required for command execution#1	Computers that can execute the command	HADB server status	Connection to HADB server
				the HADB administrators group				
26		Display version information	`adbls -d ver`	HADB administrator and OS users who belong to the HADB administrators group	--	HADB server	Y	No
27	Execution of SQL statements	Execute SQL statements from the standard input or a file	`adbsql`	No restriction	• `CONNECT` privilege • Access privileges required for an SQL statement to be executed	HADB server and HADB client (for Linux)	Active	As client
28	Dictionary management	Perform operation such as registering and deleting a synonym dictionary	`adbsyndict`	HADB administrator	--	HADB server	Active	Yes#2
29	Information collection and analysis	Analyze the database status	`adbdbstatus`	HADB administrator and OS users who belong to the HADB administrators group	--	HADB server	Active	Yes#2
30		Collect cost information	`adbgetcst`	HADB administrator and OS users who belong to the HADB administrators group	• `CONNECT` privilege • `GET COSTINFO` privilege for a table for which cost information is to be collected	HADB server	Active	Yes#2

No.	Function		Command name	Command execution environment				
				OS users who can execute the command	Privileges required for command execution#1	Computers that can execute the command	HADB server status	Connection to HADB server
31		Display the HADB server's operation information	`adbstat`	HADB administrator and OS users who belong to the HADB administrators group	--	HADB server	Active	No
32		Start or stop output of SQL trace information	`adbchgsqltrc`	HADB administrator	--	HADB server	Active	No
33	Troubleshooting	Forcibly terminate transactions or commands	`adbcancel`	HADB administrator	--	HADB server	Active	No
34		Collect troubleshooting information	`adbinfoget`	HADB administrator	--	HADB server	Y#4	No
35		Delete troubleshooting information	`adbinfosweep`	HADB administrator	--	HADB server	Inactive	No
36	Multi-node function	Display the HADB server status on each node	`adbls -d node`	HADB administrator and OS users who belong to the HADB administrators group	--	HADB server	C	No
37		Change the node type	`adbchgnodetype`	HADB administrator	--	HADB server	Active	No
38		Monitor the HADB server	`adbmonitor`	HADB administrator	--	HADB server	Active	No
39	Centralized management of client definitions	Centralized management of client definitions	`adbclientdefmang`	HADB administrator	• `DBA` privilege • `CONNECT` privilege	HADB server	Active	Yes
40	Audit trail facility	Manage the audit trail facility	`adbaudittrail`	HADB administrator and OS users who belong to the HADB administrators group	• `CONNECT` privilege • Audit admin privilege • Audit viewer	HADB server	Active	Yes

No.	Function		Command name	Command execution environment				
				OS users who can execute the command	Privileges required for command execution#1	Computers that can execute the command	HADB server status	Connection to HADB server
					privilege#3			
41		Conversion of audit trail files	`adbconvertaudittrailfile`	HADB administrator and OS users who belong to the HADB administrators group	• `CONNECT` privilege • Audit viewer privilege	HADB server	Active	Yes
42	Updated-row columnizing facility	Manage the updated-row columnizing facility	`adbcolumnize`	HADB administrator	--	HADB server	Active	No

Legend:

--: Not applicable.

Active: The command can be executed while the HADB server is running.

Inactive: The command can be executed while the HADB server is not running.

Y: The command can be executed regardless of whether the HADB server is running.

C: The command can be executed while the HADB server is running, or between the time the HADB server terminated abnormally and the time the HADB server is restarted or until troubleshooting information is deleted by the `adbinfosweep` command.

Yes: Establishes a connection to the HADB server during command execution.

As client: Establishes a connection to the HADB server as an HADB client.

No: Does not establish a connection to the HADB server.

#1

You specify the authorization identifier of an HADB user who has this privilege in each command's `-u` option.

#2

You must consider not only the number of HADB clients but also the number of commands that connect to the HADB server.

For details about the number of commands that can connect concurrently to the HADB server, see *Points to consider when executing commands concurrently* in *Preparing Resources* in the *HADB Setup and Operation Guide*.

#3

An HADB user with the audit viewer privilege and without the audit admin privilege can execute the `adbaudittrail` command with only the `-d` option specified. To execute the `adbaudittrail` command with any other option specified, the audit admin privilege is required.

#4

To execute the `adbinfoget` command with the `-g` option specified, make sure that the HADB server is running.

The following table describes database recovery by using a backup of system data only.

Table 1-2: List of commands (database recovery by using a backup of system data only)

No.	Function		Command name	Database recovery by using a backup of system data only
1	Installation and environment setup	Install	`adbinstall`	Y
2		Initialize the database	`adbinit`	N
3		Add and change DB areas	`adbmodarea`	N
4		Change buffer	`adbmodbuff`	Y
5	Start and terminate the HADB server	Start the HADB server	`adbstart`	Y
6		Terminate the HADB server	`adbstop`	Y
7	Database maintenance and backup	Change the HADB server operation mode	`adbchgsrvmode`	Y
8	Table data import and export	Import data to base tables	`adbimport`	N
9		Export table data	`adbexport`	Y
10	Index maintenance	Rebuild indexes	`adbidxrebuild`	N
11	Chunk maintenance	Merge chunks	`adbmergechunk`	N
12		Archive chunks	`adbarchivechunk`	N
13		Unarchive chunks	`adbunarchivechunk`	N
14		Change chunk status	`adbchgchunkstatus`	N
15		Set, change, and delete comments for chunks	`adbchgchunkcomment`	N
16	System table (base table) maintenance	Reorganize the system table (base table)	`adbreorgsystemdata`	Y
17	HADB server status display	Display information on client groups and command groups	`adbls -d cltgrp`	Y
18		Display the connection status	`adbls -d cnct`	Y
19		Display global buffer information	`adbls -d gbuf`	Y
20		Display local work table buffer information	`adbls -d lbuf`	Y
21		Display the status of locked resources	`adbls -d lock`	Y

No.	Function		Command name	Database recovery by using a backup of system data only
22		Display the memory usage status	`adbls -d mem`	Y
23		Display shared memory information	`adbls -d shm`	Y
24		Display the HADB server status	`adbls -d srv`	Y
25		Display the thread status	`adbls -d thd`	Y
26		Display version information	`adbls -d ver`	Y
27	Execution of SQL statements	Execute SQL statements from the standard input or a file	`adbsql`	C
28	Dictionary management	Perform operation such as registering and deleting a synonym dictionary	`adbsyndict`	N
29	Information collection and analysis	Analyze the database status	`adbdbstatus`	Y
30		Collect cost information	`adbgetcst`	N
31		Display the HADB server's operation information	`adbstat`	Y
32		Start or stop output of SQL trace information	`adbchgsqltrc`	Y
33	Troubleshooting	Forcibly terminate transactions or commands	`adbcancel`	Y
34		Collect troubleshooting information	`adbinfoget`	Y
35		Delete troubleshooting information	`adbinfosweep`	Y
36	Multi-node function	Display the HADB server status on each node	`adbls -d node`	Y
37		Change the node type	`adbchgnodetype`	Y
38		Monitor the HADB server	`adbmonitor`	Y

No.	Function		Command name	Database recovery by using a backup of system data only
39	Centralized management of client definitions	Centralized management of client definitions	`adbclientdefmang`	Y
40	Audit trail facility	Manage the audit trail facility	`adbaudittrail`	Y
41		Conversion of audit trail files	`adbconvertaudittrailfile`	Y
42	Updated-row columnizing facility	Manage the updated-row columnizing facility	`adbcolumnize`	N

Legend:

Y: Even if this command is executed after upgrading of the HADB server, the HADB server can be returned to the old version by using a backup of system data only.

N: If this command is executed after upgrading of the HADB server, the HADB server cannot be returned to the old version from a backup of system data only. In this case, a full backup is required to return the HADB server to the old version.

C: If operation on only the following is performed, the HADB server can be returned to the old version by using a backup of system data only.

- SQL statement (`SELECT` statement, `COMMIT` statement, or `ROLLBACK` statement)
- All `adbsql` subcommands

A backup of system data only must be created before the version upgrade.

For details about backing up the system data only when upgrading the HADB server, see *Steps to take before upgrading the server version* in the *HADB Setup and Operation Guide*.

1.2 Nodes on which commands can be executed when the multi-node function is used

The table below lists the nodes on which commands can be executed when the multi-node function is used.

For details about the execution conditions for each command, see 1.1 List of commands.

Table 1-3: Nodes on which commands can be executed when the multi-node function is used

No.	Function		Command name	Type of nodes on which commands can be executed	
				Master node	Slave node
1	Installation and environment setup	Install	`adbinstall`	Y	Y
2		Initialize the database	`adbinit`	Y	Y
3		Add and change DB areas	`adbmodarea`	Y	N
4		Change buffers	`adbmodbuff`	Y	N
5	Start and terminate the HADB server	Start the HADB server	`adbstart`	Y	Y
6		Terminate the HADB server	`adbstop`	Y	Y#1
7	Database maintenance and backup	Change the HADB server operation mode	`adbchgsrvmode`	Y	N
8	Table data import and export	Import data to base tables	`adbimport`	Y	N
9		Export table data	`adbexport`	Y	N
10	Index maintenance	Rebuild indexes	`adbidxrebuild`	Y	N
11	Chunk maintenance	Merge chunks	`adbmergechunk`	Y	N
12		Archive chunks	`adbarchivechunk`	Y	N
13		Unarchive chunks	`adbunarchivechunk`	Y	N
14		Change chunk status	`adbchgchunkstatus`	Y	N
15		Set, change, and delete comments for chunks	`adbchgchunkcomment`	Y	N
16	System table (base table) maintenance	Reorganize the system table (base table)	`adbreorgsystemdata`	Y	N
17	HADB server status display	Display information on client groups and command groups	`adbls -d cltgrp`	Y	Y
18		Display the connection status	`adbls -d cnct`	Y	Y

No.	Function		Command name	Type of nodes on which commands can be executed	
				Master node	Slave node
19		Display global buffer information	adbls -d gbuf	Y	Y
20		Display local work table buffer information	adbls -d lbuf	Y	Y
21		Display the status of locked resources	adbls -d lock	Y	Y
22		Display the memory usage status	adbls -d mem	Y	Y
23		Display shared memory information	adbls -d shm	Y	Y
24		Display the HADB server status	adbls -d srv	Y	Y
25		Display the thread status	adbls -d thd	Y	Y
26		Display version information	adbls -d ver	Y	Y
27	Execution of SQL statements	Execute SQL statements from the standard input or a file	adbsql	Y	Y
28	Dictionary management	Perform operation such as registering and deleting a synonym dictionary	adbsyndict	Y	N
29	Information collection and analysis	Analyze the database status	adbdbstatus	Y	N
30		Collect cost information	adbgetcst	Y	N
31		Display the HADB server's operation information	adbstat	Y	Y
32		Start or stop output of SQL trace information	adbchgsqltrc	Y	Y
33	Trouble-shooting	Forcibly terminate transactions or commands	adbcancel	Y	Y
34		Collect trouble-shooting information	adbinfoget	Y	Y
35		Delete trouble-shooting information	adbinfosweep	Y	Y

No.	Function		Command name	Type of nodes on which commands can be executed	
				Master node	Slave node
36	Multi-node function	Display the HADB server status on each node	`adbls -d node`	Y	Y
37		Change node type	`adbchgnodetype`	N	Y
38		Monitor the HADB Server	`adbmonitor`	Y	Y
39	Centralized management of client definitions	Centralized management of client definitions	`adbclientdefmang`	Y	N
40	Audit trail facility	Manage the audit trail facility	`adbaudittrail`	Y	Y#2
41		Conversion of audit trail files	`adbconvertaudittrailfile`	Y	Y
42	Updated-row columnizing facility	Manage the updated-row columnizing facility	`adbcolumnize`	Y	N

Legend:

Y: Command can be executed.

N: Command cannot be executed.

#1

When executing the `adbstop` command on a slave node, specify the `--node` option.

#2

The node on which execution is performed differs depending on the specified option. For details, see Table 3-3: Correspondences between the options to specify and the nodes on which the adbaudittrail command can be executed in 3.5 Notes.

1.3 Format of command explanations

The following shows the command specification format:

```
command-name option command-argument
```

The following explains *command-name*, *option*, and *command-argument* using the `adbimport` command as an example.

```
adbimport  -u ADBUSER01  TABLE01  ...
   [1]         [2]         [3]
```

Explanation:

1. This is the name of the command. Specify either the name or the path name of the command to execute.
2. This is an option that is associated with the command. An option consists of a string of alphanumeric characters beginning with one hyphen (`-`) or two hyphens (`--`).

   Some options require one or more option arguments and some options require no arguments. In the above example, `-u` requires an option argument; `ADBUSER01` is its option argument. Specify multiple option arguments by delimiting them with a comma (`,`).
3. This is a command argument.

Rules for options and option arguments

- Alphabetic characters specified as options are case sensitive.
- When an option requires an option argument, the option argument cannot be omitted.
- If an option argument contains a space, you must enclose the argument in double quotation marks (`"`).

  Example: `adbimport -z "/home/importΔoption/env01.txt"`

  In this example, `-z` is an option for specifying a path name. If the path name contains a space, the path name must be enclosed in double quotation marks (`"`), as shown in the example (where Δ indicates the space).
- To include a double quotation mark (`"`) in an option argument, enclose the entire value in single quotation marks (`'`).

  Example: `adbsql -u '"adbuser01"'`
- If the same option is specified more than once, only the last instance of that option is used.

  Example: `adbcancel -u 1 -u 2`

  In this example, only `-u 2` is used.

How to display a command usage message (how to display command help)

Specifying `-h` or `--help` displays a usage message that explains the command's input format.

Example:

```
adbcancel -h

KFAA91250-I Usage: adbcancel {--ALL | -u <connection-identifier>}
```

The following table explains the meanings of the symbols used in command usage messages.

Table 1-4: Meanings of the symbols used in command usage messages

No.	Symbol used in usage message	Meaning
1	{ }	Curly brackets indicate that only one of the enclosed options is to be selected.

No.	Symbol used in usage message	Meaning
2	|	A vertical bar separates multiple items, and has the meaning of OR.
3	[ ]	Square brackets indicate that the enclosed option or options can be omitted.
4	< >	Angle brackets enclose a character string that explains the type of option argument or command argument to specify.
5	-	A hyphen in a character string for an option argument or a command argument separates explanatory words.

1.4 Common command rules

1.4.1 Common rules

A file with the extension `BOM` cannot be used as an input file for commands. No file output as a result of executing a command will have the extension `BOM`.

1.4.2 Rules for specifying authorization identifiers and passwords

This section explains the rules for specifying authorization identifiers by using a command's `-u` option, and for specifying passwords by using a command's `-p` option.

(1) Rules for specifying authorization identifiers

- If the character string of an authorization identifier specified in the `-u` option includes a lowercase letter, enclose the authorization identifier in backslashes followed by double quotation marks (`\"`). Alternatively, enclose the entire authorization identifier between double quotation marks (`"`) and then between single quotation marks (`'`).

  Example 1: `-u \"ADBuser01\"`

  Example 2: `-u '"ADBuser01"'`

  If an authorization identifier is not specified as shown above, the character string used as the authorization identifier is treated as being in all uppercase letters. For example, `-u ADBuser01` is treated as `-u ADBUSER01`.

- If the character string of the authorization identifier specified in the `-u` option includes a backslash (`\`), specify the escape character (`\`) immediately before the backslash (`\`). Alternatively, enclose the entire authorization identifier in single quotation marks (`'`).

  Example 1: `-u ADBUSER\\01`

  Example 2: `-u 'ADBUSER\01'`

  In this example, `ADBUSER\01` is specified as the authorization identifier.

For details about the rules for specifying authorization identifiers, see *Authorization identifier specification rules* in the *HADB Setup and Operation Guide*.

(2) Rules for specifying passwords

- Unlike the authorization identifier specified in the `-u` option, the password specified in the `-p` option is case sensitive.

- If the password includes a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), specify the escape character (`\`) immediately before that character. Alternatively, enclose the entire password in single quotation marks (`'`).

  Example 1: `-p Password\|01`

  Example 2: `-p 'Password|01'`

  In this example, `Password|01` is specified as the password.

For details about the rules for specifying passwords, see *Password specification rules* in the *HADB Setup and Operation Guide*.

(3) Examples of specifying authorization identifiers and passwords

Character strings that include lowercase letters are specified differently between the `-u` and `-p` options. Example specifications (using the `adbarchivechunk` command) are as follows:

Example 1:

Specifying `ADBUSER01` as the authorization identifier and `PASSWORD01` as the password

```
adbarchivechunk -u ADBUSER01 -p PASSWORD01
```

Example 2:

Specifying `ADBuser01` as the authorization identifier and `Password01` as the password

```
adbarchivechunk -u \"ADBuser01\" -p Password01
```

Example 3:

Specifying `ADBuser\01` as the authorization identifier and `Password\01` as the password

```
adbarchivechunk -u \"ADBuser\\01\" -p Password\\01
```

1.4.3 Table name specification rules

This subsection explains the rules that apply to specifying a table name for a command argument.

(1) Specification rules that apply if a table name contains any lowercase letter

If the table name to be specified for a command argument contains any lowercase letter, use either of the following specification formats:

- Individually enclose the schema name and the table identifier between pairs of a backslash and a double quotation mark (`\"`).

  Example: If the table name is `ADBUser01.Table01`:

  `\"ADBUser01\".\"Table01\"`

- Individually enclose the schema name and the table identifier between double quotation marks (`"`), and then enclose the entire table name between single quotation marks (`'`).

  Example: If the table name is `ADBUser01.Table01`:

  `'"ADBUser01"."Table01"'`

If these specification rules are not followed, all characters in the schema name and table identifier are interpreted as uppercase letters. For example, if `ADBUser01.Table01` is specified, it is assumed that `ADBUSER01.TABLE01` is specified.

(2) Specification rules that apply if a table name contains a backslash (\)

If the table name to be specified for a command argument contains a backslash (`\`), use either of the following specification formats:

1. Enclose the entire table name between single quotation marks (`'`).
2. Specify the escape character (`\`) immediately before the backslash.

If the table name contains lowercase letters, use specification format 1.

Example 1: If the table name is `ADBUser01.Table\01`:

`'"ADBUser01"."Table\01"'`

Example 2: If the table name is `ADBUSER01.TABLE\01`:

`'ADBUSER01.TABLE\01'` or `ADBUSER01.TABLE\\01`

1.5 Warnings about messages output when a command is executed

When a command is executed, SQL statements executed in extension to the command might cause an error message starting with `KFAA3` to be output.

If an error message starting with `KFAA3` is output when a command is executed, resolve the issue as if *SQL statement* in the solution column of the output message says *command* instead. For details about the solution column of output messages, see the manual *HADB Messages*.

1.6 When suspending (forcibly terminating) an executing command

Use the `adbcancel` command to suspend an executing command. Using the **Ctrl**+**C** keys to forcibly terminate the command process will not suspend the command because command processing is continued by server processes.

The `adbcancel` command is used only to suspend a command (the `adbexport` command, for example) that has established a connection with the HADB server.

Note

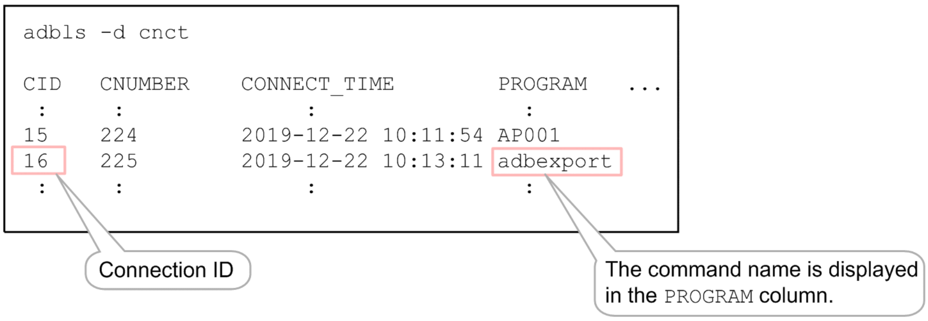
For details about the `adbcancel` command, see 4. adbcancel (Forcibly Terminate Transactions or Commands).

The following shows how to suspend the executing command (example of suspending the `adbexport` command).

To suspend the command:

1. Execute the `adbls -d cnct` command to check the connection ID.

   Check the connection ID of the executing `adbexport` command.



```
adbls -d cnct

CID   CNUMBER    CONNECT_TIME         PROGRAM   ...
 :     :              :                  :
15    224        2019-12-22 10:11:54 AP001
16    225        2019-12-22 10:13:11 adbexport
 :     :              :                  :
```

Connection ID

The command name is displayed in the `PROGRAM` column.

2. Execute the `adbcancel` command to suspend the `adbexport` command.

```
adbcancel -u 16
```

   For the `-u` option, specify the connection ID that you checked in step 1.

Note

For details about the `adbls -d cnct` command, see 23. adbls -d cnct (Display the Connection Status).

2 adbarchivechunk (Archive Chunk)

This chapter explains the functionality provided by the `adbarchivechunk` command and how to use the command.

2.1 Description

This section explains the adbarchivechunk command and the threads it uses.

When executed, the adbarchivechunk command establishes a connection to the HADB server.

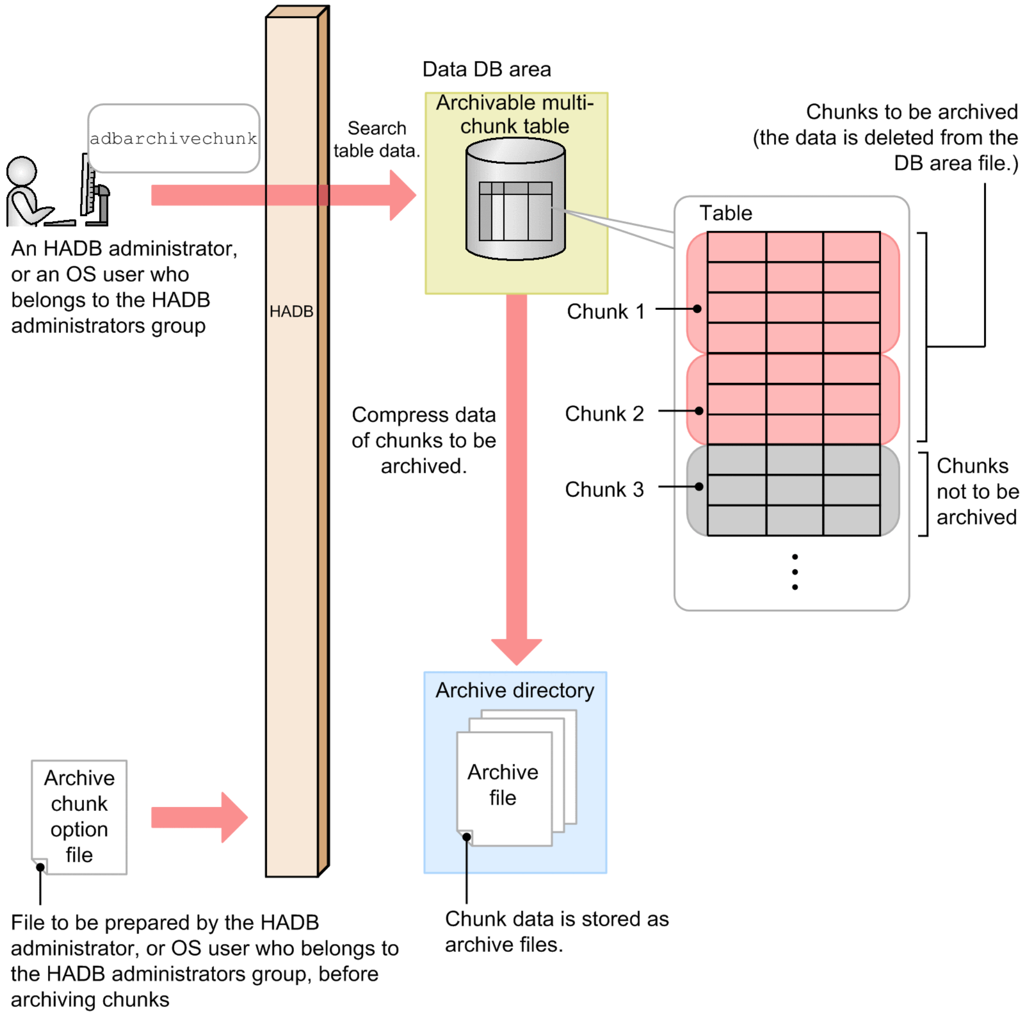
2.1.1 Functional overview

When the adbarchivechunk command is executed, table data can be archived (compressed) in units of chunks. The archiving of data in units of chunks is called *chunk archive processing*. In addition, tables whose data can be archived in units of chunks are called *archivable multi-chunk tables*.

When chunk archive processing is executed, the target chunk data is compressed and stored in a file (archive file). That data is also deleted from the DB area files.

The following figure provides an overview of chunk archive processing.

Figure 2-1: Overview of the adbarchivechunk command


adbarchivechunk
An HADB administrator, or an OS user who belongs to the HADB administrators group
HADB
Search table data.
Data DB area
Archivable multi-chunk table
Chunks to be archived (the data is deleted from the DB area file.)
Table
Chunk 1
Chunk 2
Chunk 3
Chunks not to be archived
Compress data of chunks to be archived.
Archive directory
Archive file
Archive chunk option file
File to be prepared by the HADB administrator, or OS user who belongs to the HADB administrators group, before archiving chunks
Chunk data is stored as archive files.

Explanation:

When the adbarchivechunk command is executed, data stored in the chunks specified in the archivable multi-chunk table is searched. That data is stored as an archive file in the archive directory.

After the archive file is stored, the specified chunk data is deleted from the DB area files.

Note that multiple chunks can be archived during one execution of the `adbarchivechunk` command. When multiple chunks are archived, the storage processing and deletion processing is repeated as many times as the number of chunks.

The following chunks can be archived (are subject to archiving) by the `adbarchivechunk` command.

▪ **Chunks that are subject to archiving**

- Chunks in normal status
  However, the current chunk cannot be archived.
- Chunks in wait status that have once been in normal status in the past

> **Important**
>
> A disadvantage of archived chunks is that search processing on their corresponding chunk data is slowed down. Make sure that you take into account the preceding point before deciding whether to use chunk archive processing.
>
> For details about archiving chunks, see *Chunk archiving function (compressing data in a chunk)* in the *HADB Setup and Operation Guide*.

2.1.2 Relationships among threads used by the adbarchivechunk command

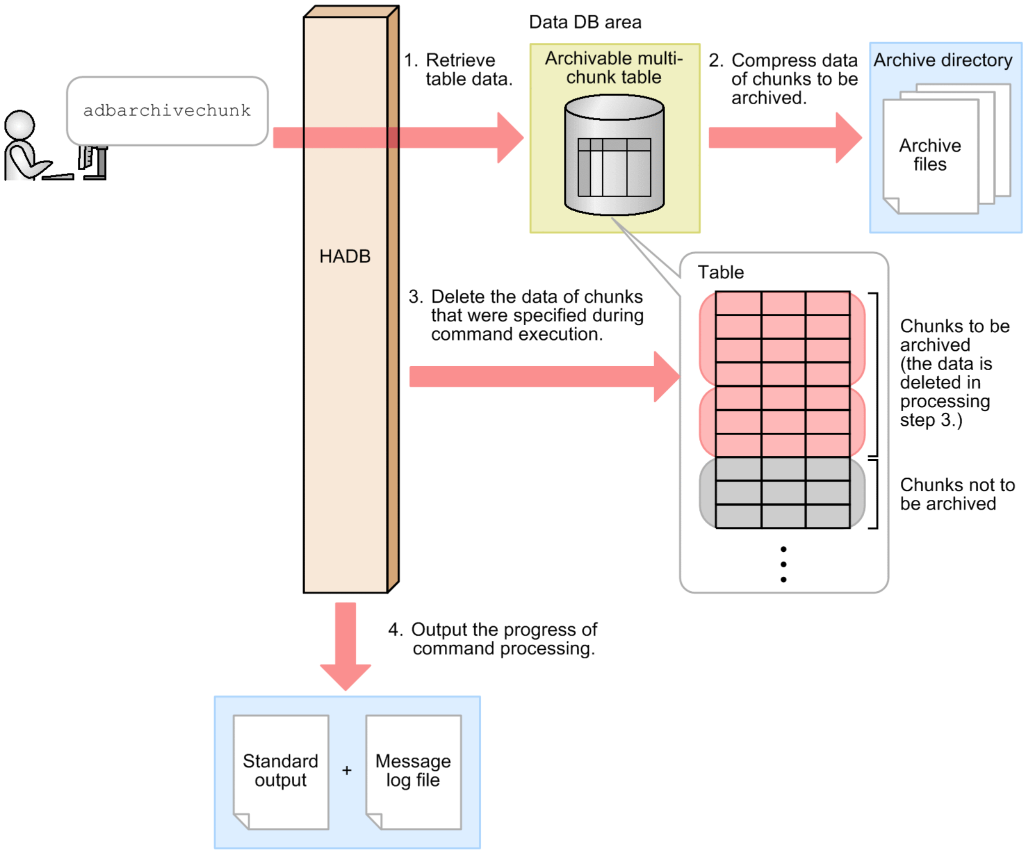
The `adbarchivechunk` command uses processing real threads in the following processing:

1. Retrieving table data
   One or more processing real threads are used.
2. Compressing data of chunks to be archived
   One or more processing real threads are used.
3. Deleting the data of chunks that were specified when the command was executed
   One or more processing real threads are used.
4. Outputting the progress of command processing
   One processing real thread is used.

Steps 1 and 2 are performed at the same time. Then, step 3 is performed. Step 4 is always performed while steps 1 to 3 are in progress.

The following figure shows the processing of the `adbarchivechunk` command using processing real threads. The preceding numbers correspond to the numbers in the figure.

Figure 2-2: Processing of the adbarchivechunk command using processing real threads


adbarchivechunk
HADB
Data DB area
1. Retrieve table data.
Archivable multi-chunk table
2. Compress data of chunks to be archived.
Archive directory
Archive files
Table
3. Delete the data of chunks that were specified during command execution.
Chunks to be archived (the data is deleted in processing step 3.)
Chunks not to be archived
4. Output the progress of command processing.
Standard output
+
Message log file

The number of processing real threads used by the `adbarchivechunk` command for each step is automatically calculated from the *number of processing real threads to be used by the* `adbarchivechunk` *command*. For details about the *number of processing real threads used by the* `adbarchivechunk` *command*, see the description of the archive chunk option adb_arcv_rthd_num in (2) Explanation of specification format in 2.2.2 Format of archive chunk options.

2.2 Specification format for the adbarchivechunk command

This section explains the specification format for the `adbarchivechunk` command and how to specify the archive chunk options.

The following OS users can execute the `adbarchivechunk` command:

- HADB administrator
- OS users who belong to the HADB administrators group

2.2.1 Explanation of the specification format and options

(1) Specification format

```
adbarchivechunk
  -u authorization-identifier
  [-p password]
  [-g output-interval-for-chunk-archive-processing-progress-messages]
  [-z archive-chunk-option-file-name]
  [-t]
  {-c chunk-ID-specification[,chunk-ID-specification]...|-r range-specification}
  table-to-be-processed
```

Important

You must specify the table to be processed as the last option that is specified.

(2) Explanation of options

- `-u` *authorization-identifier*

  ~<character string>((1 to 100 bytes))

  Specifies the authorization identifier of the HADB user who executes the `adbarchivechunk` command. To execute the `adbarchivechunk` command, the user must have the following two privileges:

  - `CONNECT` privilege
  - `ARCHIVE CHUNK` privilege for the table to be processed

  For this option, specify an authorization identifier in the range from 1 to 100 bytes. Note that the byte count (1 to 100 bytes) does not include the double quotation marks used to enclose the authorization identifier.

  Important

  If the character string used as the authorization identifier includes any lowercase letter or a backslash (\), make sure that you check the rules for specifying authorization identifiers. For the rules for specifying authorization identifiers, see 1.4.2 Rules for specifying authorization identifiers and passwords.

- `-p` *password*

  ~<character string>((1 to 255 bytes))

Specifies the password for the authorization identifier that was specified in the `-u` option.

> **Important**
>
> If the password includes a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), make sure that you check the rules for specifying passwords. For the rules for specifying passwords, see 1.4.2 Rules for specifying authorization identifiers and passwords.

If the `adbarchivechunk` command is executed with this option omitted, a message prompting the user to enter a password is displayed. In an environment in which a password cannot be entered from the standard input, such as when the command is executed in the background, make sure that you do not omit this option.

- `-g` *output-interval-for-chunk-archive-processing-progress-messages*

  ~<integer>((0 to 1,000,000,000))<<0>>(in 1,000 rows)

  Specifies the interval at which to output progress messages for chunk archive processing.

  For example, if `2` is specified in this option, the `KFAA80205-I` progress message is output each time 2,000 rows of chunk are archived.

  Furthermore, when chunk archive processing is performed on multiple chunks, progress messages are output for each chunk. For example, if `2` is specified in this option, and three chunks are undergoing chunk archive processing, the `KFAA80205-I` progress message is output each time 2,000 rows of data are archived for each of the three chunks. You can tell from the progress message which chunk's processing it refers to.

  If this option is omitted or `0` is specified in this option, no progress message is output.

- `-z` *archive-chunk-option-file-name*

  ~<OS path name>((2 to 510 bytes))

  Specifies the absolute path name of the archive chunk option file that contains the archive chunk options. If this option is omitted, all archive chunk options are assumed to be their default values because no archive chunk option file is specified.

  For details about the archive chunk options, see 2.2.2 Format of archive chunk options.

- `-t`

  Specify this option if you want to check the chunks to be archived before actually executing chunk archive processing.

  If you specify this option, the chunk IDs of chunks subject to archiving that correspond to the range specified in the `-c` option or the `-r` option are output in the `KFAA80245-I` message.

  To perform chunk archive processing on these chunks, do not change any options other than the `-t` option, and execute the `adbarchivechunk` command with the `-t` option omitted. To revise which chunks are to be archived, correct the specified values in the `-c` option or the `-r` option, and then execute the `adbarchivechunk` command.

  > **Important**
  >
  > If you specify this option, chunk archive processing is not executed.

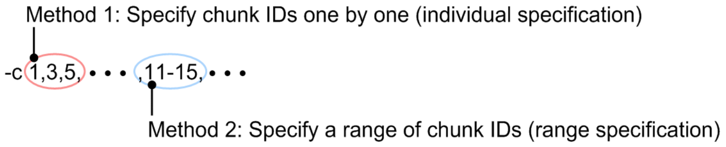
- `-c` *chunk-ID-specification* `[,` *chunk-ID-specification* `]...`

  ~<integer>((1 to 9,223,372,036,854,775,807))

  Specify the chunk IDs of chunks to be archived among those in the table to be processed. You can specify one or more chunk IDs.

  You can specify chunk IDs as follows. Note that you can combine multiple methods of specifying chunk IDs.

Figure 2-3: Example of specifying chunk IDs in the -c option


Method 1: Specify chunk IDs one by one (individual specification)
-c 1,3,5, • • • ,11-15, • • •
Method 2: Specify a range of chunk IDs (range specification)

- **Specifying individual chunk IDs (individual specification)**
  If you specify the IDs of individual chunks, delimit the chunk IDs by using commas (`,`).

> **Important**
>
> There must be no spaces before or after a comma (`,`).

- **Specifying a range of chunk IDs (range specification)**
  If you specify a range of chunk IDs, use the format *smallest-chunk-ID-in-the-range*`-`*largest-chunk-ID-in-the-range* (connect the two chunk IDs with a hyphen (`-`)).

> **Important**
>
> There must be no spaces before or after the hyphen (`-`). The chunk ID specified to the right of the hyphen must be greater than the chunk ID specified to the left of the hyphen.

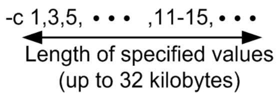
Among the chunks whose IDs are specified, the following chunks will not be archived (only chunks in the table to be processed that are subject to archiving will be archived):

- Chunks that are not in the table to be processed
- Chunks that are not subject to archiving

For details about chunks that are subject to archiving, see 2.1.1 Functional overview.

The rules for specifying this information are as follows:

- You can specify maximum of 30,000 chunk IDs. The following explains how chunk IDs are counted:
  • For a range specification, all chunk IDs in the specified range are counted.
  • If any duplicate chunk IDs are specified, the duplicate chunk IDs are counted.
  Examples of specifying chunk IDs and how to count them
  Example 1: `-c 1,3,5`: The command assumes that three chunk IDs are specified (`1`, `3`, and `5`).
  Example 2: `-c 1,3,5-8,10`: The command assumes that seven chunk IDs are specified (`1`, `3`, `5`, `6`, `7`, `8`, and `10`).
  Example 3: `-c 1,3,5,5-8,10`: The command assumes that eight chunk IDs are specified (`1`, `3`, `5`, `5`, `6`, `7`, `8`, and `10`).
- The maximum permitted length of the entire value specified in the `-c` option is 32 kilobytes. If the length would exceed 32 kilobytes, use range specifications or execute the `adbarchivechunk` command more than once.



- For details about specifications that result in an error and specification examples, see (3) Errors that can be caused by the -c option.

▪ **Searching for chunk IDs of chunks that are in a table to be processed**

The following explains how to determine the chunk IDs of chunks that are in a table to be processed:

- Searching system tables for `STATUS_CHUNKS`

    Search system tables for STATUS_CHUNKS, and check the chunk IDs.

    For details, see *Checking the information about all chunks in a table based on a table name* in *Searching system tables* in *System Tables* in the *HADB Setup and Operation Guide*.

  - Executing the adbdbstatus command

    Execute the following command to output the usage information for DB areas, tables, and indexes, and then check the chunk IDs (Chunk_ID) (the results are output in CSV format). For details about the items that are output as usage information for DB areas, tables, and indexes, see 13.7.2 List of items that are output as usage information for DB areas, tables, and indexes.

    ```
    adbdbstatus -d used -c table -n name-of-table-to-be-processed
    ```

    Specify the table name in the format *schema-name*.*table-identifier*.

- -r *range-specification*

  Specify a range of chunks to be archived. For details about the -r option, see (4) Explanation of the -r option (range specification).

- *table-to-be-processed*

  Specify the target archivable multi-chunk table for chunk archive processing.

  The rules for specifying this information are as follows:

  - Specify in the format *schema-name*.*table-identifier*.

    If the user owns this table, the schema name can be omitted. If another HADB user owns this table, schema name must be specified.

  - For details about the specification rules that apply when a schema name or table identifier includes any lowercase letter or backslash (\), see 1.4.3 Table name specification rules.

  - Tables other than archivable multi-chunk tables cannot be specified.

  - The user can specify only a table for which the user has the ARCHIVE CHUNK privilege.

  - Specify a table that can be updated (that is not non-updatable). Specifying a non-updatable table results in an error.

  - A viewed table cannot be specified.

(3) Errors that can be caused by the -c option

The following are two possible causes of errors in the adbarchivechunk command that can be caused by the -c option's specification:

- The specification format of the -c option is invalid.
- An invalid chunk ID is specified in the -c option.

In either case, correct the specification, and then re-execute the adbarchivechunk command. For details about the -c option, see -c in (2) Explanation of options.

(a) When the specification format of the -c option is invalid

The following table lists the message IDs that are displayed when the specification format of the -c option is invalid, and describes the possible errors. If any of these errors occurs, correct the error, and then re-execute the adbarchivechunk command.

Table 2-1: Message IDs that are displayed when the specification format of the -c option is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	`KFAA50225-E`	The number of specified chunk IDs exceeds 30,000.	`-c 1-30001`	`-c 1-30000`
2		The same chunk ID is specified on both sides of a hyphen (-).	`-c 3-3,7`	`-c 3-5,7`
3		The value of the chunk ID on the right side of a hyphen (-) is less than the value of the chunk ID on the left side.	`-c 10-3`	`-c 3-10`
4	• `KFAA90002-E` • `KFAA96830-I`	There is a space between a chunk ID and a comma.	`-c 1,3,Δ10`	`-c 1,3,10`
5		There is a space between a chunk ID and a hyphen (-).	`-c 1,3Δ-Δ10`	`-c 1,3-10`
6	`KFAA90003-E`	The length of the specified value exceeds 32 kilobytes.	-c 1,2, • • • ,xxxxx ⟷ 33 kilobytes or more	`-c 1-`*xxxxx* Change from individual specification to range specification (use a hyphen (-), not a comma (,)).

Legend:

Δ: Space

(b) When an invalid chunk ID is specified in the -c option

The following table lists the message IDs that are displayed when an invalid chunk ID is specified in the `-c` option, and describes the possible errors. If any of these errors occurs, correct the error, and then re-execute the `adbarchivechunk` command.

Table 2-2: Message IDs that are displayed when an invalid chunk ID is specified in the -c option, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification and corrective action to take
1	`KFAA50284-E`	None of the specified chunk IDs is in the table to be processed.	**If only chunk IDs 2 and 6 are in the table to be processed** • `-c 3-5` • `-c 4,5`	**If only chunk IDs 2 and 6 are in the table to be processed** • `-c 2,6` • `-c 2-6` Specify the chunk IDs of chunks in the table to be processed that are subject to archiving.

Note:

For details about how to search for the IDs of chunks in the table to be processed, see -c in (2) Explanation of options.

(4) Explanation of the -r option (range specification)

Use the `-r` option to specify a range of chunks to be archived. If the specified range contains any part of the range of values in the archive range column (the column used to narrow down the search range when searching the archivable multi-chunk table), the corresponding chunks are archived. However, among the chunks where data in the specified range is stored, chunks that are not subject to archiving are not archived. For details about chunks that are subject to archiving, see 2.1.1 Functional overview.

The range specification is interpreted based on the data type in the archive range column.

The following table lists the types of range specifications and links to their detailed explanations.

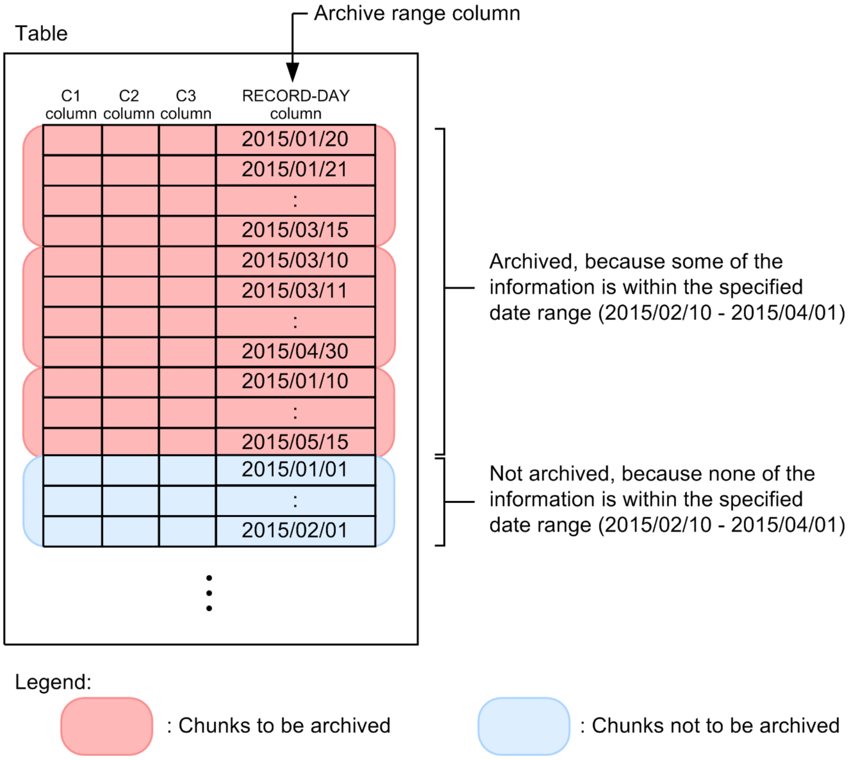
Table 2-3: Types of range specifications and their links

No.	Type of range specification	Link
1	Date range specification	(a) Specifying ranges of dates
2	Time range specification	(b) Specifying ranges of times
3	Time stamp range specification	(c) Specifying ranges of time stamps
4	Fixed-length character string range specification	(d) Specifying ranges of fixed-length character strings
5	Decimal range specification	(e) Specifying ranges of decimal numbers
6	Integer range specification	(f) Specifying ranges of integers
7	Floating point range specification	(g) Specifying ranges of floating point numbers

This section shows which chunks are archived, using the following conditions as an example:

- The data type in the archive range column is `DATE`.
- The date range `2015/02/10` - `2015/04/01` is specified in the range specification.

Figure 2-4: Relationship between the content of the date range specification and the chunks to be archived


Archive range column
Table
C1 column
C2 column
C3 column
RECORD-DAY column
2015/01/20
2015/01/21
:
2015/03/15
2015/03/10
2015/03/11
:
2015/04/30
2015/01/10
:
2015/05/15
2015/01/01
:
2015/02/01
Archived, because some of the information is within the specified date range (2015/02/10 - 2015/04/01)
Not archived, because none of the information is within the specified date range (2015/02/10 - 2015/04/01)
Legend:
: Chunks to be archived
: Chunks not to be archived

The rules for specifying the `-r` option are as follows:

- You can only specify one range. Multiple ranges cannot be specified.
- If 1,024 bytes or more are specified in the range specification, the `KFAA90003-E` message is output.
- In this option, specify the value that results after characters such as shell enclosing characters have been parsed. The specified value and the actual interpreted value might differ depending on how the shell parses the characters. The following is an example of specifying a range of fixed-length character strings.

  Option specification value: `-r 'abc'-'def'`

  Actual interpreted value: `-r abc-def`

  Single quotation marks (`'`) have been omitted from the actual interpreted value.

(a) Specifying ranges of dates

Use dates to specify a range of chunks to be archived.

Format

```
-r {YYYY-MM-DD|YYYY/MM/DD}-{YYYY-MM-DD|YYYY/MM/DD}
```

Connect the minimum value (date specification) and maximum value (date specification) by using a hyphen (-). There must be no spaces between the minimum value and the hyphen (-), or between the maximum value and the hyphen (-).

Explanation

YYYY: Year (`0001-9999`)

MM: Month (`01-12`)

DD: Day (`01` to the last day of the month specified in *MM*)

Specification example

Specification example 1: `-r 2016/01/01-2016/12/31`

Specification example 2: `-r 2016-01-01-2016-12-31`

Specify each date in either of the following formats: *YYYY*`-`*MM*`-`*DD* or *YYYY*`/`*MM*`/`*DD* (delimit *YYYY*, *MM*, and *DD* using forward slashes (`/`) or hyphens (`-`)).

In the case of specification example 1, `2016/01/01` and `2016/12/31` are treated as date specifications.

In the case of specification example 2, `2016-01-01` and `2016-12-31` are treated as date specifications.

The rules for specifying ranges of dates are as follows:

- If there are not enough digits in the values you want to specify in *YYYY*, *MM*, or *DD*, pad the left side of the value with zeros (`0`).
- The data types in the archive range column for which you can specify a date range are as follows:
  - `DATE`
  - `TIMESTAMP`
- Only dates can be specified in a date range specification. If the data type in the archive range column is `TIMESTAMP`, specified values are treated as if the time has been omitted. Therefore, the start time and end time in a specified value are corrected as shown in the following table. As a result, all chunks containing data in the specified date range are subject to archiving.

Table 2-4: Correction of the start time and end time

No.	Number of digits in the fractional seconds defined in the TIMESTAMP-type column	Start time	End time
1	`0`	0 hours 0 minutes 0 seconds	23 hours 59 minutes 59 seconds
2	`3`	0 hours 0 minutes 0.000 seconds	23 hours 59 minutes 59.999 seconds
3	`6`	0 hours 0 minutes 0.000000 seconds	23 hours 59 minutes 59.999999 seconds
4	`9`	0 hours 0 minutes 0.000000000 seconds	23 hours 59 minutes 59.999999999 seconds
5	`12`	0 hours 0 minutes 0.000000000000 seconds	23 hours 59 minutes 59.999999999999 seconds

- The following table lists the message IDs that are displayed when the specified date range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then execute the `adbarchivechunk` command.

Table 2-5: Message IDs that are displayed when the specified date range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	`KFAA50225-E`	• Too many digits in the date • Not enough digits in the date	`-r 2015/7/1-2015/008/031`	`-r 2015/07/01-2015/08/31`
2		A date outside the range is specified.	`-r 2015/01/01-2015/06/31`	`-r 2015/01/01-2015/06/30`
3	• `KFAA90002-E` • `KFAA96830-I`	The range specification format is invalid.	**Specification example** `-r 2015/01/11Δ-Δ2015/7/31`	`-r 2015/01/11-2015/07/31`

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
			Explanation Because there is a space between the date specification and the hyphen (-), the shell erroneously uses spaces to delimit the option arguments.	

Legend:

Δ: Space

(b) Specifying ranges of times

Use times to specify a range of chunks to be archived.

Format

```
-r hh:mm:ss[.[nn...n]]-hh:mm:ss[.[nn...n]]
```

Connect the minimum value (time specification) and maximum value (time specification) by using a hyphen (-). There must be no spaces between the minimum value and the hyphen (-), or between the maximum value and the hyphen (-).

Explanation

hh:: Hours (`00-23`)

mm: Minutes (`00-59`)

ss: Seconds (`00-59`)

`.`: Decimal point

nn...n: Fractional seconds (*n* is `0-9`)

Specification example

Specification example 1: `-r 00:00:00-12:59:59`

Specification example 2: `-r 00:00:00.000-12:59:59.999`

The *hh*, *mm*, and *ss* values of each time are delimited by using colons (`:`).

When specifying fractional seconds, connect the *ss* and *nn...n* values by using a decimal point (`.`).

In the case of specification example 1, `00:00:00` and `12:59:59` are treated as time specifications.

In the case of specification example 2, `00:00:00.000` and `12:59:59.999` are treated as time specifications.

The rules for specifying ranges of times are as follows:

- If there are not enough digits in the values you want to specify in *hh*, *mm*, or *ss*, pad the left side of the value with zeros (`0`).
- The data type in the archive range column for which you can specify a time range is as follows:
  - `TIME`
- If the fractional seconds have more digits than the fractional seconds defined in the `TIME`-type column, the excess digits are truncated.
- If fractional seconds are omitted, the start time and end time are corrected to reflect the number of digits in the fractional seconds defined in the `TIME`-type column. The following table lists details about how this correction is made.

Table 2-6: Correction of the start time and end time when fractional seconds are omitted

No.	Number of digits in the fractional seconds defined in the TIME-type column	Start time	End time
1	3	.000 seconds	.999 seconds
2	6	.000000 seconds	.999999 seconds
3	9	.000000000 seconds	.999999999 seconds
4	12	.000000000000 seconds	.999999999999 seconds

- For details about comparing the minimum and maximum values for time specifications, see *Comparing datetime data* in *Data types that can be compared* in *Data types that can be converted, assigned, and compared* in the manual *HADB SQL Reference*.
- The following table lists the message IDs that are displayed when the specified time range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then execute the `adbarchivechunk` command.

Table 2-7: Message IDs that are displayed when the specified time range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	`KFAA50225-E`	• Too many digits in the time • Not enough digits in the time	`-r 0:0:0-012:59:59`	`-r 00:00:00-12:59:59`
2		A time outside the range is specified.	`-r 00:00:00-12:60:00`	`-r 00:00:00-12:59:59`
3	• `KFAA90002-E` • `KFAA96830-I`	The range specification format is invalid.	**Specification example** `-r 00:00:00Δ-Δ12:59:59` **Explanation** Because there is a space between the time specification and the hyphen (`-`), the shell erroneously uses spaces to delimit the option arguments.	`-r 00:00:00-12:59:59`

Legend:

Δ: Space

(c) Specifying ranges of time stamps

Use time stamps to specify a range of chunks to be archived.

Format

```
-r '{YYYY-MM-DD|YYYY/MM/DD}Δhh:mm:ss[.[nn...n]]'
    -'{YYYY-MM-DD|YYYY/MM/DD}Δhh:mm:ss[.[nn...n]]'
```

Connect the minimum value (time stamp specification) and maximum value (time stamp specification) by using a hyphen (`-`). There must be no spaces between the minimum value and the hyphen (`-`), or between the maximum value and the hyphen (`-`).

Explanation

YYYY: Year (`0001-9999`)

MM: Month (`01`-`12`)

DD: Day (`01` to the last day of the month specified in *MM*)

Δ: Halfwidth space or tab

hh:: Hours (`00`-`23`)

mm: Minutes (`00`-`59`)

ss: Seconds (`00`-`59`)

.: Decimal point

nn...n: Fractional seconds (*n* is `0`-`9`)

Specification example

Specification example 1: `-r '2016/01/01 00:00:00'-'2016/12/31 12:59:59'`

Specification example 2: `-r '2016/01/01 00:00:00.000'-'2016/12/31 12:59:59.999'`

Connect the date and time of each specified time stamp by using a halfwidth space, and enclose entire date or time values in single quotation marks (`'`). When specifying dates, use either of the following formats: *YYYY*`-`*MM*`-`*DD* or *YYYY*`/`*MM*`/`*DD* (delimit *YYYY*, *MM*, and *DD* using forward slashes (`/`) or hyphens (`-`)). When specifying times, delimit the *hh*, *mm*, and *ss* values using colons (`:`). When specifying fractional seconds, connect the *ss* and *nn...n* values by using a decimal point (`.`).

In the case of specification example 1, `2016/01/01 00:00:00` and `2016/12/31 12:59:59` are treated as time stamp specifications.

In the case of specification example 2, `2016/01/01 00:00:00.000` and `2016/12/31 12:59:59.999` are treated as time stamp specifications.

The rules for specifying ranges of time stamps are as follows:

- If there are not enough digits in the values you want to specify in *YYYY*, *MM*, *DD*, *hh*, *mm*, or *ss*, pad the left side of the value with zeros (`0`).
- The data type in the archive range column for which you can specify a time stamp range is as follows:
  - `TIMESTAMP`
- If the fractional seconds have more digits than the fractional seconds defined in the `TIMESTAMP`-type column, the excess digits are truncated.
- If fractional seconds are omitted, the start time and end time are corrected to reflect the number of digits in the fractional seconds defined in the `TIMESTAMP`-type column. The following table shows details about how this correction is made.

Table 2-8: Correction of the start time and end time when fractional seconds are omitted

No.	Number of digits in the fractional seconds defined in the TIMESTAMP-type column	Start time	End time
1	`3`	.000 seconds	.999 seconds
2	`6`	.000000 seconds	.999999 seconds
3	`9`	.000000000 seconds	.999999999 seconds
4	`12`	.000000000000 seconds	.999999999999 seconds

- For details about comparing the minimum and maximum values for time stamp specifications, see *Comparing datetime data* in *Data types that can be compared* in *Data types that can be converted, assigned, and compared* in the manual *HADB SQL Reference*.

- The following table lists the message IDs that are displayed when the specified time stamp range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then execute the `adbarchivechunk` command.

Table 2-9: Message IDs that are displayed when the specified time stamp range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	`KFAA50225-E`	• Too many digits in the time stamp • Not enough digits in the time stamp	`-r '2015/7/1Δ00:00:00'-'2015/008/031Δ12:59:59'`	`-r '2015/07/01Δ00:00:00'-'2015/08/31Δ12:59:59'`
2		A time stamp outside the range is specified.	`-r '2015/01/01Δ00:00:00'-'2015/06/31Δ12:60:00'`	`-r '2015/01/01Δ00:00:00'-'2015/06/30Δ12:59:59'`
3	• `KFAA90002-E` • `KFAA96830-I`	The range specification format is invalid.	**Specification example** `-r '2016/04/01Δ00:00:00'Δ-Δ'2016/06/30Δ12:59:59'` **Explanation** Because there is a space between the time stamp specification and the hyphen (`-`), the shell erroneously uses spaces to delimit the option arguments.	`-r '2016/04/01Δ00:00:00'-'2016/06/30Δ12:59:59'`
4			**Specification example** `-r 2016/04/01Δ00:00:00-2016/06/30Δ12:59:59` **Explanation** Because the time stamp specification is not enclosed in single quotation marks (`'`), the shell sees the space between the date and time values, and erroneously uses spaces to delimit the option arguments.	

Legend:

Δ: Space

(d) Specifying ranges of fixed-length character strings

Use fixed-length character strings to specify a range of chunks to be archived.

Format

```
-r a...a-a...a
```

Connect the minimum value (fixed-length character string specification) and maximum value (fixed-length character string specification) by using a hyphen (`-`). There must be no spaces between the minimum value and the hyphen (`-`), or between the maximum value and the hyphen (`-`).

Explanation

a...a: String of one or more characters

Specification example

```
-r 20160101-20161231
```

`20160101` and `20161231` are treated as fixed-length character string specifications.

The rules for specifying ranges of fixed-length character strings are as follows:

- Specify a fixed-length character string of one or more characters.
- Fixed-length character string specifications are case sensitive.
- The data type in the archive range column for which you can specify a fixed-length character string range is as follows:
    - `CHARACTER`
- If the fixed-length character string includes a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), specify the escape character (`\`) immediately before that character. Alternatively, enclose the entire fixed-length character string in single quotation marks (`'`).

  Specification example

  Specification example 1: `-r a\|c-def`

  Specification example 2: `-r 'a|c'-def`

  In both examples, *`a|c`* and `def` are treated as fixed-length character string specifications.
- If a character string is enclosed in double quotation marks (`"`) and then the entire string is enclosed in single quotation marks (`'`), both the character string and the double quotation marks (`"`) are treated as a fixed-length character string.

  Specification example

  ```
  -r '"abc"'-def
  ```

  `"abc"` and `def` are treated as fixed-length character string specifications.
- If you specify a fixed-length character string that includes any hyphens (`-`), enclose the entire fixed-length character string in single quotation marks (`'`). This is to differentiate hyphens (`-`) in the specification from hyphens used to separate the minimum and maximum values.

  Specification example

    - When enclosing a specification in `\'` (backslashes and single quotation marks)

      ```
      -r \'a-c\'-def
      ```
    - When first enclosing a specification in single quotation marks (`'`), and then enclosing the entire string in double quotation marks (`"`)

      ```
      -r "'a-c'"-def
      ```

  In both examples, `a-c` and `def` are treated as fixed-length character string specifications.

  In addition, even if a specified fixed-length character string does not include any hyphens (`-`), if you enclose the specification in single quotation marks (`'`), the enclosed value is treated as the fixed-length character string specification, with the single quotation marks (`'`) omitted.

  Specification example

    - When enclosing a specification in `\'` (backslashes and single quotation marks)

      ```
      -r \'abc\'-def
      ```

      `abc` and `def` are treated as fixed-length character string specifications.
- If a specified fixed-length character string includes any single quotation marks (`'`), specify two single quotation marks (`'`) in succession to express one single quotation mark (`'`).

  Specification example

- When specifying two `\'` (backslashes and single quotation marks)

  `-r a\'\'c-def`

- When enclosing a specification that includes a single quotation mark (`'`), in double quotation marks (`"`)

  `-r "a''c"-def`

In both examples, `a'c` and `def` are treated as fixed-length character string specifications.

- For details about comparing the minimum and maximum values for fixed-length character string specifications, see *Comparing character string data* in *Data types that can be compared* in *Data types that can be converted, assigned, and compared* in the manual *HADB SQL Reference*.
- The following table lists the message IDs that are displayed when the specified fixed-length character string range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then execute the `adbarchivechunk` command.

Table 2-10: Message IDs that are displayed when the specified fixed-length character string range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification

1	`KFAA50225-E`	The fixed-length character string specification includes multiple hyphens (`-`) that are used to separate the minimum and maximum values.	`-r a-c-def`
			`-r 'a-c-def'`
				`-r \'a-c\'-def`

2	• `KFAA90002-E` • `KFAA96830-I`	The range specification format is invalid.	**Specification example** `-r abcΔ-Δdef` **Explanation** Because there is a space between the fixed-length character string specification and the hyphen (`-`), the shell erroneously uses spaces to delimit the option arguments.	`-r abc-def`

Legend:

Δ: Space

(e) Specifying ranges of decimal numbers

Use decimal numbers to specify a range of chunks to be archived.

Format

```
-r [{+|-}]{a...a[.[b...b]]|.b...b}
    -[{+|-}]{a...a[.[b...b]]|.b...b}
```

Connect the minimum value (decimal specification) and maximum value (decimal specification) by using a hyphen (`-`). There must be no spaces between the minimum value and the hyphen (`-`), or between the maximum value and the hyphen (`-`).

Explanation

+, -: Signs (the "+" sign can be omitted)

a...a: Integer part (*a* is `0`-`9`)

`.`: Decimal point

b...b: Fractional part (*b* is `0`-`9`)

Example of specifying DECIMAL(5,2)

Specification example 1: `-r -100-100`

Specification example 2: `-r -100.00-+100.00`

`-100.00` and `+100.00` are treated as decimal specifications.

The rules for specifying ranges of decimal numbers are as follows:

- Either the integer part or the fractional part must be specified. If there is no fractional part, the decimal point can be omitted.
- The data type in the archive range column for which you can specify a decimal range is as follows:
  - `DECIMAL`
- If the fractional parts have more digits than the fractional parts defined in the `DECIMAL`-type column, the excess digits are truncated.
- The following table lists the message IDs that are displayed when the specified decimal range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then execute the `adbarchivechunk` command.

Table 2-11: Message IDs that are displayed when the specified decimal range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	`KFAA50225-E`	A numerical value has been specified that cannot be used with the data type in the archive range column.	**Specification example** `-r 0-123456789012345678901234567890123456789` **Explanation** A decimal specification contains 39 digits.	**Specification example** `-r 0-12345678901234567890123456789012345678` **Explanation** Specify a decimal using 38 digits.
2	• `KFAA90002-E` • `KFAA96830-I`	The range specification format is invalid.	**Specification example** `-r 0Δ-Δ1` **Explanation** Because there is a space between the decimal specification and the hyphen (`-`), the shell erroneously uses spaces to delimit the option arguments.	`-r 0-1`

Legend:

Δ: Space

(f) Specifying ranges of integers

Use integers to specify a range of chunks to be archived.

Format

```
-r [{+|-}]a...a-[{+|-}]a...a
```

Connect the minimum value (integer specification) and maximum value (integer specification) by using a hyphen (`-`). There must be no spaces between the minimum value and the hyphen (`-`), or between the maximum value and the hyphen (`-`).

Explanation

+, -: Signs (the "+" sign can be omitted)

a...*a*: Numerical value (*a* is `0`-`9`)

Specification example

Specification example 1: `-r -100-100`

Specification example 2: `-r -100-+100`

`-100` and `+100` are treated as integer specifications.

The rules for specifying ranges of integers are as follows:

- The data types in the archive range column for which you can specify an integer range are as follows:
  - `INTEGER`
  - `SMALLINT`
- The following table lists the message IDs that are displayed when the specified integer range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then execute the `adbarchivechunk` command.

Table 2-12: Message IDs that are displayed when the specified integer range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification

1	`KFAA50225-E`	A numerical value has been specified that cannot be used with the data type in the archive range column.	**In the case of INTEGER**
			`-r 0-9223372036854775808`
			In the case of SMALLINT
			`-r 0-2147483648`
				In the case of INTEGER
				`-r 0-9223372036854775807`
				In the case of SMALLINT
				`-r 0-2147483647`

2	• `KFAA90002-E` • `KFAA96830-I`	The range specification format is invalid.	**Specification example** `-r 0Δ-Δ1` **Explanation** Because there is a space between the integer specification and the hyphen (-), the shell erroneously uses spaces to delimit the option arguments.	`-r 0-1`

Legend:

Δ: Space

(g) Specifying ranges of floating point numbers

Use floating point numbers to specify a range of chunks to be archived.

Format

```
-r [{+|-}]{a...a[.[b...b]]|.b...b}[{E|e}[{+|-}]c...c]
    - [{+|-}]{a...a[.[b...b]]|.b...b}[{E|e}[{+|-}]c...c]
```

Connect the minimum value (floating point specification) and maximum value (floating point specification) by using a hyphen (-). There must be no spaces between the minimum value and the hyphen (-), or between the maximum value and the hyphen (-).

Explanation

+, -: Signs (the "+" sign can be omitted) *a*...*a*: Integer part of the mantissa (*a* is `0`-`9`)

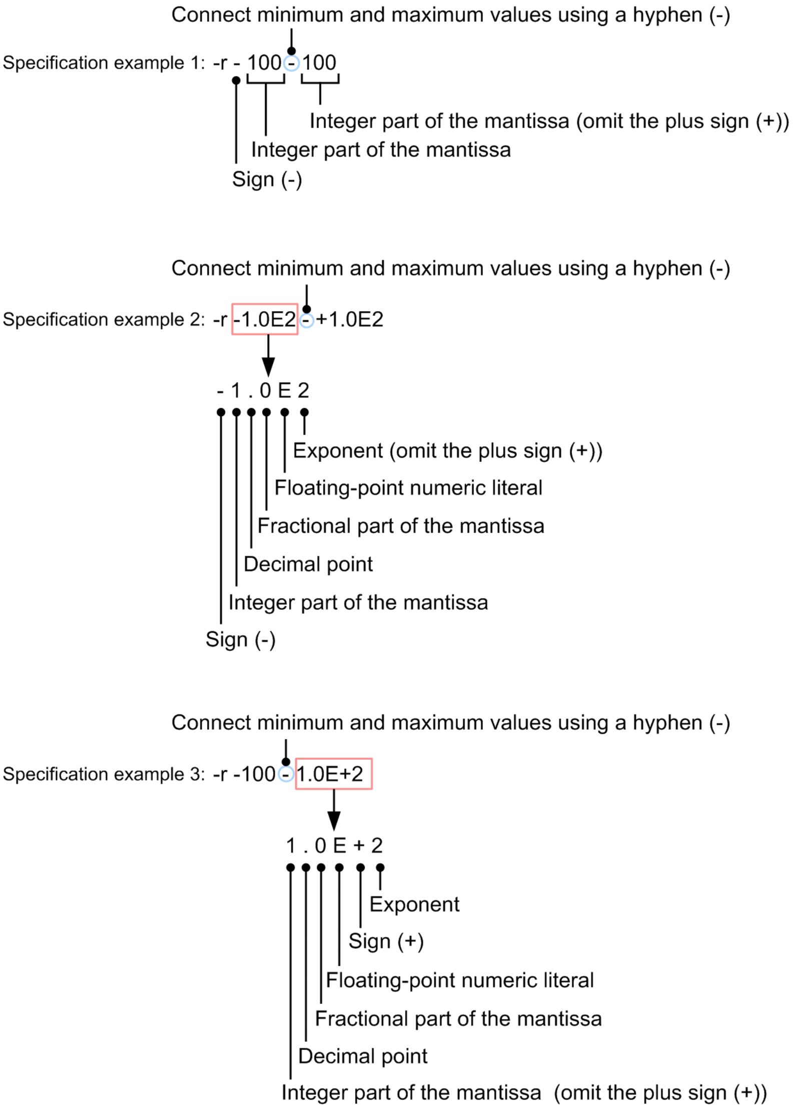
`.`: Decimal point

b...*b*: Fractional part of the mantissa (*b* is `0`-`9`)

`E`, `e`: Floating-point numeric literal

c...*c*: Exponent (*c* is `0`-`9`)

Figure 2-5: Example of specifying a floating point range by using the -r option


Connect minimum and maximum values using a hyphen (-)
Specification example 1: -r - 100-100
Integer part of the mantissa (omit the plus sign (+))
Integer part of the mantissa
Sign (-)
Connect minimum and maximum values using a hyphen (-)
Specification example 2: -r -1.0E2-+1.0E2
- 1 . 0 E 2
Exponent (omit the plus sign (+))
Floating-point numeric literal
Fractional part of the mantissa
Decimal point
Integer part of the mantissa
Sign (-)
Connect minimum and maximum values using a hyphen (-)
Specification example 3: -r -100-1.0E+2
1 . 0 E + 2
Exponent
Sign (+)
Floating-point numeric literal
Fractional part of the mantissa
Decimal point
Integer part of the mantissa (omit the plus sign (+))

In the case of specification example 1, `-100` and `100` are treated as floating point specifications.

In the case of specification example 2, `-1.0E2` and `+1.0E2` are treated as floating point specifications.

In the case of specification example 3, `-100` and `1.0E+2` are treated as floating point specifications.

The rules for specifying ranges of floating point numbers are as follows:

- The data type in the archive range column for which you can specify a floating point range is as follows:
  - `DOUBLE PRECISION`
- The following table lists the message IDs that are displayed when the specified floating point range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then execute the `adbarchivechunk` command.

Table 2-13: Message IDs that are displayed when the specified floating point range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	KFAA50225-E	The number of specifiable characters is exceeded.	**Specification example** `-r 0-123...(omitted)...789` **Explanation** A floating point specification contains 510 characters.	**Specification example** `-r 0-123...(omitted)...78` **Explanation** Specify a floating point specification using 509 characters.
2	• KFAA90002-E • KFAA96830-I	The range specification format is invalid.	**Specification example** `-r 0Δ-Δ1` **Explanation** Because there is a space between the floating point specification and the hyphen (-), the shell erroneously uses spaces to delimit the option arguments.	`-r 0-1`

Legend:

Δ: Space

2.2.2 Format of archive chunk options

(1) Specification format

```
[set adb_arcv_rthd_num = number-of-processing-real-threads-to-be-used-by-adbarchivech
unk-command]
[set adb_arcv_scan_buff_size = table-data-scan-buffer-size]
```

The rules for specifying these archive chunk options are the same as for the server definition. For the rules for specifying server definitions, see the topic *Syntax rules for the server definition* in the *HADB Setup and Operation Guide*.

(2) Explanation of specification format

- adb_arcv_rthd_num = *number-of-processing-real-threads-to-be-used-by-adbarchivechunk-command*

  ~<integer>((3 to 4,096))

  Specifies the number of processing real threads to be used for executing the adbarchivechunk command.

  Specify 9 for this option. If you do not specify this option, or you specify a value other than 9, chunk archive processing might take longer.

  The adbarchivechunk command uses the number of processing real threads specified in this option. For details about the processing of the adbarchivechunk command using processing real threads and the number of processing real threads to be used for each step of processing, see 2.1.2 Relationships among threads used by the adbarchivechunk command.

  If this option is omitted, the number of processing real threads to be used for executing the adbarchivechunk command is determined by the adb_sql_exe_max_rthd_num operand in the server definition. The following table shows the relationship between the value of this option or the adb_sql_exe_max_rthd_num operand in the server definition and the number of processing real threads used for executing the command.

Table 2-14: Number of processing real threads used for executing the command

No.	Value in this option or in the server definition		Number of processing real threads used for executing the command
1	When this option is omitted	If the adb_sql_exe_max_rthd_num operand in the server definition is not specified	The default value of the adb_sql_exe_max_rthd_num operand in the server definition is used.
2		If a value in the range from 0 to 2 is specified in the adb_sql_exe_max_rthd_num operand in the server definition	--
3		If 3 or a greater value is specified for the adb_sql_exe_max_rthd_num operand in the server definition	The value specified for the adb_sql_exe_max_rthd_num operand in the server definition is used.
4	If 3 or a greater value is specified in this option		The value specified in this option is used.

Legend:

--: The adbarchivechunk command results in an error.

Note:

For details about the adb_sql_exe_max_rthd_num operand in the server definition, see *Operands related to performance (set format)* in the *HADB Setup and Operation Guide*.

- adb_arcv_scan_buff_size = *table-data-scan-buffer-size*

~<integer>((1 to 2,048))<<16>> (units: megabytes)

Specify the size (in megabytes) of the buffer to be used for retaining table data search results. A larger value can reduce overhead in searching table data.

As a guideline, specify a value equivalent to the data length in the search results times about 10,000.

For details about how to obtain data lengths in search results, see *Data length of each data type* in *Determining the memory requirement during normal operation* in *Estimating the HADB server's memory requirement* in *Preparing Resources* in the *HADB Setup and Operation Guide*.

Note that all the scan results stored in one scan buffer are output to the same archive file.

If the specified buffer size is too small to store any data item obtained in the scan results, the adbarchivechunk command will result in an error.

2.3 Examples

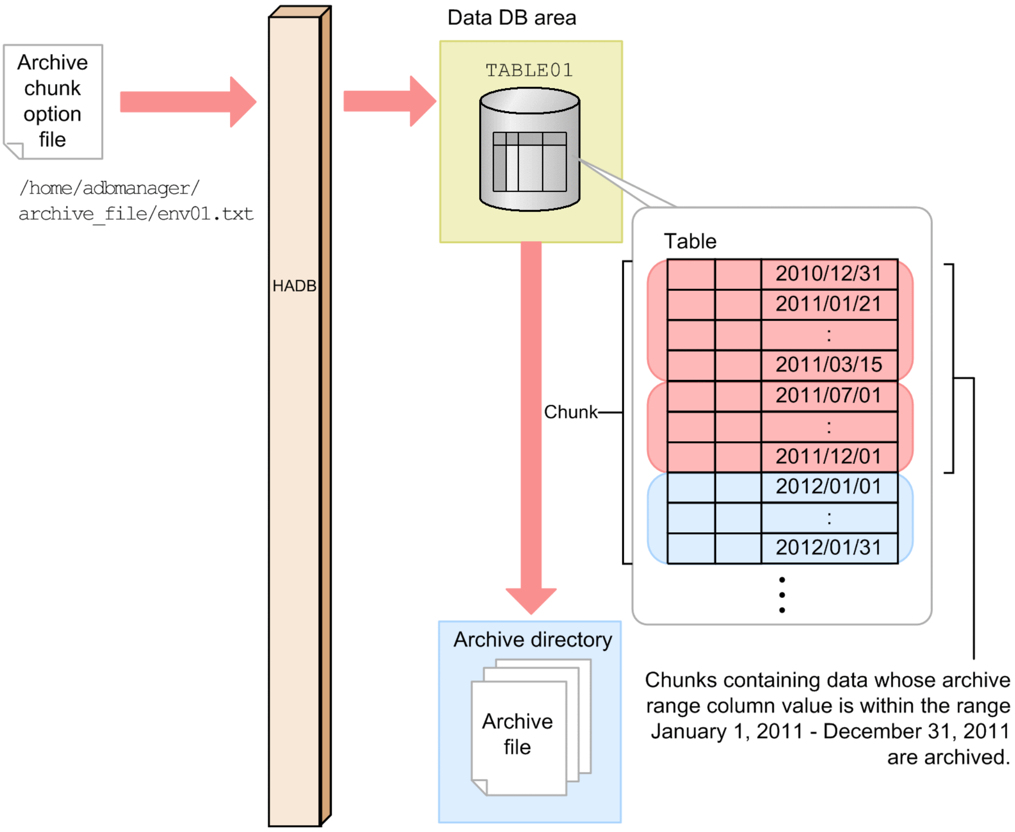
This section explains by way of examples how to use the adbarchivechunk command to archive chunks created in a table.

If the adbarchivechunk command terminates abnormally during execution, take appropriate action based on the explanation in 2.5 Handling abnormal termination of the adbarchivechunk command.

2.3.1 When archiving chunks that include specific date range data

Archive chunks that were created in a table, that include specific date range data, under the following conditions:

- The base table TABLE01 is an archivable multi-chunk table.
- The data type in the archive range column of the base table TABLE01 is DATE.
- Among the chunks created in the base table TABLE01, archive chunks whose values in the archive range column contain data in the range from January 1st, 2011 to December 31st, 2011.


Archive chunk option file
/home/adbmanager/archive_file/env01.txt
HADB
Data DB area
TABLE01
Table
2010/12/31
2011/01/21
:
2011/03/15
2011/07/01
:
2011/12/01
2012/01/01
:
2012/01/31
Chunk
Archive directory
Archive file
Chunks containing data whose archive range column value is within the range January 1, 2011 - December 31, 2011 are archived.

This example uses the following procedure to execute the adbarchivechunk command.

Procedure

1. Create an archive chunk option file.
2. Execute the adbarchivechunk command.

Using examples, the following subsections explain each of these steps.

(1) Creating an archive chunk option file

Create an archive chunk option file, and specify archive chunk options in that file.

Example of an archive chunk option file

```
set adb_arcv_rthd_num = 9          ...1
set adb_arcv_scan_buff_size = 64   ...2
```

Explanation:

1. Specifies the number of processing real threads to be used for executing the `adbarchivechunk` command.
2. Specifies the size of the buffer to be used for retaining table data search results.

(2) Executing the adbarchivechunk command

```
adbarchivechunk -u ADBUSER01                                 ...1
                -p '#HelloHADB_01'                           ...2
                -z /home/adbmanager/archive_file/env01.txt   ...3
                -r 2011/01/01-2011/12/31                     ...4
                TABLE01                                      ...5
```

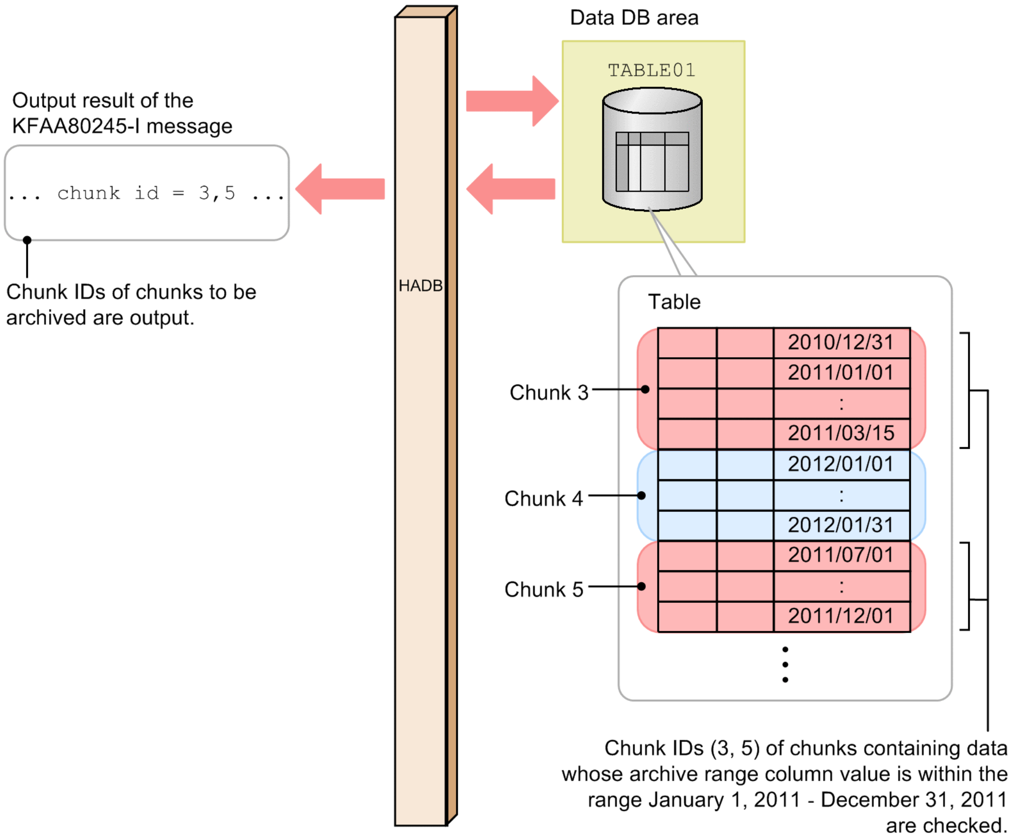
Explanation:

1. Specifies the authorization identifier of the HADB user who executes the `adbarchivechunk` command.
2. Specifies the password for the authorization identifier that was specified in the `-u` option.
3. Specifies the absolute path name of the archive chunk option file created in (1) Creating an archive chunk option file.
4. Specify a date range of chunks to be archived.
5. Specifies the table to be processed.

2.3.2 When you want to check which chunks are to be archived based on a specified date range before actually executing chunk archive processing

Check which chunks are to be archived based on a specified date range before actually executing chunk archive processing, under the following conditions:

- The base table `TABLE01` is an archivable multi-chunk table.
- The data type in the archive range column of the base table `TABLE01` is `DATE`.
- Among the chunks created in the base table `TABLE01`, check the chunk IDs of the chunks subject to archiving whose values in the archive range column contain data in the range from January 1st, 2011 to December 31st, 2011.


Data DB area
TABLE01
Output result of the KFAA80245-I message
... chunk id = 3,5 ...
Chunk IDs of chunks to be archived are output.
HADB
Table
2010/12/31
2011/01/01
:
2011/03/15
2012/01/01
:
2012/01/31
2011/07/01
:
2011/12/01
Chunk 3
Chunk 4
Chunk 5
Chunk IDs (3, 5) of chunks containing data whose archive range column value is within the range January 1, 2011 - December 31, 2011 are checked.

This example uses the following procedure to execute the `adbarchivechunk` command.

Procedure

1. Execute the `adbarchivechunk` command.
2. Check the `KFAA80245-I` message to see which chunks include data in the specified date range.

Using examples, the following subsections explain each of these steps.

(1) Executing the adbarchivechunk command

```
adbarchivechunk -u ADBUSER01                ...1
                -p '#HelloHADB_01'          ...2
                -r 2011/01/01-2011/12/31    ...3
                -t                          ...4
                TABLE01                     ...5
```

Explanation:

1. Specifies the authorization identifier of the HADB user who executes the `adbarchivechunk` command.
2. Specifies the password for the authorization identifier that was specified in the `-u` option.
3. Specify a date range of chunks to be archived.
4. Specify this option if you want to check the chunk IDs of chunks that will actually be archived before executing chunk archive processing.
5. Specifies the table to be processed.

(2) Using the KFAA80245-I message to check which chunks include data in the specified date range

See the `KFAA80245-I` message that is output after the `adbarchivechunk` command is executed with the `-t` option specified.

The `KFAA80245-I` message outputs the chunk IDs of chunks subject to archiving whose values in the archive range column include data in the specified date range (January 1st, 2011 to December 31st, 2011).

To perform chunk archive processing on these chunks, do not change any options other than the `-t` option, and execute the `adbarchivechunk` command with the `-t` option omitted. To revise which chunks are to be archived, correct the specified date range (in the `-r` option), and then execute the `adbarchivechunk` command.

2.4 Return codes

The following table lists and describes the return codes of the `adbarchivechunk` command.

Table 2-15: Return codes of the adbarchivechunk command

Return code	Description
0	The `adbarchivechunk` command terminated normally.
4	The `adbarchivechunk` command terminated normally, but a warning message was issued. Check the displayed message or the message output to a message log file and take appropriate action.
8	One of the following errors occurred: • The specified `adbarchivechunk` command did not execute because it was invalid. • An error occurred during execution of the `adbarchivechunk` command. For the cause of the error and corrective action to take, see the displayed error message or the message that has been output to a message log file.

2.5 Handling abnormal termination of the adbarchivechunk command

If the `adbarchivechunk` command terminates abnormally, check if the following messages have been output:

- `KFAA50225-E`
- `KFAA96606-E`

If either of these messages has been output, there is an error in the specified command options or archive chunk options. Correct the error in the command options or archive chunk options, and then re-execute the `adbarchivechunk` command.

For details about invalid `-c` option specifications, see (3) Errors that can be caused by the -c option in 2.2.1 Explanation of the specification format and options.

Also, for details about invalid `-r` option specifications, see the following resources for each type of range specification.

- Date range specification

  See Table 2-5: Message IDs that are displayed when the specified date range is invalid, and possible errors in (a) Specifying ranges of dates in (4) Explanation of the -r option (range specification) in 2.2.1 Explanation of the specification format and options.

- Time range specification

  See Table 2-7: Message IDs that are displayed when the specified time range is invalid, and possible errors in (b) Specifying ranges of times in (4) Explanation of the -r option (range specification) in 2.2.1 Explanation of the specification format and options.

- Time stamp range specification

  See Table 2-9: Message IDs that are displayed when the specified time stamp range is invalid, and possible errors in (c) Specifying ranges of time stamps in (4) Explanation of the -r option (range specification) in 2.2.1 Explanation of the specification format and options.

- Fixed-length character string range specification

  See Table 2-10: Message IDs that are displayed when the specified fixed-length character string range is invalid, and possible errors in (d) Specifying ranges of fixed-length character strings in (4) Explanation of the -r option (range specification) in 2.2.1 Explanation of the specification format and options.

- Decimal range specification

  See Table 2-11: Message IDs that are displayed when the specified decimal range is invalid, and possible errors in (e) Specifying ranges of decimal numbers in (4) Explanation of the -r option (range specification) in 2.2.1 Explanation of the specification format and options.

- Integer range specification

  See Table 2-12: Message IDs that are displayed when the specified integer range is invalid, and possible errors in (f) Specifying ranges of integers in (4) Explanation of the -r option (range specification) in 2.2.1 Explanation of the specification format and options.

- Floating point range specification

  See Table 2-13: Message IDs that are displayed when the specified floating point range is invalid, and possible errors in (g) Specifying ranges of floating point numbers in (4) Explanation of the -r option (range specification) in 2.2.1 Explanation of the specification format and options.

Note that if you had specified multiple chunks for a `adbarchivechunk` command that terminated abnormally, only chunks that were processed before the abnormality occurred are archived. In this case, after eliminating the cause of the command terminating abnormally, re-execute the `adbarchivechunk` command by using the same specifications as when the command terminated abnormally. Chunk archive processing will be executed for the remaining chunks.

If none of the preceding messages was output, eliminate the cause of the command terminating abnormally, and re-execute the `adbarchivechunk` command.

Note that the `adbarchivechunk` command does not change the contents of tables or indexes. Therefore, there is no need to restore tables or indexes if the `adbarchivechunk` command terminates abnormally.

2.6 Notes

1. The adbarchivechunk command can only be executed when the HADB server is running.
2. Use X'0A' (LF) or X'0D0A' (CRLF) for the line break code in any file specified in the options of the adbarchivechunk command.
3. Use the adbcancel command to cancel execution of the adbarchivechunk command. Alternatively, use a method such as the adbstop --force command or the OS's kill command to forcibly terminate the server process. For details about how to use the adbcancel command to suspend an executing command, see 1.6 When suspending (forcibly terminating) an executing command.
4. Archive chunk processing is performed by server processes. Therefore, using the **Ctrl**+**C** keys to forcibly terminate the adbarchivechunk command process will not interrupt archiving of chunks that is being performed by server processes (such processing continues). When the archive chunk processing is complete, the KFAA81001-I message is output to the message log file.
5. When multiple archive files are created, the number of data items output for each file varies. Consequently, a file containing zero data items might be created. Archive files containing zero data items are deleted.
6. If a chunk containing no data items is archived, an archive file is not created.
7. If the adbarchivechunk command is executed while another command or application program is running, HADB might not be able to allocate the required number of processing real threads. If this happens, HADB waits until the required number of processing real threads can be allocated before executing the adbarchivechunk command. If allocation of processing real threads is placed in wait status, see the topic *Points to consider about the number of processing real threads to be used during command execution* in *Points to consider when executing commands concurrently* in *Preparing Resources* in the *HADB Setup and Operation Guide* and take appropriate action.
8. Specify 9 for the archive chunk option adb_arcv_rthd_num. If you do not specify this option, or you specify a value other than 9, chunk archive processing might take longer.
9. The following chunks cannot be archived:
    - Current chunk
    - Chunks that have never been changed to the normal status after being imported in wait status
    - Deletion-pending chunks
10. If you execute the adbarchivechunk command with the -t option specified, chunk archive processing is not executed. Specify the -t option if you want to check in advance which chunks are to be archived, among those corresponding to the range specified in the -r option or the -c option.

    To perform chunk archive processing on the chunks after checking which chunks are to be archived, do not change any options other than the -t option, and execute the adbarchivechunk command with the -t option omitted.

3 adbaudittrail (Manage the Audit Trail Facility)

This chapter explains the functionality provided by the `adbaudittrail` command and how to use the command.

3.1 Description

The adbaudittrail command is executed when managing the audit trail facility.

For details about the audit trail facility, see *Audit trail facility* in the *HADB Setup and Operation Guide*.

When executed, the adbaudittrail command establishes a connection to the HADB server.

The adbaudittrail command allows you to perform the following four operations:

- Enabling the audit trail facility
  Enable the audit trail facility while the HADB server is running. When the audit trail facility is enabled after environment setup of the audit trail facility has finished, audit trails are output to an audit trail file.
- Disabling the audit trail facility
  Disable the audit trail facility while the HADB server is running. When the audit trail facility is disabled, audit trails are not output to an audit trail file.
- Swapping the audit trail file
  Swap the current audit trail file to which audit trails are output. After the current audit trail file is swapped, you can search for audit trails in the audit trail file used as the current audit trail file before swapping.
- Displaying information related to the audit trail facility
  Display information related to the audit trail facility. You can check whether the audit trail facility is enabled or disabled. You can also check the information in the audit trail file.

3.2 Specification format for the adbaudittrail command

This section explains the specification format and options for the `adbaudittrail` command.

The following OS users can execute the `adbaudittrail` command:

- HADB administrator
- OS users who belong to the HADB administrators group

3.2.1 Explanation of the specification format and options

(1) Specification format

```
adbaudittrail
    -u authorization-identifier
  [-p password]
  {--start [--write-error {DOWN|FAILSOFT}]
  |--stop
  |--swap [-n node-number]
  |-d [-n node-number]}
```

(2) Explanation of options

● `-u` *authorization-identifier*

~<character string>((1 to 100 bytes))

Specifies the authorization identifier of the HADB user who executes the `adbaudittrail` command. The privileges required for executing the `adbaudittrail` command vary depending on the specified options. The following table lists the specified options and the required privileges.

Table 3-1: Privileges required for execution of the adbaudittrail command

No.	Option to specify	Privileges required
1	`--start`	• `CONNECT` privilege • Audit admin privilege
2	`--stop`	• `CONNECT` privilege • Audit admin privilege
3	`--swap`	• `CONNECT` privilege • Audit admin privilege
4	`-d`	• `CONNECT` privilege • Audit admin privilege or audit viewer privilege

For this option, specify an authorization identifier in the range from 1 to 100 bytes. Note that the byte count (1 to 100 bytes) does not include the double quotation marks used to enclose the authorization identifier.

Important

If the character string used as the authorization identifier includes any lowercase letter or a backslash (\\), make sure that you check the rules for specifying authorization identifiers. For the rules for specifying authorization identifiers, see 1.4.2 Rules for specifying authorization identifiers and passwords.

- `-p` *password*

  ~<character string>((1 to 255 bytes))

  Specifies the password for the authorization identifier that is specified in the `-u` option.

  > **Important**
  >
  > If the password includes a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), make sure that you check the rules for specifying passwords. For the rules for specifying passwords, see 1.4.2 Rules for specifying authorization identifiers and passwords.

  If the `adbaudittrail` command is executed with this option omitted, a message prompting the user to enter a password will be displayed. In an environment in which a password cannot be entered from the standard input, such as when the command is executed in the background, make sure that you do not omit this option.

- `--start`

  Specify this option if you want to enable the audit trail facility.

  If the multi-node function is being used, you can execute the `adbaudittrail --start` command on the master node only.

  > **Note**
  >
  > When the audit trail facility is enabled, a new audit trail file is created. Audit trails are output to the created audit trail file.

  - `--write-error {DOWN|FAILSOFT}`

    Specifies the processing method (whether to stop the HADB server) if audit trails can no longer be written to the audit trail file because, for example, the disk is full or a disk failure occurs. When this option is omitted, `DOWN` is assumed.

    `DOWN`:

    If an audit trail cannot be written to the audit trail file, the HADB server stops. At this time, the termination mode of the HADB server is abnormal termination.

    `FAILSOFT`:

    Even if an audit trail cannot be written to the audit trail file, the HADB server does not stop. The audit trail that cannot be written to the audit trail file is discarded, and the HADB server continues operation.

    When the cause of the audit-trail write error to the audit trail file is eliminated, output to the audit trail file automatically restarts from the next write timing of audit trails.

    > **Important**
    >
    > The specification of the `--write-error` option can be changed only when the `adbaudittrail --start` command is executed to enable the audit trail facility.
    >
    > Therefore, if you want to change the specification of the `--write-error` option when the audit trail facility is enabled, first execute the `adbaudittrail --stop` command to disable the audit trail facility. Then, change the specification of the `--write-error` option, and then re-execute the `adbaudittrail --start` command to enable the audit trail facility.

- `--stop`

  Specify this option if you want to disable the audit trail facility. When the audit trail facility is disabled, output of audit trails stops.

  If the multi-node function is being used, you can execute the `adbaudittrail --stop` command on the master node only.

  > **Note**
  >
  > When the audit trail facility is disabled, the current audit trail file is renamed.
  >
  > For details about renaming of the audit trail file, see *Output destination of audit trails (audit trail file)* in the *HADB Setup and Operation Guide*.

- `--swap`

  Specify this option to swap the current audit trail file to which audit trails are output.

  You cannot search for or convert audit trails in the current audit trail file. If you want to search for or convert an audit trail in the current audit trail file, execute the `adbaudittrail --swap` command to swap the current audit trail file.

  If the multi-node function is being used, the `adbaudittrail --swap` command with the `-n` option omitted can be executed on either the master node or slave node. In this situation, the audit trail file to be output on the node on which the command is executed is swapped.

  - `-n` *node-number*

    ~<integer> ((1 to 4))

    Swaps the audit trail file to be output on the node whose number is specified for this option.

    You can specify this option when the following two conditions are met.

    - If the multi-node function is used
    - If the `adbaudittrail` command is executed on the master node

    When the multi-node function is not being used, this option is ignored.

  > **Note**
  >
  > When the `adbaudittrail --swap` command is executed, the current audit trail file before swapping is renamed. Then, a new current audit trail file is created. After the `adbaudittrail --swap` command is executed, you can view audit trails in the renamed audit trail file.
  >
  > For details about renaming of the audit trail file, see *Output destination of audit trails (audit trail file)* in the *HADB Setup and Operation Guide*.

- `-d`

  Specify this option if you want to display information related to the audit trail facility.

  You can check whether the audit trail facility is enabled or disabled. You can also check the information in the audit trail file. For details about the output format and the displayed information, see 3.3.2 Displaying information related to the audit trail facility.

  If the multi-node function is being used, the `adbaudittrail -d` command with the `-n` option omitted can be executed on either the master node or slave node. In this situation, information related to the audit trail facility of the node on which the command is executed is displayed.

  - `-n` *node-number*

    ~<integer> ((1 to 4))

Information related to the audit trail facility on the node with the node number specified in this option will be displayed.

You can specify this option when the following two conditions are met.

- If the multi-node function is used
- If the `adbaudittrail` command is executed on the master node

When the multi-node function is not being used, this option is ignored.

3.3 Examples

This section uses examples to explain how to accomplish the following tasks by using the `adbaudittrail` command:

- Enable the audit trail facility
- Display information related to the audit trail facility

3.3.1 Enabling the audit trail facility

This example enables the audit trail facility.

```
adbaudittrail -u ADBAUDITADMIN
              -p '#HelloHADB_ADMIN'
              --start --write-error DOWN
```

When an HADB user who has the audit admin privilege executes the `adbaudittrail --start` command, the audit trail facility is enabled.

In this example, `DOWN` is specified for the `--write-error` option in order to abnormally terminate the HADB server if an audit trail can no longer be written to the audit trail file.

3.3.2 Displaying information related to the audit trail facility

This example displays information related to the audit trail facility.

```
adbaudittrail -u ADBAUDITVIEWER
              -p '#HelloHADB_VIEWER'
              -d
```

When the `adbaudittrail -d` command is executed, you can check the information related to the audit trail facility.

Output example

```
audit   write-error audit-directory-path      audit-file-max-size audit-file-numbe
r
ACTIVE  DOWN        /home/ADBAUDITADMIN/audit 256                 100
```

Explanation of the output items

`audit`

Displays the status information for the audit trail facility. One of the following values is displayed:

- `ACTIVE`

  The audit trail facility is enabled.
- `ACTIVE(NO OUTPUT)`

  The audit trail facility is enabled. However, output of audit trails to the audit trail file is suppressed.
- `INACTIVE`

  The audit trail facility is disabled.

`write-error`

Displays the processing method if audit trails can no longer be written to the audit trail file because, for example, the disk is full or a disk failure occurs. The value specified for the `--write-error` option is displayed.

- `DOWN`

  The HADB server is set to terminate abnormally if an attempt to write to the audit trail file fails.

- `FAILSOFT`

  The HADB server is set to continue operation if an attempt to write to the audit trail file fails.

`audit-directory-path`

Displays the audit trail directory. The directory name specified for the `adb_audit_log_path` operand in the server definition is displayed.

`audit-file-max-size`

Displays the maximum size (in megabytes) of an audit trail file. The value of the `adb_audit_log_max_size` operand in the server definition is displayed.

`audit-file-number`

Displays the maximum number of audit trail file generations. The value of the `adb_audit_log_max_num` operand in the server definition is displayed.

If you specify `0` for the `adb_audit_log_max_num` operand or if you omit the `adb_audit_log_max_num` operand, `0` is displayed. `0` indicates that the maximum number of audit trail file generations is not limited.

3.4 Return codes

The following table lists and describes the return codes of the `adbaudittrail` command.

Table 3-2: Return codes of the adbaudittrail command

Return code	Description
`0`	The `adbaudittrail` command terminated normally.
`4`	The `adbaudittrail` command terminated normally, but a warning message or an error message was output. Check the displayed message or the message output to a message log file and take appropriate action.
`8`	An error occurred during execution of the `adbaudittrail` command. For the cause of the error and corrective action to take, see the displayed error message or the message output to a message log file.

3.5 Notes

1. To enable the audit trail facility, all the following conditions must be met. If the conditions are not met, the `adbaudittrail` command results in an error.
   - An existing directory is specified for the `adb_audit_log_path` operand in the server definition.
   - Read, write, and execution privileges are set so that the HADB administrator can access the directory specified for the `adb_audit_log_path` operand in the server definition.
   - The number of audit trail files stored under the directory specified for the `adb_audit_log_path` operand in the server definition is equal to or smaller than the value specified for the `adb_audit_log_max_num` operand in the server definition.
2. If you enable the audit trail facility, the output of audit trail starts from the SQL statement and command that started after normal termination of the `adbaudittrail` command. No audit trails are output for SQL statements and commands that were being executed when the `adbaudittrail` command terminated normally.
3. If you disable the audit trail facility, no audit trails are output for SQL statements and commands that were being executed when the `adbaudittrail` command terminated normally. Also note that audit trails will not be output for SQL statements and commands that are executed subsequently.
4. Only one `adbaudittrail` command can be executed at a time. If you execute multiple `adbaudittrail` commands at the same time, an error occurs.
5. If the multi-node function is being used, the node on which the `adbaudittrail` command can be executed differs depending on the specified option. The following table shows the correspondences between the options to specify and the nodes on which the `adbaudittrail` command can be executed.

   Table 3-3: Correspondences between the options to specify and the nodes on which the adbaudittrail command can be executed

No.	Option to specify	Node on which the command can be executed
1	`--start`	Master node
2	`--stop`	Master node
3	`--swap`#1	Master node or slave node
4	`--swap -n` *node-number*	Master node
5	`-d`#2	Master node or slave node
6	`-d -n` *node-number*	Master node

   #1

   On the node on which the command is executed, the audit trail file that contains the audit trails to be output is swapped.

   #2

   Information related to the audit trail facility of the node on which the command is executed is output.

   When the audit trail facility is being enabled, the `adbaudittrail` command results in an error if the audit trail facility cannot be enabled on at least one node. This also applies when the audit trail facility is being disabled.
6. To use the multi-node function, the node number `-n` must meet all the following conditions. If the conditions are not met, the `adbaudittrail` command results in an error.
   - The node number is equal to or smaller than the number of nodes containing HADB servers in a multi-node configuration.
   - The HADB server with the specified node number is running.

7. Once enabled, the audit trail facility remains in effect until it is disabled by using the `adbaudittrail` command. If the HADB server is terminated with the audit trail facility enabled, the audit trail facility will remain in effect the next time the HADB server is started. The specification of the `--write-error` option (that is, the processing method if audit trails can no longer written to the audit trail file) is also inherited.

Note

For the following server definitions related to the audit trail facility, the specified values existing when the HADB server is started are applied (the specified values that existed when the HADB server previously terminated are not inherited):

- `adb_audit_log_path` operand (audit trail directory)
- `adb_audit_log_max_size` operand (maximum size of an audit trail file)
- `adb_audit_log_max_num` operand (maximum number of audit trail file generations)

4 adbcancel (Forcibly Terminate Transactions or Commands)

This chapter explains the functionality provided by the `adbcancel` command and how to use the command.

4.1 Description

The adbcancel command forcibly terminates HADB server processing that is executing in response to a processing request issued by an application program or command. The adbcancel command then disconnects the application program or command from the HADB server.

You use the adbcancel command to forcibly terminate HADB server processing when an application program or command has terminated abnormally but the HADB server processing has not finished.

If you execute the adbcancel command while an application program is executing a transaction, the command rolls back the transaction and then disconnects the application program from the HADB server.

When you are using the multi-node function, you can execute the adbcancel command on all nodes.

You can also use the adbcancel command to suspend (forcibly terminate) an executing command. For details about how to suspend (forcibly terminate) a command, see 1.6 When suspending (forcibly terminating) an executing command.

4.2 Specification format for the adbcancel command

This section explains the specification format and options for the `adbcancel` command.

The only OS users who can execute the `adbcancel` command are HADB administrators.

4.2.1 Explanation of the specification format and options

(1) Specification format

```
adbcancel {--ALL | -u connection-ID}
```

(2) Explanation of options

- `--ALL`

  Specifies that all HADB server processes connected to the HADB server are to be terminated forcibly and then disconnected from the HADB server.

  When you execute the `adbcancel` command with `--ALL` specified, the command issues the `KFAA91251-Q` message. You must reply to this message to confirm that the `adbcancel` command is to be executed. To execute the `adbcancel` command, enter `Y`. If you enter any character other than `Y`, execution of the `adbcancel` command is canceled.

  If the multi-node function is used, all connections to all HADB servers in the multi-node configuration will be affected by the `adbcancel` command.

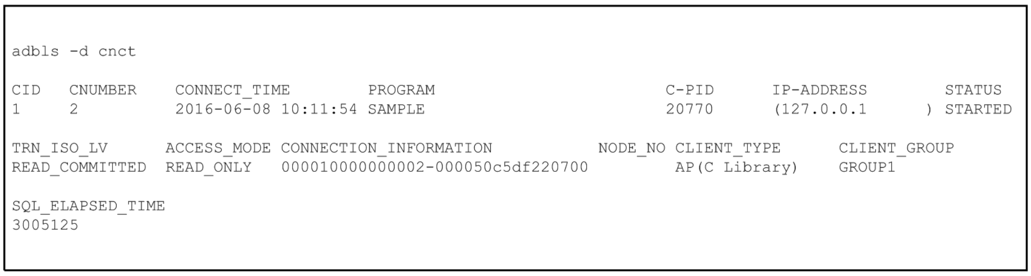
- `-u` *connection-ID*

  ~<integer>((1 to 1,024))

  Specifies the connection ID of the connection that is to be the target of forced termination.

  This is the connection ID shown in the `CID` column of the execution results of the `adbls -d cnct` command. For details about the `adbls -d cnct` command, see 23. adbls -d cnct (Display the Connection Status).

  ■ Example of execution results of the `adbls -d cnct` command



```
adbls -d cnct

CID   CNUMBER    CONNECT_TIME        PROGRAM                         C-PID      IP-ADDRESS        STATUS
1     2          2016-06-08 10:11:54 SAMPLE                          20770      (127.0.0.1      ) STARTED

TRN_ISO_LV      ACCESS_MODE CONNECTION_INFORMATION            NODE_NO CLIENT_TYPE      CLIENT_GROUP
READ_COMMITTED  READ_ONLY   000010000000002-000050c5df220700          AP(C Library)    GROUP1

SQL_ELAPSED_TIME
3005125
```

4.3 Example

This example forcibly terminates a transaction of the application program whose connection ID is 10.

```
adbcancel -u 10

KFAA90000-I adbcancel processing started.
KFAA90001-I adbcancel processing ended. (return code = 0)
```

4.4 Return codes

The following table lists and describes the return codes of the adbcancel command.

Table 4-1: Return codes of the adbcancel command

Return code	Description
0	The adbcancel command terminated normally.
4	The adbcancel command issued a warning message, and then terminated itself. An asynchronous cancellation request to the HADB server has been completed. Check the displayed message or the message output to a message log file and take appropriate action.
8	An error occurred during execution of the adbcancel command. Check the displayed message or the message output to a message log file and take appropriate action.

4.5 Notes

1. Because the `adbcancel` command is executed asynchronously with the HADB server's processing, use the `adbls -d cnct` command to verify that the target of the processing has been disconnected from HADB. If no target connection information is displayed for the connection ID specified in the `adbcancel` command, the processing target has been disconnected from HADB.
2. The `adbcancel` command is used only to forcibly terminate a command (the `adbimport` command, for example) that has established a connection with the HADB server. To determine whether a connection has been established, check the `PROGRAM` column of the execution results of the `adbls -d cnct` command. If a connection has been established, the command name is displayed in the `PROGRAM` column.
3. If you use the `adbcancel` command to forcibly terminate an executing application program, the command rolls back the transaction and then disconnects the application program from the HADB server. To determine whether the target application program was executing a transaction when the `adbcancel` command was issued, check if `STARTED` is displayed in the `STATUS` column of the `adbls -d cnct` command's execution results. A transaction was executing if `STARTED` is displayed.
4. In the case where an application program has terminated abnormally but its transaction is still running, the transaction will be rolled back and the application program disconnected from the HADB server when the time specified in the `adb_clt_rpc_sql_wait_time` operand in the corresponding application program's client definition is reached, without having to execute the `adbcancel` command.
5. If you are using the multi-node function, the `adbcancel` command is asynchronously executed between nodes. Therefore, information displayed by the `adbls` command might be temporarily inconsistent between nodes.

5 adbchgchunkcomment (Set, Change, and Delete Comments for Chunks)

This chapter explains the functionality provided by the `adbchgchunkcomment` command and how to use the command.

Note that the term *tables* in this chapter means base tables.

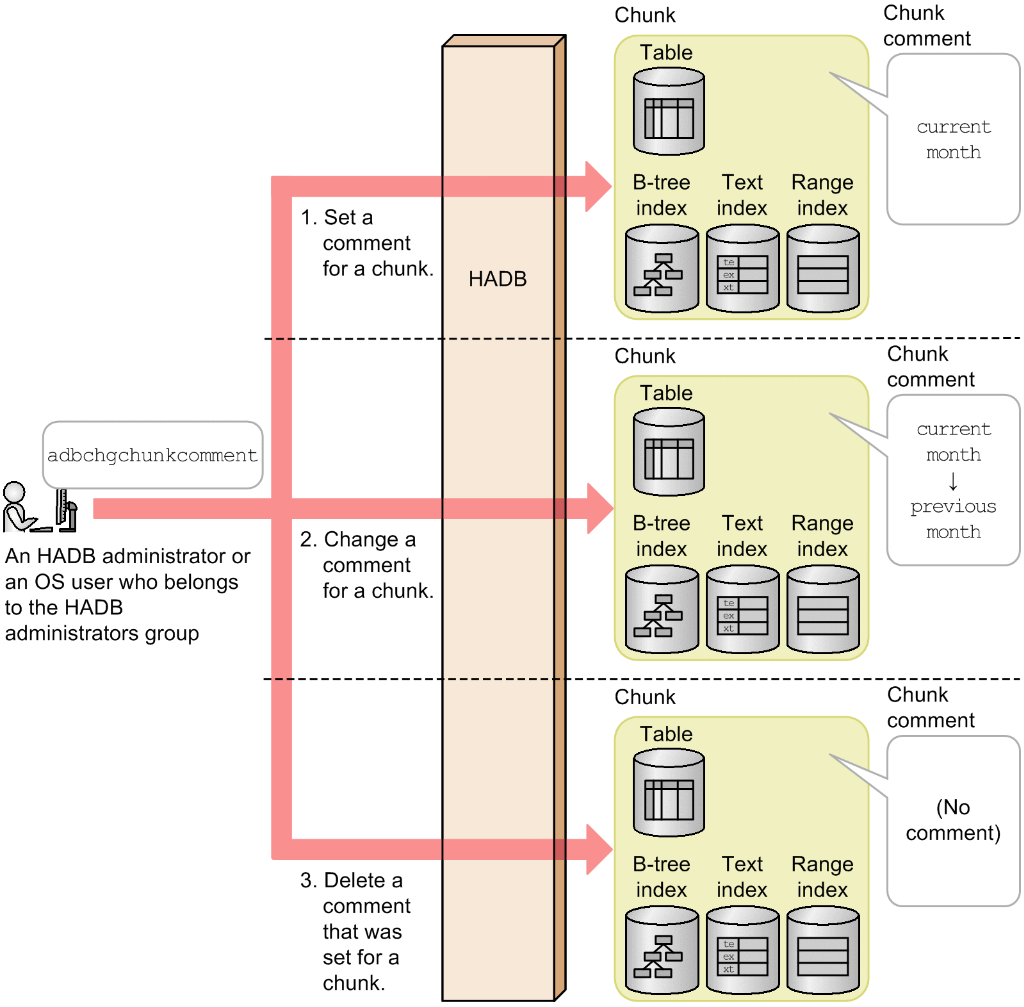
5.1 Description

The adbchgchunkcomment command sets, changes, and deletes comments for chunks in tables for which CHUNK was specified (multi-chunk tables) in the table definition.

When executed, the adbchgchunkcomment command establishes a connection to the HADB server.

The following figure provides an overview of the processing that sets, changes, and deletes comments for chunks.

Figure 5-1: Setting, changing, and deleting comments for chunks



adbchgchunkcomment
An HADB administrator or an OS user who belongs to the HADB administrators group
HADB
1. Set a comment for a chunk.
2. Change a comment for a chunk.
3. Delete a comment that was set for a chunk.
Chunk
Table
B-tree index
Text index
Range index
Chunk comment
current month
current month
↓
previous month
(No comment)

Explanation:

1. Sets a comment for a chunk.
   Example: A new comment (current month) is set for a chunk.

2. Changes a comment that has been set for a chunk.
   Example: An existing comment (current month) for a chunk is changed to a new comment (previous month).

3. Deletes a comment that has been set for a chunk.
   Example: A comment that has been set for a chunk is deleted.

5.2 Specification format for the adbchgchunkcomment command

This section explains the specification format and options for the `adbchgchunkcomment` command.

The following OS users can execute the `adbchgchunkcomment` command:

- HADB administrator
- OS users who belong to the HADB administrators group

5.2.1 Explanation of the specification format and options

(1) Specification format

```
adbchgchunkcomment
   -u authorization-identifier
  [-p password]
  {-m chunk-comment-to-be-set|-d}
   -c chunk-ID
   table-to-be-processed
```

Important

You must specify the table to be processed as the last option that is specified.

(2) Explanation of options

- `-u` *authorization-identifier*

  ~<character string>((1 to 100 bytes))

  Specifies the authorization identifier of the HADB user who executes the `adbchgchunkcomment` command. To execute the `adbchgchunkcomment` command, the user must have the following two privileges:

  - `CONNECT` privilege
  - `CHANGE CHUNK COMMENT` privilege for the table to be processed

  For this option, specify an authorization identifier in the range from 1 to 100 bytes. Note that the byte count (1 to 100 bytes) does not include the double quotation marks used to enclose the authorization identifier.

  Important

  If the character string used as the authorization identifier includes any lowercase letter or a backslash (`\`), make sure that you check the rules for specifying authorization identifiers. For the rules for specifying authorization identifiers, see 1.4.2 Rules for specifying authorization identifiers and passwords.

- `-p` *password*

  ~<character string>((1 to 255 bytes))

  Specifies the password for the authorization identifier that is specified in the `-u` option.

Important

If the password includes a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), make sure that you check the rules for specifying passwords. For the rules for specifying passwords, see 1.4.2 Rules for specifying authorization identifiers and passwords.

If the `adbchgchunkcomment` command is executed with this option omitted, a message prompting the user to enter a password is displayed. In an environment in which a password cannot be entered from the standard input, such as when the command is executed in the background, make sure that you do not omit this option.

- `-m` *chunk-comment-to-be-set*

  ~<character string>((1 to 1,024 bytes))

  Specifies the text that is to be set as the new or changed comment for a chunk.

  Specify this option for a multi-chunk table. An error results if this option is specified for a single-chunk table.

  The following rules apply to specifying this information:

  - The specified comment is case-sensitive.
  - If the comment contains a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or a vertical bar (`|`), specify the escape character (`\`) immediately before that character. Alternatively, enclose the entire comment between single quotation marks (`'`).

    Example 1: `-m abc\|def`

    Example 2: `-m 'abc|def'`

    In both examples, `abc|def` is set as the comment.
  - If a character string is enclosed between double quotation marks (`"`) and then the entire string is enclosed between single quotation marks (`'`), both the character string and the double quotation marks are treated as the comment.

    Example: `-m '"abcdef"'`

    In this example, `"abcdef"` is set as the comment.
  - The character encoding specified in the HADB server's `ADBLANG` environment variable is used as the character encoding for the comment.

- `-d`

  Specifies that the comment set for the chunk is to be deleted. If the `adbchgchunkcomment` command with the `-d` option specified is executed for a chunk for which no comment is set, the command terminates normally without deleting any comment.

- `-c` *chunk-ID*

  ~<integer>((1 to 9,223,372,036,854,775,807))

  Specifies the chunk ID of the chunk for which a comment is to be set, changed, or deleted.

  If a nonexistent chunk ID is specified, the `KFAA50294-E` message is displayed.

- *table-to-be-processed*

  Specifies the table for which a chunk comment is to be set, changed, or deleted. Specify a multi-chunk table.

  The following rules apply to specifying this information:

  - Specify in the format *schema-name*`.`*table-identifier*.

    If the user owns this table, the schema name can be omitted. If another HADB user owns this table, the schema name must be specified.
  - For details about the specification rules that apply when a schema name or table identifier includes any lowercase letter or backslash (`\`), see 1.4.3 Table name specification rules.

- The user can specify any table for which the user has the `CHANGE CHUNK COMMENT` privilege.
- A single-chunk table cannot be specified.
- A viewed table cannot be specified.

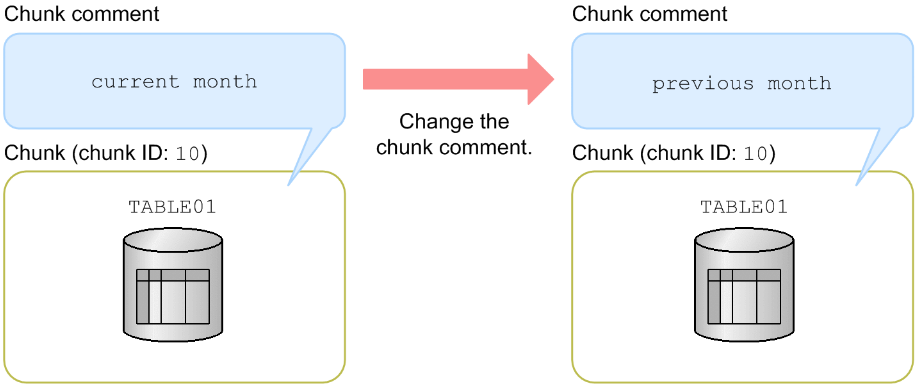
5.3 Examples

This section explains by way of examples how to use the adbchgchunkcomment command to change and delete comments that have been set for chunks.

5.3.1 Changing comments set for chunks

This example changes a comment set for a chunk to a new comment (previous month) under the following conditions:

- CHUNK was specified when table TABLE01 was defined (table TABLE01 is a multi-chunk table).
- Table TABLE01 has a chunk whose chunk ID is 10.
- A comment (current month) has been set for the chunk whose chunk ID is 10.



■ adbchgchunkcomment command that is executed

```
adbchgchunkcomment -u ADBUSER01        ...1
                   -p '#HelloHADB_01'  ...2
                   -m 'previous month' ...3
                   -c 10               ...4
                   TABLE01             ...5
```

Explanation:

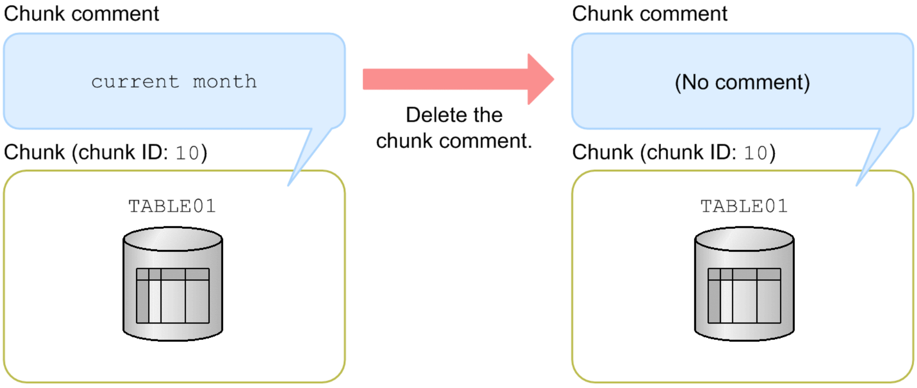
1. Specifies the authorization identifier of the HADB user who executes the adbchgchunkcomment command.
2. Specifies the password for the authorization identifier that is specified in the -u option.
3. Specifies a new comment (previous month).
4. Specifies the chunk ID (10) of the chunk whose comment is to be changed.
5. Specifies the table to be processed (TABLE01).

5.3.2 Deleting comments set for chunks

This example deletes a comment set for a chunk under the following conditions:

- CHUNK was specified when table TABLE01 was defined (table TABLE01 is a multi-chunk table).

- Table TABLE01 has a chunk whose chunk ID is 10.
- A comment (current month) has been set for the chunk whose chunk ID is 10.



■ adbchgchunkcomment command that is executed

```
adbchgchunkcomment -u ADBUSER01         ...1
                   -p '#HelloHADB_01'   ...2
                   -d                   ...3
                   -c 10                ...4
                   TABLE01              ...5
```

Explanation:

1. Specifies the authorization identifier of the HADB user who executes the adbchgchunkcomment command.
2. Specifies the password for the authorization identifier that is specified in the -u option.
3. Specifies that a comment is to be deleted.
4. Specifies the chunk ID (10) of the chunk whose comment is to be deleted.
5. Specifies the table to be processed (TABLE01).

5.4 Return codes

The following table lists and describes the return codes of the `adbchgchunkcomment` command.

Table 5-1: Return codes of the adbchgchunkcomment command

Return code	Description
`0`	The `adbchgchunkcomment` command terminated normally.
`4`	The `adbchgchunkcomment` command terminated normally, but a warning message was output. Check the displayed message or the message log file and take appropriate action.
`8`	An error occurred during execution of the `adbchgchunkcomment` command. For the cause of the error, see the displayed message or the message that has been output to a message log file.

5.5 Handling abnormal termination of the adbchgchunkcomment command

If the `adbchgchunkcomment` command terminates abnormally, check if either of the following messages has been output:

- `KFAA50225-E`
- `KFAA96606-E`

If either of these messages has been output, errors were detected in the specified command options. Correct all errors in options and re-execute the `adbchgchunkcomment` command.

If neither of these messages was output, eliminate the cause of the abnormal termination and re-execute the `adbchgchunkcomment` command.

Note that because the `adbchgchunkcomment` command does not change the contents of tables or indexes, there is no need to restore tables and indexes after the `adbchgchunkcomment` command has terminated abnormally.

5.6 Notes

1. Use the `adbcancel` command to cancel execution of the `adbchgchunkcomment` command. Alternatively, use a method such as the `adbstop --force` command or the OS's `kill` command to forcibly terminate the server process. For details about how to use the `adbcancel` command to suspend an executing command, see 1.6 When suspending (forcibly terminating) an executing command.
2. The processing to set, change, or delete a chunk comment is performed by a server process. Therefore, using the **Ctrl**+**C** keys to forcibly terminate the `adbchgchunkcomment` command process will not interrupt the chunk comment setting, change, or deletion processing that is being performed by server processes (the processing continues). When the processing is completed, the command will output the `KFAA81001-I` message to the message log file.
3. You cannot execute the `adbchgchunkcomment` command on the following tables:
    - Non-updatable tables
    - Tables for which execution of the `adbimport` or `adbidxrebuild` command has been halted

6 adbchgchunkstatus (Change Chunk Status)

This chapter explains the functionality provided by the `adbchgchunkstatus` command and how to use the command.

Note that the term *tables* in this chapter means base tables.

6.1 Description

The adbchgchunkstatus command changes chunk status from normal to wait or from wait to normal.

When executed, the adbchgchunkstatus command establishes a connection to the HADB server.

6.2 Specification format for the adbchgchunkstatus command

This section explains the specification format and options for the `adbchgchunkstatus` command.

The following OS users can execute the `adbchgchunkstatus` command:

- HADB administrator
- OS users who belong to the HADB administrators group

6.2.1 Explanation of the specification format and options

(1) Specification format

```
adbchgchunkstatus
   -u authorization-identifier
  [-p password]
  [-w chunk-ID-of-chunk-to-be-changed-to-wait-status[,chunk-ID-of-chunk-to-be-changed
-to-wait-status]...]
  [-n chunk-ID-of-chunk-to-be-changed-to-normal-status[,chunk-ID-of-chunk-to-be-chang
ed-to-normal-status]...]
   table-to-be-processed
```

Important

- Make sure that you specify at least one of the `-w` and `-n` options. You can also specify both options.
- You must specify the table to be processed as the last option that is specified.

(2) Explanation of options

- `-u` *authorization-identifier*

  ~<character string>((1 to 100 bytes))

  Specifies the authorization identifier of the HADB user who executes the `adbchgchunkstatus` command. To execute the `adbchgchunkstatus` command, the user must have the following two privileges:

  - `CONNECT` privilege
  - `CHANGE CHUNK STATUS` privilege for the table to be processed

  For this option, specify an authorization identifier in the range from 1 to 100 bytes. Note that the byte count (1 to 100 bytes) does not include the double quotation marks used to enclose the authorization identifier.

  Important

  If the character string used as the authorization identifier includes any lowercase letter or a backslash (`\`), make sure that you check the rules for specifying authorization identifiers. For the rules for specifying authorization identifiers, see 1.4.2 Rules for specifying authorization identifiers and passwords.

- `-p` *password*

  ~<character string>((1 to 255 bytes))

Specifies the password for the authorization identifier that is specified in the `-u` option.

> **Important**
>
> If the password includes a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), make sure that you check the rules for specifying passwords. For the rules for specifying passwords, see 1.4.2 Rules for specifying authorization identifiers and passwords.

If the `adbchgchunkstatus` command is executed with this option omitted, a message prompting the user to enter a password is displayed. In an environment in which a password cannot be entered from the standard input, such as when the command is executed in the background, make sure that you do not omit this option.

- `-w` *chunk-ID-of-chunk-to-be-changed-to-wait-status* `[,`*chunk-ID-of-chunk-to-be-changed-to-wait-status*`]...`

  ~<integer>((1 to 9,223,372,036,854,775,807))

  Specifies the chunk IDs of the chunks whose status is to be changed to wait status. Separate multiple chunk IDs with the comma (`,`), or use the hyphen (`-`) to separate a range of chunk IDs.

  - Example 1: Placing chunks with chunk IDs 1, 3, and 5 in wait status:

    `-w 1,3,5`

  - Example 2: Placing chunks with chunk IDs 1 through 5 in wait status:

    `-w 1-5`

  - Example 3: Placing chunks with chunk IDs 1, 3, 5 through 10, and 15 in wait status:

    `-w 1,3,5-10,15`

  If a specified chunk ID does not exist in the table to be processed or belongs to a chunk that is not in normal status, that chunk ID is ignored (only chunks whose status can be changed are processed). If there is no chunk whose status can be changed, the command results in an error.

- `-n` *chunk-ID-of-chunk-to-be-changed-to-normal-status* `[,`*chunk-ID-of-chunk-to-be-changed-to-normal-status*`]...`

  ~<integer>((1 to 9,223,372,036,854,775,807))

  Specifies the chunk IDs of the chunks whose status is to be changed to normal status. Separate multiple chunk IDs with the comma (`,`) or use the hyphen (`-`) to separate a range of chunk IDs.

  - Example 1: Placing chunks with chunk IDs 1, 3, and 5 in normal status:

    `-n 1,3,5`

  - Example 2: Placing chunks with chunk IDs 1 through 5 in normal status:

    `-n 1-5`

  - Example 3: Placing chunks with chunk IDs 1, 3, 5 through 10, and 15 in normal status:

    `-n 1,3,5-10,15`

  If a specified chunk ID does not exist in the table to be processed or belongs to a chunk that is not in wait status, that chunk ID is ignored (only chunks whose status can be changed are processed). If there is no chunk whose status can be changed, the command results in an error.

- *table-to-be-processed*

  Specifies the name of the table for which chunk status is to be changed. Specify a multi-chunk table.

  The following rules apply to specifying this information:

  - Specify in the format *schema-name*`.`*table-identifier*.

    If the user owns this table, the schema name can be omitted. If another HADB user owns this table, the schema name must be specified.

- For details about the specification rules that apply when a schema name or table identifier includes any lowercase letter or backslash (\), see 1.4.3 Table name specification rules.
- The user can specify any table for which the user has the `CHANGE CHUNK STATUS` privilege.
- A single-chunk table cannot be specified.
- A viewed table cannot be specified.

(3) Rules for specifying the -w and -n options

- When you specify multiple chunk IDs, do not specify any spaces before or after a comma or a hyphen.
- When a hyphen is used to specify a range of chunk IDs, values cannot be specified in descending order (such as `3-1`). Also, a specification such as `3-3` is not allowed.
- A maximum of 29,999 chunk IDs can be specified.

  Furthermore, if you specify a range of chunk IDs by using a hyphen, all chunk IDs in the range are counted. Note that every chunk ID is counted, including duplicates.

  Examples:

  - When `-w 1,3,5` is specified

    The command assumes that three chunk IDs are specified.
  - When `-w 1-10` is specified

    The command assumes that ten chunk IDs are specified.
  - When `-n 1,3-5,5` is specified

    The command assumes that five chunk IDs are specified.
- The permitted maximum length of the value that can be specified for the `-w` or `-n` option is 32 kilobytes. Therefore, if the length of the value specified for the `-w` or `-n` option exceeds 32 kilobytes, use range specifications or execute the `adbchgchunkstatus` command more than once.
- The following table shows examples of invalid `-w` and `-n` option specifications.

Table 6-1: Examples of invalid -w and -n option specifications

Error message displayed when the adbchgchunkstatus command is executed	Cause of the error

`KFAA50225-E`	The format of a range specification using a hyphen is invalid.
	• Example of invalid specifications
	`-w 3-3,4`
	`-n 10-3`
	• Example of corrected specifications
	`-w 3,4`
	`-n 3-10`

	More than 29,999 chunk IDs are specified:
	• Example of invalid specification
	`-w 1-30000`
	• Example of corrected specification
	`-w 1-29999`

`KFAA50235-E`	• Neither the `-w` option nor the `-n` option is specified. Specify either the `-w` or the `-n` option or specify both options. • There are no chunks whose status can be changed. Check if chunks with the specified chunk IDs exist. Or, check the statuses of the specified chunks.

Error message displayed when the adbchgchunkstatus command is executed	Cause of the error
`KFAA50294-E`	A specified chunk ID is outside the range of chunk IDs in the table to be processed. Check the chunk IDs in the table to be processed.

• `KFAA90002-E`
• `KFAA96820-I`
	A space is specified before or after a comma or a hyphen.
	• Example of invalid specifications
	`-w 1,Δ3`
	`-n 1Δ-10`
	• Example of corrected specifications
	`-w 1,3`
	`-n 1-10`

`KFAA90003-E`	The length of the `-w` or `-n` option value exceeds 32 kilobytes. Use range specifications for the `-w` or `-n` option to shorten the length of the option value.

Legend:

Δ: Space

6.3 Examples

This section explains by way of examples how to use the `adbchgchunkstatus` command to change chunk status.

6.3.1 Changing the chunk status from normal to wait

This example changes the following chunks for table `TABLE01` from normal status to wait status:

- Chunks with chunk IDs 1, 3 through 5, and 8

(1) Checking the chunk status

Before you change the chunk status, use one of the following methods to check the status of the target chunks:

- Execute the `adbdbstatus` command.
- Retrieve information from the `CHUNK_STATUS` column in the `STATUS_CHUNKS` system table.

(2) Executing the adbchgchunkstatus command

```
adbchgchunkstatus -u ADBUSER01              ...1
                  -p '#HelloHADB_01'        ...2
                  -w 1,3-5,8                ...3
                  TABLE01                   ...4
```

Explanation:

1. Specifies the authorization identifier of the HADB user who executes the `adbchgchunkstatus` command.
2. Specifies the password for the authorization identifier that is specified in the `-u` option.
3. Specifies the chunk IDs of the chunks whose status is to be changed from normal status to wait status.
4. Specifies the table to be processed (`TABLE01`).

6.3.2 Changing the status of chunks in normal and wait statuses

This example changes the status of the following chunks for table `TABLE01` as follows:

- Changing the chunks with chunk IDs 1 through 3 to wait status
- Changing the chunks with chunk IDs 4 through 6 to normal status

(1) Checking the chunk status

Before you change the chunk status, use one of the following methods to check the status of the target chunks:

- Execute the `adbdbstatus` command.
- Retrieve information from the `CHUNK_STATUS` column in the `STATUS_CHUNKS` system table.

(2) Executing the adbchgchunkstatus command

```
adbchgchunkstatus -u ADBUSER01            ...1
                  -p '#HelloHADB_01'      ...2
                  -w 1-3                  ...3
                  -n 4-6                  ...4
                  TABLE01                 ...5
```

Explanation:

1. Specifies the authorization identifier of the HADB user who executes the `adbchgchunkstatus` command.
2. Specifies the password for the authorization identifier that is specified in the `-u` option.
3. Specifies the chunk IDs of the chunks whose status is to be changed from normal status to wait status.
4. Specifies the chunk IDs of the chunks whose status is to be changed from wait status to normal status.
5. Specifies the table to be processed (`TABLE01`).

6.4 Return codes

The following table lists and describes the return codes of the `adbchgchunkstatus` command.

Table 6-2: Return codes of the adbchgchunkstatus command

Return code	Description
`0`	The `adbchgchunkstatus` command terminated normally.
`4`	The `adbchgchunkstatus` command terminated normally, but one of the following warning messages was displayed: • The disk containing the message log file is full. • Other warning message was displayed.
`8`	An error occurred during execution of the `adbchgchunkstatus` command. For the cause of the error and corrective action to take, see the displayed error message or the message that has been output to a message log file.

6.5 Handling abnormal termination of the adbchgchunkstatus command

Take the appropriate action according to the displayed message.

If the `adbchgchunkstatus` command terminates abnormally, the status of the specified chunks remains unchanged from before the command was executed.

6.6 Notes

1. Use the `adbcancel` command to cancel execution of the `adbchgchunkstatus` command. Alternatively, use a method such as the `adbstop --force` command or the OS's `kill` command to forcibly terminate the server process. For details about how to use the `adbcancel` command to suspend an executing command, see 1.6 When suspending (forcibly terminating) an executing command.
2. The processing to change chunk status is performed by a server process. This means that using the **Ctrl**+**C** keys to forcibly terminate the `adbchgchunkstatus` command's process will not stop the chunk status change processing that is being performed by the server process (the processing continues). When the processing is completed, the command will output the `KFAA81001-I` message to the message log file.
3. You cannot execute the `adbchgchunkstatus` command on the following tables:
    - Non-updatable tables
    - Tables for which execution of the `adbimport` or `adbidxrebuild` command has been halted

7

adbchgnodetype (Change the HADB Server's Node Type)

This chapter explains the functionality provided by the `adbchgnodetype` command and how to use the command.

7.1 Description

The `adbchgnodetype` command changes the HADB server's node type.

However, normally the `adbchgnodetype` command is executed alone, and the node type cannot be changed.

The `adbchgnodetype` command is used typically in conjunction with a shell command that is specified in the `actcommand` operand in HA Monitor's `servers` file (definition file used to set up an environment for a server).

7.2 Specification format for the adbchgnodetype command

This section explains the specification format and options for the `adbchgnodetype` command.

The following OS users can execute the `adbchgnodetype` command:

- HADB administrator

7.2.1 Explanation of the specification format and options

(1) Specification format

```
adbchgnodetype --master -n HA-Monitor's-server-identification-name
```

(2) Explanation of options

- `--master`
  Specifies that the HADB server's node type is to be changed from slave node to master node.
- `-n` *HA-Monitor's-server-identification-name*
  Specifies the server identification name of the HA Monitor that will be monitoring the HADB server.

7.3 Return codes

The following table lists and describes the return codes of the `adbchgnodetype` command.

Table 7-1: Return codes of the adbchgnodetype command

Return code	Description
`0`	The `adbchgnodetype` command terminated normally.
`4`	The `adbchgnodetype` command issued a warning message and then terminated. Check the displayed message or the message output to a message log file and take appropriate action.
`8`	An error occurred during execution of the `adbchgnodetype` command. For the cause of the error and corrective action to take, see the displayed error message or the message that has been output to a message log file.

8 adbchgsqltrc (Start or Stop Output of SQL Trace Information)

This chapter explains the functionality provided by the `adbchgsqltrc` command and how to use the command.

8.1 Description

The adbchgsqltrc command specifies whether to output SQL trace information, or changes the output content of the SQL trace information while the HADB server is active. It also displays the output status of the SQL trace information.

When the multi-node function is used, the adbchgsqltrc command changes whether to output the SQL trace information of the node on which the command is executed. It also displays the output status of the SQL trace information of the node on which the command is executed.

For details about SQL trace information, see *Running SQL tracing* in *Scheduled Operations* in the *HADB Setup and Operation Guide*.

When you use the adbchgsqltrc command to specify whether to output SQL trace information or change the output content of the SQL trace information, the change made by this command is lost when the HADB server is terminated. To maintain the change at the next startup, apply the change to the server definition.

For details about how to modify the server definition, see the topic *Modifying the server definition* in *Creating and modifying a server definition* in *Building a System* in the *HADB Setup and Operation Guide*.

8.2 Specification format for the adbchgsqltrc command

This section explains the specification format and options for the `adbchgsqltrc` command.

The only OS users who can execute the `adbchgsqltrc` command are HADB administrators.

8.2.1 Explanation of the specification format and options

(1) Specification format

```
adbchgsqltrc {-d
               |-e
               |-s [-y information-whose-output-is-to-start]
                   [-n information-whose-output-is-to-stop]
                   [-l output-level-after-change]
               |-y information-whose-output-is-to-start
                   [-n information-whose-output-is-to-stop]
                   [-l output-level-after-change]
               |-n information-whose-output-is-to-stop
                   [-y information-whose-output-is-to-start]
                   [-l output-level-after-change]
               |-l output-level-after-change
                   [-y information-whose-output-is-to-start]
                   [-n information-whose-output-is-to-stop]
               }
```

(2) Explanation of options

- `-d`

  Displays the output status of the SQL trace information.

  For details about the content that is output when this option is specified, see 8.3.1 Checking the output status of the SQL trace information.

- `-e`

  Specify this option to stop outputting the SQL trace information.

- `-s`

  Specify this option to start outputting the SQL trace information.

- `-y` *information-whose-output-is-to-start*

  ~<character string>((1 to 16 bytes))

  Specifies the type of information to be output to the SQL trace information.

  `param`: Dynamic parameter information

  Outputs dynamic parameter information to the SQL trace information.

  `accesspath`: Access path information and access path statistical information

  Outputs access path information and access path statistical information to the SQL trace information.

  The following rules apply to specifying this information:

  - You can specify multiple information items by separating them with commas (`,`). You can use only commas as delimiters.

- When you specify multiple information items, the order in which they are specified does not matter.
- The same information item cannot be specified more than once.

- `-n` *information-whose-output-is-to-stop*

  ~<character string>((1 to 16 bytes))

  Specifies the type of information whose output to the SQL trace information is to be stopped.

  `param`: Dynamic parameter information

  Does not output dynamic parameter information to the SQL trace information.

  `accesspath`: Access path information and access path statistical information

  Does not output access path information and access path statistical information to the SQL trace information.

  The following rules apply to specifying this information:

  - You can specify multiple information items by separating them with commas (`,`). You can use only commas as delimiters.
  - When you specify multiple information items, the order in which they are specified does not matter.
  - The same information item cannot be specified more than once.

- `-l` *output-level-after-change*

  ~<character string>((1 to 4 bytes))

  Specifies the unit in which to output the SQL trace information.

  `sql`: Each SQL statement

  Outputs the SQL trace information for each SQL statement.

  `call`: Each call

  Outputs the SQL trace information for each call.

(3) Relationships between command options and server definitions

Some of the options of the `adbchgsqltrc` command correspond to the server definition operations related to SQL tracing. The following table shows the relationships between the server definition operations related to SQL tracing and the `adbchgsqltrc` command options.

Table 8-1: Correspondence between the server definition operations related to SQL tracing and adbchgsqltrc command options

No.	Server definition operand name	Server definition specification value	Corresponding adbchgsqltrc command option
1	`adb_sql_trc_out`	`Y`	`-s`
2		`N`	`-e`
3	`adb_sql_trc_param`	`Y`	`-y param`
4		`N`	`-n param`
5	`adb_sql_trc_accesspath`	`Y`	`-y accesspath`
6		`N`	`-n accesspath`
7	`adb_sql_trc_level`	`SQL`	`-l sql`
8		`CALL`	`-l call`

For details about server definitions related to SQL tracing, see the topic *Operands related to SQL statements (set format)* in *Detailed descriptions of the server definition operands* in *Designing the Server Definition* in the *HADB Setup and Operation Guide*.

8.3 Examples

This section uses examples to explain how to accomplish the following tasks using the `adbchgsqltrc` command:

- Checking the output status of the SQL trace information
- Changing whether the SQL trace information is being output
- Changing the output content of the SQL trace information

8.3.1 Checking the output status of the SQL trace information

This command outputs the status of SQL trace information output to standard output.

```
adbchgsqltrc -d

sqltrace param    accesspath level
ACTIVE   ACTIVE   INACTIVE   SQL
```

Explanation of the output items

`sqltrace`

Displays whether the SQL trace information is output.

- `ACTIVE`: The SQL trace information is output.
- `INACTIVE`: The SQL trace information is not output.

`param`

Displays whether dynamic parameter information is output to the SQL trace information.

- `ACTIVE`: Dynamic parameter information is output to the SQL trace information.
- `INACTIVE`: Dynamic parameter information is not output to the SQL trace information.

`accesspath`

Displays whether access path information and access path statistical information are output to the SQL trace information.

- `ACTIVE`: Access path information and access path statistical information are output to the SQL trace information.
- `INACTIVE`: Access path information and access path statistical information are not output to the SQL trace information.

`level`

Displays the unit in which the SQL trace information is output.

- `CALL`: The SQL trace information is output for each call.
- `SQL`: The SQL trace information is output for each SQL statement.

8.3.2 Examples of starting output of the SQL trace information

This subsection explains how to start output of the SQL trace information.

During this step, you can also change the type of SQL trace information to be output and the output level of the SQL trace information.

For items that are not changed, the following operation takes place:

- When the `adbchgsqltrc` command is executed for the first time following HADB server startup, the specification values in the server definition are followed.
- In all other cases, the states that were in effect the previous time the SQL trace information was output are inherited without any change.

> **Note**
>
> To check the current status of the type of SQL trace information to be output and its output level, see 8.3.1 Checking the output status of the SQL trace information.

(1) To start output of the SQL trace information

Start output of the SQL trace information.

Do not change the type of SQL trace information to be output or the unit in which the SQL trace information is output.

```
adbchgsqltrc -s
```

(2) To output dynamic parameter information and access-path-related information

Start output of the SQL trace information.

During this step, change the type of SQL trace information to be output as follows:

- Output dynamic parameter information.
- Output access path information and access path statistical information.

Do not change the unit in which the SQL trace information is output.

```
adbchgsqltrc -s -y param,accesspath
```

(3) To output access-path-related information without outputting dynamic parameter information

Start output of the SQL trace information.

During this step, change the type of SQL trace information to be output as follows:

- Do not output dynamic parameter information.
- Output access path information and access path statistical information.

Do not change the unit in which the SQL trace information is output.

```
adbchgsqltrc -s -n param -y accesspath
```

(4) To output the SQL trace information for each call without outputting access-path-related information

Start output of the SQL trace information.

During this step, do not output access path information or access path statistical information. Do not change whether to output dynamic parameter information.

Additionally, change the unit in which the SQL trace information is output to `call`.

```
adbchgsqltrc -s -n accesspath -l call
```

(5) To output the SQL trace information for each SQL statement

Start output of the SQL trace information.

During this step, change the unit in which the SQL trace information is output to an SQL statement.

Do not change the type of SQL trace information to be output.

```
adbchgsqltrc -s -l sql
```

8.3.3 Examples of changing the output content of the SQL trace information

When the SQL trace information is being output, you can change the type of SQL trace information that is output and the unit in which it is output.

> **Note**
>
> To check the current status of the type of SQL trace information to be output and its output level, see 8.3.1 Checking the output status of the SQL trace information.

(1) To output dynamic parameter information and access-path-related information

Change the type of SQL trace information to be output as follows:

- Output dynamic parameter information.
- Output access path information and access path statistical information.

Do not change the unit in which the SQL trace information is output.

```
adbchgsqltrc -y param,accesspath
```

(2) To output access-path-related information without outputting dynamic parameter information

Change the type of SQL trace information to be output as follows:

- Do not output dynamic parameter information.
- Output access path information and access path statistical information.

Do not change the unit in which the SQL trace information is output.

```
adbchgsqltrc -n param -y accesspath
```

(3) To output the SQL trace information for each call without outputting access-path-related information

Stop output of access path information and access path statistical information. Do not change whether to output dynamic parameter information.

Additionally, change the unit in which the SQL trace information is output to `call`.

```
adbchgsqltrc -n accesspath -l call
```

(4) To output the SQL trace information for each SQL statement

Change the unit in which the SQL trace information is output to `sql`.

Do not change the type of SQL trace information to be output.

```
adbchgsqltrc -l sql
```

8.3.4 Stopping output of SQL trace information

This subsection explains how to stop output of the SQL trace information.

```
adbchgsqltrc -e
```

8.4 Return codes

The following table lists and describes the return codes of the `adbchgsqltrc` command.

Table 8-2: Return codes of the adbchgsqltrc command

Return code	Description
`0`	The `adbchgsqltrc` command terminated normally.
`4`	The `adbchgsqltrc` command issued a warning message and then terminated. Check the displayed message or the message output to a message log file and take appropriate action.
`8`	An error occurred during execution of the `adbchgsqltrc` command. Check the displayed message or the message output to a message log file and take appropriate action.

8.5 Notes

1. In the following cases, any change made by the `adbchgsqltrc` command takes effect together with the transaction that follows the transaction currently being executed:
   - The `adbchgsqltrc` command is used to start output of the SQL trace information.
   - The `adbchgsqltrc` command is used to start output of information that was not output to the SQL trace information.
   - The `adbchgsqltrc` command is used to change the unit (output level) in which the SQL trace information is output.
2. In the following cases, any change made by the `adbchgsqltrc` command takes effect immediately after this command is executed:
   - The `adbchgsqltrc` command is used to stop output of the SQL trace information.
   - The `adbchgsqltrc` command is used to stop output of the information that was being output to the SQL trace information.
3. When you specify the `-y` and `-n` options concurrently for the `adbchgsqltrc` command, you cannot specify the same information for the SQL trace information whose output is to start, and for the SQL trace information whose output is to stop. If the same information is specified for both, an error message is output and the `adbchgsqltrc` command terminates.
4. You cannot specify the `-d` option concurrently with any other option. If the `-d` option is specified concurrently with any other option, an error message is output and then the `adbchgsqltrc` command terminates.
5. You cannot specify the `-e` option concurrently with any other option. If the `-e` option is specified concurrently with any other option, an error message is output and the `adbchgsqltrc` command terminates.
6. Only one `adbchgsqltrc` command can be executed at a time. If multiple `adbchgsqltrc` commands are executed concurrently, an error message is output and the `adbchgsqltrc` commands terminate.
7. Even if you execute the `adbchgsqltrc` command to change whether to output the SQL trace information, or to change the output content of the SQL trace information, the values specified in operands of the server definition file are not changed. To use the change made by the `adbchgsqltrc` command at the next and subsequent startup of the HADB server, you need to change the server definition. For details about how to modify the server definition, see the topic *Modifying the server definition* in *Creating and modifying a server definition* in *Building a System* in the *HADB Setup and Operation Guide*.
8. If you execute the `adbchgsqltrc` command while the HADB server is stopped, an error message is output. If the HADB server is stopped, change the value specified for the operand of the server definition related to SQL tracing.

9 adbchgsrvmode (Change the HADB Server Operation Mode)

This chapter explains the functionality provided by the `adbchgsrvmode` command and how to use the command.

9.1 Description

The `adbchgsrvmode` command changes the HADB server operation mode.

When the multi-node function is used, the operation mode of the master node HADB server can be changed by using the `adbchgsrvmode` command. However, if a node is being returned to the multi-node configuration, the HADB server operation mode of the master node cannot be changed. The HADB server operation mode of the slave node can never be changed (because the HADB server operation mode of the slave node is constantly in quiescence mode).

The four HADB server operation modes are as follows:

- Normal mode
- Quiescence mode
- Offline mode
- Maintenance mode

For details about the HADB server operation modes, see *HADB server operation modes* in *Starting and terminating the HADB server and its operation modes* in *Scheduled Operations* in the *HADB Setup and Operation Guide*.

9.2 Specification format for the adbchgsrvmode command

This section explains the specification format and options for the `adbchgsrvmode` command.

The only OS users who can execute the `adbchgsrvmode` command are HADB administrators.

9.2.1 Explanation of the specification format and options

(1) Specification format

```
adbchgsrvmode {--normal|--quiescence|--offline|--maintenance}
```

(2) Explanation of options

Specifies the new HADB server operation mode that is to be set:

- `--normal`
  Specifies that the HADB server operation mode is to be changed to normal mode.
  The mode can be changed to normal from quiescence, offline, or maintenance.
- `--quiescence`
  Specifies that the HADB server operation mode is to be changed to quiescence mode.
  The mode can be changed to quiescence only if the current HADB server operation mode is normal.
- `--offline`
  Specifies that the HADB server operation mode is to be changed from normal mode to offline mode.
  The mode can be changed to offline only if the current HADB server operation mode is normal.
- `--maintenance`
  Specifies that the HADB server operation mode is to be changed from normal mode to maintenance mode.
  The mode can be changed to maintenance only if the current HADB server operation mode is normal.

9.3 SQL statements and commands that can be executed in each HADB server operation mode

The SQL statements and commands that can be executed depend on the HADB server operation mode. For details about the SQL statements and commands that can be executed in each HADB server operation mode, see 39.3 SQL statements and commands that can be executed in each HADB server operation mode.

9.4 Example

This example changes the HADB server operation mode from normal to quiescence.

```
adbchgsrvmode --quiescence

KFAA80068-I The HADB server operation mode changed from "NORMAL" to "QUIESCENCE".
```

9.5 Return codes

The following table lists and describes the return codes of the `adbchgsrvmode` command.

Table 9-1: Return codes of the adbchgsrvmode command

Return code	Description
`0`	The `adbchgsrvmode` command terminated normally.
`4`	The `adbchgsrvmode` command issued a warning message, and then terminated itself. Check the displayed message or the message output to a message log file and take appropriate action.
`8`	An error occurred during execution of the `adbchgsrvmode` command. Check the displayed message or the message output to a message log file and take appropriate action.

9.6 Notes

1. The `adbchgsrvmode` command must be executed while the HADB server is running.
2. When the HADB server is terminated with the `adbstop` command or terminates abnormally, the previous HADB server operation mode is inherited the next time the HADB server is started. For example, if the HADB server operation mode when the HADB server is terminated is the quiescence mode, the HADB server operation mode will be quiescence the next time the HADB server is started. However, if a different HADB server operation mode is specified in the `adbstart` command, the HADB server is started in the specified mode.
3. The following table lists and describes the SQL statements and commands that can be executed while the `adbchgsrvmode` command is running (while the HADB server operation mode is being changed).

Table 9-2: SQL statements and commands that can be executed while the HADB server operation mode is being changed

No.	SQL statement or command	From normal to quiescence	From quiescence to normal	From normal to offline		From offline to normal		From normal to maintenance	From maintenance to normal
				Execution on the server machine #1	Execution from a machine other than the server machine #2	Execution on the server machine #1	Execution from a machine other than the server machine #2		
1	Retrieval SQL statement	Y#3	Y	Y	N#4	Y	N	N#10	N
2	Update SQL statement	N#3	N	Y	N#4	Y	N	N#10	N
3	Definition SQL statement	N#3	N	Y	N#4	Y	N	N#10	N
4	Control SQL statement	Y#3	Y	Y	N#4	Y	N	N#10	N
5	`adbarchivechunk` command	N#5	N	Y	--	Y	--	N#10	N
6	`adbaudittrail` command	Y	Y	Y	--	Y	--	Y	Y
7	`adbcancel` command	Y	Y	Y	--	Y	--	Y	Y
8	`adbchgchunkcomment` command	N#5	N	Y	--	Y	--	N#10	N
9	`adbchgchunkstatus` command	N#5	N	Y	--	Y	--	N#10	N
10	`adbchgnodetype` command	N	N	N	--	N	--	N	N
11	`adbchgsqltrc` command	Y	Y	Y	--	Y	--	Y	Y
12	`adbchgsrvmode` command#6	N	N	N	--	N	--	N	N
13	`adbclientdefmang` command	Y	Y	Y	--	Y	--	Y	Y
14	`adbcolumnize` command	N	N	Y	--	Y	--	Y	Y

No.	SQL statement or command	From normal to quiescence	From quiescence to normal	From normal to offline		From offline to normal		From normal to maintenance	From maintenance to normal
				Execution on the server machine[#1]	Execution from a machine other than the server machine[#2]	Execution on the server machine[#1]	Execution from a machine other than the server machine[#2]		
15	`adbconvertaudittrailfile` command	Y	Y	Y	--	Y	--	Y	Y
16	`adbdbstatus` command	Y	Y	Y	--	Y	--	N[#10]	N
17	`adbexport` command	Y	Y	Y	--	Y	--	N[#10]	N
18	`adbgetcst` command	N[#5]	N	Y	--	Y	--	N[#10]	N
19	`adbidxrebuild` command	N[#5]	N	Y	--	Y	--	N[#10]	N
20	`adbimport` command	N[#5]	N	Y	--	Y	--	N[#10]	N
21	`adbinfoget` command	Y	Y	Y	--	Y	--	Y	Y
22	`adbinfosweep` command[#7]	N	N	N	--	N	--	N	N
23	`adbinit` command[#7]	N	N	N	--	N	--	N[#10]	N
24	`adbinstall` command[#7]	N	N	N	N	N	N	N	N
25	`adbls` command	Y	Y	Y	--	Y	--	Y	Y
26	`adbmergechunk` command	N[#5]	N	Y	--	Y	--	N[#10]	N
27	`adbmodarea` command	N[#5]	N	Y	--	Y	--	Y	Y
28	`adbmodbuff` command	Y	Y	Y	--	Y	--	N[#10]	N
29	`adbmonitor` command	Y	Y	Y	--	Y	--	Y	Y
30	`adbreorgsystemdata` command	N[#5]	N	Y	--	Y	--	N[#10]	N
31	`adbsql` command[#8]	Y	Y	Y	N	Y	N	N[#10]	N
32	`adbstart` command[#7]	N	N	N	--	N	--	N	N
33	`adbstat` command	Y	Y	Y	--	Y	--	Y	Y
34	`adbstop` command[#9]	N	N	N	--	N	--	N	N
35	`adbsyndict` command	N[#5]	N	Y	--	Y	--	N[#10]	N
36	`adbunarchivechunk` command	N[#5]	N	Y	--	Y	--	N[#10]	N

Legend:

Y: Can be executed while the HADB server operation mode is being changed.

N: Cannot be executed while the HADB server operation mode is being changed (an error results).

--: Not applicable because the command is executed on the server machine on which the HADB server is installed

#1

Execution on the server machine means that the SQL statement or command is executed on the server machine on which the HADB server is installed.

#2

Execution from a machine other than the server machine means that the SQL statement or command is executed on a machine other than the server machine on which the HADB server is installed.

#3

If the HADB server operation mode is changed to quiescence and there is a transaction that has executed an update SQL statement, that transaction is canceled and the connection being used to execute the transaction is closed.

#4

If the HADB server operation mode is changed to offline and there is a connection from a machine other than the server machine on which the HADB server is installed, the transaction using that connection is canceled and the connection is closed.

#5

If a command is running when the HADB server operation mode is changed to quiescence, that command is canceled.

#6

Only one `adbchgsrvmode` command can be executed at a time.

#7

This command cannot be executed while the HADB server is running.

#8

For details about the SQL statements that can be executed, see Nos. 1 through 4.

#9

The `adbchgsrvmode` and `adbstop` commands cannot be executed concurrently.

#10

If the operation mode changes to maintenance and there is a connection for commands that cannot be executed in maintenance mode, the transaction using that connection is canceled and the connection is closed.

10 adbclientdefmang (Centralized Management of Client Definitions)

This chapter explains the functionality provided by the `adbclientdefmang` command and how to use the command.

10.1 Description

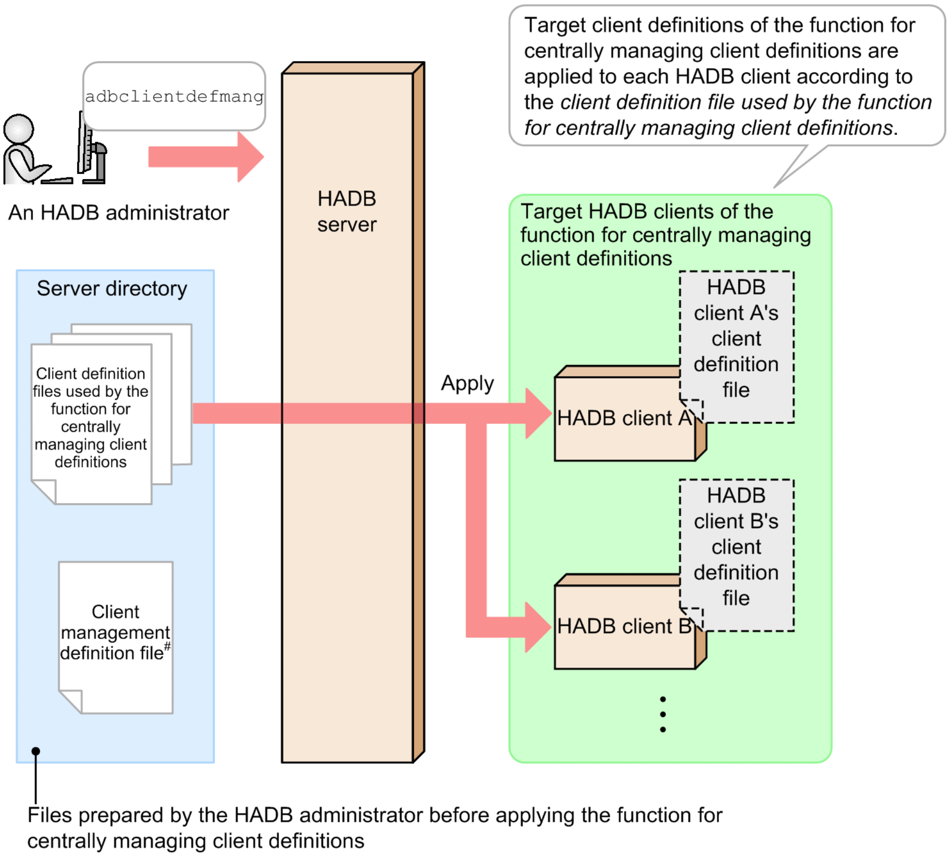
The adbclientdefmang command is used to apply the function for centrally managing client definitions.

Execute the adbclientdefmang command when you want to change the way the function for centrally managing client definitions is applied, while the HADB server is running.

When executed, the adbclientdefmang command establishes a connection to the HADB server.

The following figure gives an overview of changing the way the function for centrally managing client definitions is applied.

Figure 10-1: Changing the way the function for centrally managing client definitions is applied


adbclientdefmang
An HADB administrator
HADB server
Server directory
Client definition files used by the function for centrally managing client definitions
Client management definition file#
Apply
Target client definitions of the function for centrally managing client definitions are applied to each HADB client according to the *client definition file used by the function for centrally managing client definitions*.
Target HADB clients of the function for centrally managing client definitions
HADB client A's client definition file
HADB client A
HADB client B's client definition file
HADB client B
Files prepared by the HADB administrator before applying the function for centrally managing client definitions

#
Specifies bindings between HADB client authorization identifiers, and the *client definition file used by the function for centrally managing client definitions*.

Explanation:

When you execute the adbclientdefmang command, the content of the *client definition file used by the function for centrally managing client definitions* pertaining to the client definitions to be centrally managed, is applied to each HADB client.

Create the following files before executing the adbclientdefmang command:

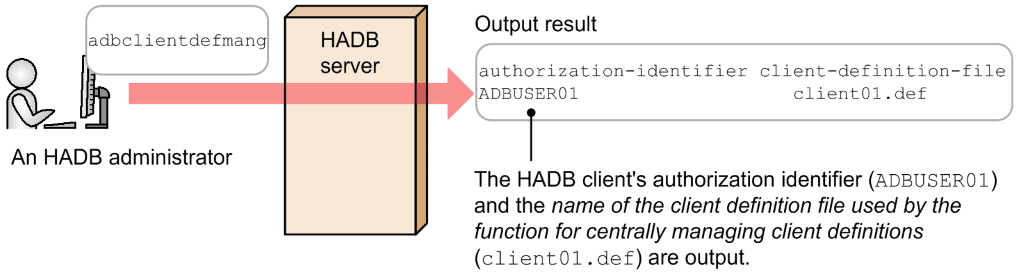
- Client management definition file
- Client definition file used by the function for centrally managing client definitions

For details about each file, see *Files required for the function for centrally managing client definitions* in the *HADB Setup and Operation Guide*.

You can also execute the adbclientdefmang command to check the correspondence between the *target HADB clients of the function for centrally managing client definitions* and the *name of the client definition file used by the function for centrally managing client definitions* (check the application status of the function for centrally managing client definitions).

The following figure gives an overview of checking the application status of the function for centrally managing client definitions.

Figure 10-2: Checking the application status of the function for centrally managing client definitions



```
authorization-identifier client-definition-file
ADBUSER01                         client01.def
```

The HADB client's authorization identifier (ADBUSER01) and the *name of the client definition file used by the function for centrally managing client definitions* (client01.def) are output.

Explanation:

When you execute the adbclientdefmang command, the authorization identifier of the HADB client and the corresponding *name of the client definition file used by the functionality for centrally managing client definitions* are output.

10.2 Specification format for the adbclientdefmang command

This section explains the specification format and options for the `adbclientdefmang` command.

The only OS users who can execute the `adbclientdefmang` command are HADB administrators.

10.2.1 Explanation of the specification format and options

(1) Specification format

```
adbclientdefmang -u authorization-identifier
                 [-p password]
                 {--update |[-i output-target-authorization-identifier]}
```

(2) Explanation of options

● `-u` *authorization-identifier*

~<character string>((1 to 100 bytes))

Specifies the authorization identifier of the HADB user who executes the `adbclientdefmang` command. To execute the `adbclientdefmang` command, the user must have the following two privileges:

- `DBA` privilege
- `CONNECT` privilege

For this option, specify an authorization identifier in the range from 1 to 100 bytes. Note that the byte count (1 to 100 bytes) does not include the double quotation marks used to enclose the authorization identifier.

> **Important**
>
> If the character string used as the authorization identifier includes any lowercase letter or a backslash (`\`), make sure that you check the rules for specifying authorization identifiers. For the rules for specifying authorization identifiers, see 1.4.2 Rules for specifying authorization identifiers and passwords.

● `-p` *password*

~<character string>((1 to 255 bytes))

Specifies the password for the authorization identifier that was specified in the `-u` option.

> **Important**
>
> If the password includes a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), make sure that you check the rules for specifying passwords. For the rules for specifying passwords, see 1.4.2 Rules for specifying authorization identifiers and passwords.

If the `adbclientdefmang` command is executed with this option omitted, a message prompting the user to enter a password is displayed. In an environment in which a password cannot be entered from the standard input, such as when the command is executed in the background, make sure that you do not omit this option.

- `--update`

  Specify this when you want to change the way the function for centrally managing client definitions is applied, while the HADB server is running.

  When you execute the `adbclientdefmang` command with the `--update` option specified, the content of the *client definition file used by the function for centrally managing client definitions* is applied to each HADB client according to the content of the client management definition file (`$ADBDIR/conf/adbclientdefmang.def`).

  However, if the `adbclientdefmang` command is executed with the `--update` option specified in the following situations, the function for centrally managing client definitions is disabled:

  - The client management definition file does not exist.
  - The client management definition file is empty.
  - The content of the client management definition file consists entirely of comments.

- `-i` *output-target-authorization-identifier*

  ~<character string>((1 to 100 bytes))

  Specify this when you want to check the *name of the client definition file used by the function for centrally managing client definitions* applied to the authorization identifier of a specific HADB client.

  When you execute the `adbclientdefmang` command with the `-i` option specified, the authorization identifier of the HADB client specified in the `-i` option, and the *name of the client definition file used by the function for centrally managing client definitions* applied to that authorization identifier are displayed.

  In this option, specify an authorization identifier for the output target in the range from 1 to 100 bytes. Note that the byte count (1 to 100 bytes) does not include the double quotation marks used to enclose the authorization identifier of the output target.

  Specifying a character string that violates the rules for specifying authorization identifiers results in an error.

  > **Important**
  >
  > If the character string used as the authorization identifier includes any lowercase letter or a backslash (\), make sure that you check the rules for specifying authorization identifiers. For the rules for specifying authorization identifiers, see 1.4.2 Rules for specifying authorization identifiers and passwords.

  The following is an example of displaying the *name of the client definition file used by the function for centrally managing client definitions* applied to the HADB client authorization identifier `ADBUSER02`:

  Example execution (when the `-i` option is specified)

  ```
  adbclientdefmang -u ADBUSER01 -p '#HelloHADB_01' -i ADBUSER02
  ```

  Output example

  ```
  authorization-identifier client-definition-file
  ADBUSER02                client01.def
  ```

  Note that if you omit the `-i` option, the authorization identifiers of HADB clients to be centrally managed, and the list of *names of client definition files used by the function for centrally managing client definitions* are output.

  Example execution (when the `-i` option is omitted)

  ```
  adbclientdefmang -u ADBUSER01 -p '#HelloHADB_01'
  ```

  Output example

  ```
  authorization-identifier client-definition-file
  ADBUSER01                client01.def
  ```

```
ADBUSER02                client01.def
ADBUSER03                client02.def
```

10.2.2 Client management definition format

A *client management definition* is a definition specified in a client management definition file. A *client management definition file* defines correspondences between *client definition files used by the function for centrally managing client definitions*, and the authorization identifiers of HADB clients to which you want to apply the content of the client definition files.

For details about client management definition, see *Files required for the function for centrally managing client definitions* in the *HADB Setup and Operation Guide*.

10.3 Examples

This section uses examples to explain how to accomplish the following tasks using the `adbclientdefmang` command:

- Applying the content of the function for centrally managing client definitions
- Checking the name of the client definition file applied to a specific HADB client

10.3.1 Applying the content of the function for centrally managing client definitions

Apply the content of the function for centrally managing client definitions to its target HADB clients.

```
adbclientdefmang -u ADBUSER01 -p '#HelloHADB_01' --update
```

When you execute the `adbclientdefmang` command with the `--update` option specified, the content of the *client definition file used by the function for centrally managing client definitions* is applied to each HADB client.

10.3.2 Checking the name of the client definition file applied to a specific HADB client

Check the *name of the client definition file used by the function for centrally managing client definitions* applied to the authorization identifier of a specific HADB client.

```
adbclientdefmang -u ADBUSER01 -p '#HelloHADB_01' -i ADBUSER02
```

Output example

```
authorization-identifier client-definition-file
ADBUSER02                client02.def
```

The *name of the client definition file used by the function for centrally managing client definitions* (`client02.def`) corresponding to the HADB client authorization identifier `ADBUSER02` is output. The content of the output client definition file is applied to the HADB client authorization identifier `ADBUSER02`.

10.4 Return codes

The following table lists and describes the return codes of the `adbclientdefmang` command.

Table 10-1: Return codes of the adbclientdefmang command

Return code	Description
`0`	The `adbclientdefmang` command terminated normally.
`4`	The `adbclientdefmang` command terminated normally, but a warning message was issued. Check the displayed message or the message output to a message log file and take appropriate action.
`8`	An error occurred during execution of the `adbclientdefmang` command. For the cause of the error, see the displayed error message or the message that has been output to a message log file.

10.5 Notes

1. The `adbclientdefmang` command can be executed while the HADB server is running.
2. When you execute the `adbclientdefmang` command with the `--update` option specified, changes to an AP that has connected to the HADB server through an HADB client authorization identifier using the function for centrally managing client definitions are not applied. The content of the function for centrally managing client definitions is applied the next time the AP is connected after it is disconnected from the HADB server.
3. If the `adbclientdefmang` command is executed with the `--update` option omitted while the function for centrally managing client definitions is disabled, only the *`authorization-identifier client-definition-file`* is output.
4. Only one `adbclientdefmang` command can be executed at a time. If you execute multiple `adbclientdefmang` commands at the same time, an error occurs.
5. To use multi-node function, execute the `adbclientdefmang` command on the master node. The `adbclientdefmang` command cannot be executed on a slave node.
6. To use the multi-node function, make sure the content of all nodes' client management definition files is consistent before executing the `adbclientdefmang` command with the `--update` option specified.

   If the `adbclientdefmang` command is executed with the `--update` option specified when the content of all nodes' client management definition files is inconsistent, an error occurs.
7. If you execute the `adbclientdefmang` command while the HADB server is stopped, an error message is output. If you want to change the content of the function for centrally managing client definitions while the HADB server is stopped, change the client management definition file, and then start the HADB server.

11 adbcolumnize (Manage the Updated-Row Columnizing Facility)

This chapter explains the functionality provided by the `adbcolumnize` command and how to use the command.

11.1 Description

The `adbcolumnize` command is executed to manage the updated-row columnizing facility.

For details about the updated-row columnizing facility, see *Using the updated-row columnizing facility (maintaining the retrieval performance for column store tables)* in the *HADB Setup and Operation Guide*.

The `adbcolumnize` command allows you to perform the following three operations:

- Enabling the updated-row columnizing facility
- Disabling the updated-row columnizing facility
- Displaying the status of the updated-row columnizing facility

11.2 Specification format for the adbcolumnize command

This section explains the specification format and options for the `adbcolumnize` command.

The only OS users who can execute the `adbcolumnize` command are HADB administrators.

11.2.1 Explanation of the specification format and options

(1) Specification format

```
adbcolumnize {--start | --stop | -d}
```

(2) Explanation of options

- `--start`

  Specify this option if you want to enable the updated-row columnizing facility.

  If this option is specified, the updated-row columnizing facility is enabled and maintenance processing starts.

- `--stop`

  Specify this option if you want to disable the updated-row columnizing facility.

  If this option is specified, the updated-row columnizing facility is disabled and maintenance processing does not start. Currently running maintenance processing is interrupted.

- `-d`

  Specify this option if you want to check the status of the updated-row columnizing facility.

  By specifying this option, you can check whether the updated-row columnizing facility is enabled or disabled.

11.3 Examples

This section uses examples to explain how to accomplish the following tasks by using the `adbcolumnize` command:

- Enable the updated-row columnizing facility
- Display the status of the updated-row columnizing facility

11.3.1 Enabling the updated-row columnizing facility

This example enables the updated-row columnizing facility.

```
adbcolumnize --start
```

Executing the `adbcolumnize --start` command enables the updated-row columnizing facility.

11.3.2 Displaying the status of the updated-row columnizing facility

This example displays the status of the updated-row columnizing facility.

```
adbcolumnize -d
```

Executing the `adbcolumnize -d` command outputs the status of the updated-row columnizing facility to the standard output. You can check whether the updated-row columnizing facility is enabled or disabled from the output result.

Output example

```
STATUS
ACTIVE
```

Explanation of the output items

`STATUS`:

The value of this item indicates the status of the updated-row columnizing facility. One of the following is displayed:

- `ACTIVE`
  The updated-row columnizing facility is enabled.
- `INACTIVE`
  The updated-row columnizing facility is disabled.

11.4 Return codes

The following table lists and describes the return codes of the `adbcolumnize` command.

Table 11-1: Return codes of the adbcolumnize command

Return code	Description
`0`	The `adbcolumnize` command terminated normally.
`4`	A warning message was output during execution of the `adbcolumnize` command. For details about the warning, see the displayed message or the message that was output to a message log file.
`8`	An error occurred during execution of the `adbcolumnize` command. For the cause of the error, see the displayed message or the message that was output to a message log file.

11.5 Notes

1. The `adbcolumnize` command can be executed while the HADB server is running.
2. If the multi-node function is used, you can execute the `adbcolumnize` command on the master node only.
3. If the multi-node function is used, you cannot execute the `adbcolumnize` command while a node is returning to the multi-node configuration.
4. You cannot execute the `adbcolumnize` command while the HADB server is operating in quiescence mode.
5. You can execute the `adbcolumnize` command on a server machine only.
6. The status of the updated-row columnizing facility changed by the `adbcolumnize` command is retained even after the HADB server is restarted. Therefore, if the updated-row columnizing facility is enabled before the HADB server is restarted, you do not need to re-enable the facility by using the `adbcolumnize` command after the HADB server is restarted.

12 adbconvertaudittrailfile (Convert the Audit Trail File)

This chapter explains the functionality provided by the `adbconvertaudittrailfile` command and how to use the command.

12.1 Description

This section explains the functionality provided by the `adbconvertaudittrailfile` command and how to swap common format audit trail files.

When executed, the `adbconvertaudittrailfile` command establishes a connection to the HADB server.

Note that you can execute this command only when the audit trail facility is enabled.

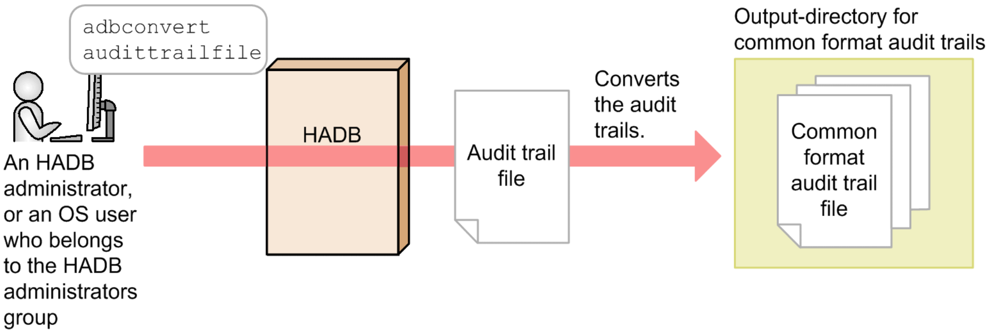
12.1.1 Functional overview

Execute the `adbconvertaudittrailfile` command when you use JP1/Audit Management - Manager (JP1/Audit) to collect and manage audit trails of the HADB server.

When the `adbconvertaudittrailfile` command is executed, the audit trails in the audit trail file can be converted into the format (common format) that can be used by JP1/Audit. The audit trails that are converted by the `adbconvertaudittrailfile` command are output to common format audit trail files.

The following figure provides an overview of the processing of the `adbconvertaudittrailfile` command.

Figure 12-1: Overview of the processing of the `adbconvertaudittrailfile` command


adbconvert
audittrailfile
An HADB administrator, or an OS user who belongs to the HADB administrators group
HADB
Audit trail file
Converts the audit trails.
Output-directory for common format audit trails
Common format audit trail file

Explanation:

When the `adbconvertaudittrailfile` command is executed, the audit trails in the audit trail file on the HADB server are converted to the format (common format) that can be used by JP1/Audit. The conversion results are output to the *common format audit trail files* under the *output-directory for common format audit trails*. This processing is called *conversion of audit trail files*.

■ **Common format audit trail files**

The audit trails converted by the `adbconvertaudittrailfile` command are output to these files. The converted audit trails can be collected and managed by JP1/Audit.

For details about common format audit trail files, see 12.1.2 Swapping common format audit trail files.

For details about coordination of the audit trail facility and JP1/Audit, see *Linkage between the audit trail facility and JP1/Audit Management - Manager* in the *HADB Setup and Operation Guide*.

Note

The current audit trail file cannot be converted. If you want to convert the current audit trail file, you must execute the `adbaudittrail --swap` command to swap the current audit trail file.

12.1.2 Swapping common format audit trail files

The audit trails converted by the `adbconvertaudittrailfile` command are output to the common format audit trail files under the output-directory for common format audit trails.

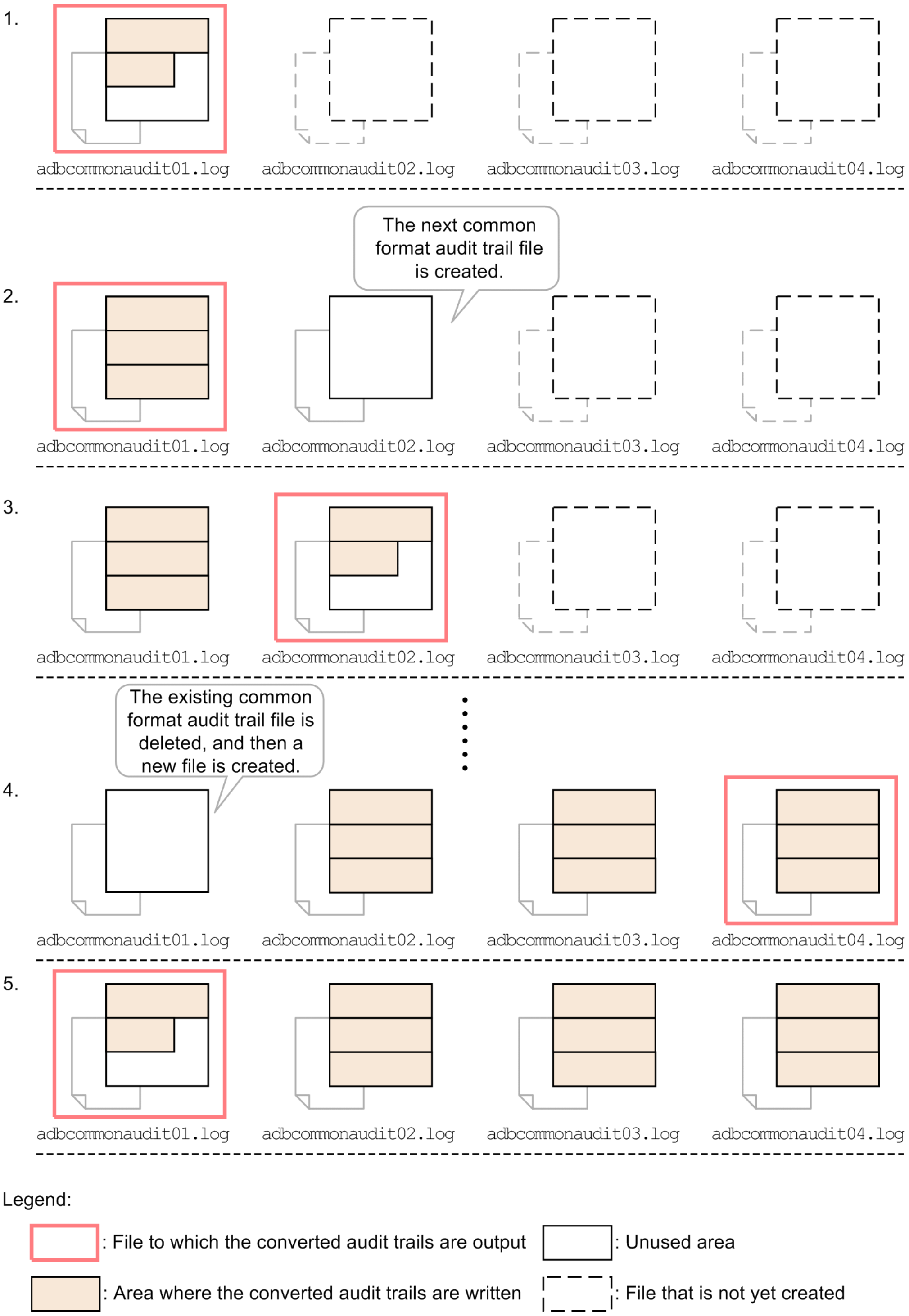
Up to four common format audit trail files (`adbcommonaudit`*`XX`*`.log`) are created. *XX* in the file name is a sequential number in the range from 01 to 04.

When the size of a common format audit trail file to which converted audit trails are output reaches the maximum (512 megabytes), the HADB server creates a new common format audit trail file. Then, the HADB server swaps to the newly created common format audit trail file to continuously output the converted audit trails. This processing is called *swapping common format audit trail files*.

If a common format audit trail file already exists in the swap destination, the existing common format audit trail file is temporarily deleted. A new file with the same file name is then created, and output of the converted audit trails continues.

The following figure provides an overview of swapping common format audit trail files.

Figure 12-2: Swapping common format audit trail files



Explanation:

1. When the `adbconvertaudittrailfile` command is executed, the HADB server creates a common format audit trail file (`adbcommonaudit01.log`) and outputs the converted audit trails.
2. When the size of the common format audit trail file (`adbcommonaudit01.log`) reaches the maximum (512 megabytes), a new common format audit trail file (`adbcommonaudit02.log`) is created as the next output destination.
3. The HADB server changes the output destination to the new common format audit trail file (`adbcommonaudit02.log`) and continuously outputs audit trails.

4. The HADB server repeats the steps from 1 to 3. When the size of the common format audit trail file (`adbcommonaudit04.log`) reaches the maximum, the 1st common format audit trail file (`adbcommonaudit01.log`) is deleted. Then, a new file with the same file name is created.

5. The HADB server changes the output destination to the new common format audit trail file (`adbcommonaudit01.log`) and continuously outputs audit trails.

12.2 Specification format for the adbconvertaudittrailfile command

This section explains the specification format and options for the `adbconvertaudittrailfile` command.

The following OS users can execute the `adbconvertaudittrailfile` command:

- HADB administrator
- OS users who belong to the HADB administrators group

12.2.1 Explanation of the specification format and options

(1) Specification format

```
adbconvertaudittrailfile -u authorization-identifier
                        [-p password]
                         --common
                         -d output-directory-path-for-common-format-audit-trail
                         audit-trail-file-path
```

(2) Explanation of options

- `-u` *authorization-identifier*

  ~<character string>((1 to 100 bytes))

  Specifies the authorization identifier of the HADB user who executes the `adbconvertaudittrailfile` command. To execute the `adbconvertaudittrailfile` command, the user must have the following two privileges:

  - `CONNECT` privilege
  - Audit viewer privilege

  For this option, specify an authorization identifier in the range from 1 to 100 bytes. Note that the byte count (1 to 100 bytes) does not include the double quotation marks used to enclose the authorization identifier.

  Important

  If the character string used as the authorization identifier includes any lowercase letter or a backslash (`\`), make sure that you check the rules for specifying authorization identifiers. For rules about specifying authorization identifiers, see 1.4.2 Rules for specifying authorization identifiers and passwords.

- `-p` *password*

  ~<character string>((1 to 255 bytes))

  Specifies the password for the authorization identifier specified in the `-u` option.

  Important

  If the password includes a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), make sure that you check the rules for specifying passwords. For the rules for specifying passwords, see 1.4.2 Rules for specifying authorization identifiers and passwords.

If the `adbconvertaudittrailfile` command is executed with this option omitted, a message prompting the user to enter a password will be displayed. In an environment in which a password cannot be entered from the standard input, such as when the command is executed in the background, make sure that you do not omit this option.

- `--common`

  Specify this option when you want to convert an audit trail file into a common format audit trail file.

- `-d` *output-directory-path-for-common-format-audit-trail*

  ~<path name>((2 to 510 bytes))

  Specifies the absolute path for the output-directory for common format audit trails.

  Create the output-directory for common format audit trails before executing the `adbconvertaudittrailfile` command. For details, see *Environment settings for the HADB server* in *Environment settings for linking the audit trail facility with JP1/Audit* in the *HADB Setup and Operation Guide*.

- *audit-trail-file-path*

  ~<character string>((2 to 1,024 bytes))

  Specifies the absolute path for the audit trail file to be converted.

  However, the current audit trail file cannot be converted. If you want to convert the current audit trail file, execute the `adbaudittrail --swap` command to swap the current audit trail file. After that, for this option specify the absolute path of the renamed audit trail file.

  Note that you can also convert audit trail files that are compressed in GZIP format.

12.3 Examples

This section uses examples to explain how to use the adbconvertaudittrailfile command to convert the audit trail file.

12.3.1 When converting the latest audit trail file

Use the adbconvertaudittrailfile command to convert the audit trails in the audit trail file, and then output the converted audit trails to the common format audit trail file. This example uses the following procedure for executing the adbconvertaudittrailfile command.

Note that the descriptions of this procedure assume that the output-directory for common format audit trails (/mnt/audittrail/adbcommon/) is already created.

Procedure

1. Swap the current audit trail file.
2. Identify the renamed audit trail file.
3. Execute the adbconvertaudittrailfile command.

Using an example, the following subsections explain each of these steps.

(1) Swapping the current audit trail file

Specify the authorization identifier of the HADB user who has the audit admin privilege, and then execute the adbaudittrail --swap command to swap the current audit trail file.

Specification example of the adbaudittrail

```
adbaudittrail -u ADBAUDITADMIN -p '#HelloHADB_ADMIN' --swap
```

(2) Identifying the renamed audit trail file

Check the KFAA81401-I message that is output when the current audit trail file is swapped, and identify the path of the renamed audit trail file (the latest audit trail file that can be converted). The path of the renamed audit trail file is underlined in the following example.

KFAA81401-I message output example

```
KFAA81401-I The file to which the audit trail is output was swapped.
(file name = /mnt/audittrail/outputarea/audit/adbaud-20180101-123456-789.aud)
```

(3) Executing the adbconvertaudittrailfile command

```
adbconvertaudittrailfile
  -u ADBAUDITVIEWER                                                 ...1
  -p '#HelloHADB_VIEWER'                                            ...2
  --common                                                          ...3
  -d /mnt/audittrail/adbcommon/                                     ...4
  /mnt/audittrail/outputarea/audit/adbaud-20180101-123456-789.aud   ...5
```

Explanation:

1. Specify the authorization identifier of an HADB user who has the audit viewer privilege.
2. Specify the password for the authorization identifier of the HADB user.
3. Specify the `--common` option.
4. Specify the path of the output-directory for common format audit trails.
5. Specify the path (identified as described in (2) Identifying the renamed audit trail file) of the renamed audit trail file.

12.4 Return codes

The following table lists and describes the return codes of the `adbconvertaudittrailfile` command.

Table 12-1: Return codes of the adbconvertaudittrailfile command

Return code	Description
0	The `adbconvertaudittrailfile` command terminated normally.
4	The `adbconvertaudittrailfile` command terminated normally, but a warning message or an error message was output. Check the displayed message or the message output to a message log file and take appropriate action.
8	An error occurred during execution of the `adbconvertaudittrailfile` command. For the cause of the error, see the displayed message or the message that has been output to a message log file.

12.5 Notes

1. You can execute the `adbconvertaudittrailfile` command when the audit trail facility is enabled.
2. The character encoding of common format audit trail files is the same as the character encoding of the HADB server that outputs the audit trail files to be converted.
3. To execute the `adbconvertaudittrailfile` command, all the following conditions must be met. If the conditions are not met, the `adbconvertaudittrailfile` command results in an error.
    - An existing directory is specified for the `-d` option.
    - Read, write, and execution privileges are set so that the HADB administrator can access the directory specified for the `-d` option.
4. You cannot convert the current audit trail file. If you want to convert the current audit trail file, execute the `adbaudittrail --swap` command to swap the current audit trail file. Then, specify the renamed audit trail file for the *audit-trail-file-path* for the `adbconvertaudittrailfile` command to perform the conversion.
5. You can use the `adbcancel` command to forcibly terminate the processing of the `adbconvertaudittrailfile` command.
6. Only one `adbconvertaudittrailfile` command can be executed at a time. If multiple instances of the `adbconvertaudittrailfile` command are executed simultaneously, an error occurs. However, if the multi-node function is used and the commands are executed on different nodes, multiple instances of the `adbconvertaudittrailfile` command can be executed simultaneously.
7. If the multi-node function is used, execute the `adbconvertaudittrailfile` command on the nodes where the audit trail files to be converted exist.
8. If the specified audit trail file is corrupted, the audit trails in the file might not be converted. Alternatively, the audit trails at and from the corrupted location might not be converted.
9. The common format audit trail files are swapped by using the round-robin method. For this reason, when the 4th common format audit trail file is full, the 1st common format audit trail file is deleted. Then, a new 1st common format audit trail file is created.

    If deletion of the common format audit trail file might be problematic, back up the common format audit trail file as necessary.
10. When the `adbconvertaudittrailfile` command is used to convert more than two million audit trails at a time, the common format audit trail files might be wrapped at a high speed. In this situation, JP1/Audit might not collect all audit trails. We recommend that you specify the default value (256 megabytes) for the `adb_audit_log_max_size` operand in the server definition, so that the number of audit trail files to be output to one audit trail file is no more than two million. For details about the `adb_audit_log_max_size` operand in the server definition, see the *Operands related to audit trail facility (set format)* in *Detailed descriptions of the server definition operands* in the *HADB Setup and Operation Guide*.

13 adbdbstatus (Analyze the Database Status)

This chapter explains the functionality provided by the `adbdbstatus` command and how to use the command.

Note that the term *tables* in this chapter means base tables. In output examples, items are arranged for readability. In the actual output results, all items are left-aligned.

13.1 Description

This section explains the functionality provided by the adbdbstatus command and the types of output results.

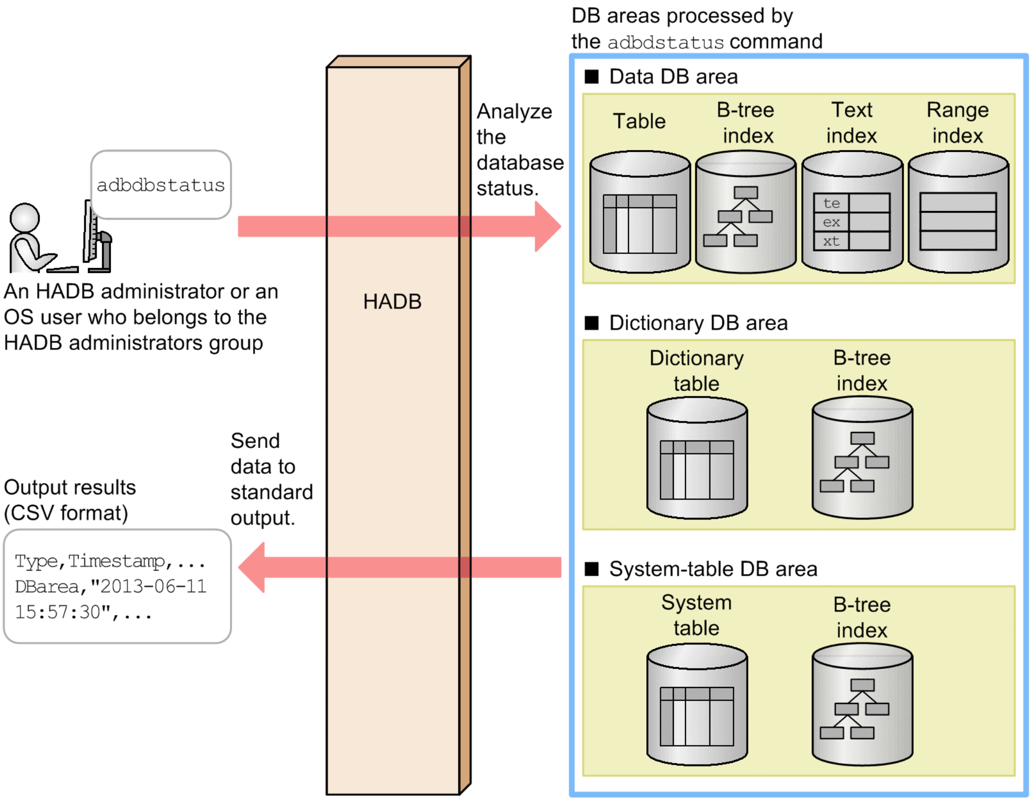
When executed, the adbdbstatus command establishes a connection to the HADB server.

13.1.1 Functional overview

The adbdbstatus command outputs such information as the amount of database space being used, the status of the database, and whether a column store table must be reorganized. The adbdbstatus command processes the data DB area, dictionary DB area, and system-table DB area.

The following figure provides an overview of the adbdbstatus command's processing.

Figure 13-1: Overview of the adbdbstatus command's processing



Explanation:

When the adbdbstatus command is executed, it outputs information about the data, dictionary, and system-table DB areas to the standard output in CSV format. You can redirect output results to files and then forward the files to spreadsheet software so that you can check information in tabular format, including the database usage, database status, and whether column store tables must be reorganized.

13.1.2 Types of results output by the adbdbstatus command

The adbdbstatus command outputs three types of results: *summary information*, *usage information*, and *information about the need for reorganization*.

The following explains the contents of each type of information.

Summary information is further classified into four types, one for each type of analysis target. Usage information is further classified into two types.

Summary information

- **Summary information for DB areas**

  The usage of DB areas is output. For details about the items that are output as summary information for DB areas, see 13.3 Items that are output as summary information for DB areas.

- **Summary information for tables**

  The usage and status of tables are output. For details about the items that are output as summary information for tables, see 13.4 Items that are output as summary information for tables.

- **Summary information for indexes**

  The usage and status of indexes are output. For details about the items that are output as summary information for indexes, see 13.5 Items that are output as summary information for indexes.

- **Summary information of archived chunks**

  The usage and status of chunks created in an archivable multi-chunk table are output. For details about the items that are output as summary information of archived chunks, see 13.6 Items that are output in the summary information of archived chunks.

Usage information

- **Usage information for DB areas, tables, and indexes**

  Detailed usage information for DB areas, tables, and indexes is output.

  Usage information for a multi-chunk table is output for each chunk in the table. For details about the items that are output as usage information for DB areas, tables, and indexes, see 13.7 Items that are output as usage information for DB areas, tables, and indexes.

- **Usage information of archived chunks**

  Detailed usage information of archived chunks created in an archivable multi-chunk table, and detailed usage information of archive files, which contain archived chunk data, corresponding to those chunks are output.

  For details about the items that are output as usage information of archived chunks, see 13.8 Items that are output in the usage information of archived chunks.

Information about the need for reorganization

Information about whether the column store table must be reorganized is output.

If the column store table is a multi-chunk table, information about the need for reorganization is output for each chunk in the table. However, information about deletion-pending chunks is not output. For details about the items that are output as information about the need for reorganization, see 13.9.2 List of items that are output as information about the need for reorganization.

13.2 Specification format for the adbdbstatus command

This section explains the specification format and options for the `adbdbstatus` command.

The following OS users can execute the `adbdbstatus` command:

- HADB administrator
- OS users who belong to the HADB administrators group

The specification format of the `adbdbstatus` command differs depending on the type of information to be output.

- To output *summary information*:
  See 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).
- To output *usage information*:
  See 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).
- To output *information about the need for reorganization* of column store tables:
  See 13.2.2 Explanation of the specification format and options (outputting information about the need for reorganization of column store tables).

13.2.1 Explanation of the specification format and options (outputting summary information or usage information)

(1) Specification format

The following indicates the specification format of the `adbdbstatus` command for outputting *summary information* or *usage information*.

For details about the specification format to output *information about the need for reorganization* of column store tables, see (1) Specification format in 13.2.2 Explanation of the specification format and options (outputting information about the need for reorganization of column store tables).

```
adbdbstatus
  [-d {summary|used}]
  [-c {dbarea|table|index|archivechunk}]
  [-n {DB-area-name|table-name|index-name}]
  [-t]
  [-k enclosing-character]
  [-s {delimiter|tab}]
  [-S {K|M|G}]
  [-r range-specification]
  [--shared-lock]
```

(2) Relationships among specified options and output information

When the `adbdbstatus` command outputs *summary information* or *usage information*, the type of information to be output is determined by the combination of the following options:

- `-d` option

- `-c` option
- `-n` option

The table below shows the relationships among the DB status analysis pattern to be executed, the type of information that is output, and the combination of options. For details about the options, see the descriptions of the -d, -c, and -n options in (3) Explanation of options.

Table 13-1: Relationships among the DB status analysis pattern to be executed, the type of information that is output, and the combination of options

No.	DB status analysis pattern		Type of information that is output	Combination of options		
	Target of analysis	Information to be analyzed		-d option specification	-c option specification	-n option specification
1	Entire database (all DB areas)	Usage of DB areas	Summary information for DB areas	`summary` or omitted	`dbarea` or omitted	--
2		Usage of DB area files and chunks	Usage information for DB areas, tables, and indexes	`used`	`dbarea` or omitted	--
3	Specific DB area	Usage of a DB area	Summary information for DB areas	`summary` or omitted	`dbarea` or omitted	*DB-area-name*
4		Usage of DB area's files and chunks	Usage information for DB areas, tables, and indexes	`used`	`dbarea` or omitted	*DB-area-name*
5	All tables	Usage and status of tables	Summary information for tables	`summary` or omitted	`table`	--
6		Usage of DB area files in DB areas where tables and chunks are stored	Usage information for DB areas, tables, and indexes	`used`	`table`	--
7	Specific table	Usage and status of a table	Summary information for tables	`summary` or omitted	`table`	*table-name*
8		Usage of DB area files in DB areas where tables and chunks are stored	Usage information for DB areas, tables, and indexes	`used`	`table`	*table-name*
9	All indexes	Usage and status of indexes	Summary information for indexes	`summary` or omitted	`index`	--
10		Usage of DB area files in DB areas where indexes and chunks are stored	Usage information for DB areas, tables, and indexes	`used`	`index`	--
11	Specific index	Usage and status of an index	Summary information for indexes	`summary` or omitted	`index`	*index-name*
12		Usage of DB area files in DB areas where indexes and chunks are stored	Usage information for DB areas, tables, and indexes	`used`	`index`	*index-name*

No.	DB status analysis pattern		Type of information that is output	Combination of options		
	Target of analysis	Information to be analyzed		-d option specification	-c option specification	-n option specification
13	Archived chunk created in a specific table	Usage and status of chunks created in a multi-chunk table	Summary information of archived chunks	`summary` or omitted	`archivechunk`	*table-name*
14		Usage information of each archive file corresponding to an archived chunk	Usage information of archived chunks	`used`	`archivechunk`	*table-name*

Legend:

--: The `-n` option is omitted.

(3) Explanation of options

The following describes the options of the `adbdbstatus` command used to output *summary information* or *usage information*. For details about the options used to output *information about the need for reorganization* of column store tables, see (2) Explanation of options in 13.2.2 Explanation of the specification format and options (outputting information about the need for reorganization of column store tables).

- `-d {summary|used}`

  Specifies the type of output results:

  `summary`: Summary information

  For details about the items that are displayed as the summary information, see the following:

  - 13.3 Items that are output as summary information for DB areas
  - 13.4 Items that are output as summary information for tables
  - 13.5 Items that are output as summary information for indexes
  - 13.6 Items that are output in the summary information of archived chunks

  `used`: Usage information

  For details about the items that are output as the usage information, see the following:

  - 13.7 Items that are output as usage information for DB areas, tables, and indexes
  - 13.8 Items that are output in the usage information of archived chunks

- `-c {dbarea|table|index|archivechunk}`

  Specifies the range of information to be collected by the `adbdbstatus` command:

  `dbarea`: DB area information

  Information about DB areas is to be collected.

  `table`: Table information

  Information about tables is to be collected.

  `index`: Index information

  Information about indexes is to be collected.

  `archivechunk`: Archived chunks created in a table

  Information about archived chunks created in a table is to be collected.

If `archivechunk` is specified, the table name of the archivable multi-chunk table must be specified in the `-n` option.

- `-n` {*DB-area-name* | *table-name* | *index-name*}

  ~<character string>

  Specifies the name of a DB area, table, or index when you want to collect information about a specific DB area, table, or index.

  DB-area-name

  Specifies the name of a DB area.

  This option can be specified when either the `-c` option is omitted or `dbarea` is specified in the `-c` option. The work table DB area name (`ADBWRK`) and the master directory DB area name (`ADBMST`) cannot be specified.

  table-name

  Specifies a table name in the format *schema-name*`.`*table-identifier*. The schema name cannot be omitted. Specifying a viewed table results in an error.

  This option can be specified when `table` is specified in the `-c` option.

  In addition, if `archivechunk` is specified in the `-c` option, the table name of the archivable multi-chunk table must be specified.

  To specify the table name of a dictionary table (base table) or system table (base table), specify `HADB` as the schema name.

  For details about the specification rules that apply when a schema name or table identifier includes any lowercase letter or backslash (\), see 1.4.3 Table name specification rules.

  index-name

  Specifies an index name in the format *schema-name*`.`*index-identifier*. The schema name cannot be omitted.

  This option can be specified when `index` is specified in the `-c` option.

  To collect information about the B-tree index corresponding to the primary key, you must specify the index identifier (same as the constraint name) that was generated automatically. For details about how to check the index identifier of the B-tree index corresponding to the primary key, see *Searching a dictionary table* in the *HADB Setup and Operation Guide*.

  To specify an index defined in a dictionary table (base table) and system table (base table), specify `HADB` as the schema name.

  The following rules apply to specification of DB area names and index names:

  - The specified character string is treated as being in all uppercase letters.
  - If the character string to be specified as the name includes a lowercase letter, enclose the entire name in double quotation marks (`"`), and then enclose it in single quotation marks (`'`). If the name is not enclosed in double quotation marks (`"`) and then within single quotation marks (`'`), it is treated as being in all uppercase letters.

    Example: `-n '"ADBuser01"'.'"Index01"'`

  For details about the naming rules for DB areas, tables, and indexes, see *Specifying names* in *SQL writing conventions* in *SQL Basics* in the manual *HADB SQL Reference*.

- `-t`

  Specifies that title lines (column headers) are not to be output.

- `-k` *enclosing-character*

  ~<character string>((1 byte))<<">>

  Specifies an enclosing character to be used for the character strings in CSV format. If this option is omitted, the double quotation mark (`"`) is assumed to be the enclosing character.

A single-byte character must be specified as the enclosing character (a character string consisting of two or more bytes cannot be specified). None of the following characters can be specified as the enclosing character:

- Space, tab, or asterisk (`*`)
- The character that is used as the delimiter (the value of or default value for the `-s` option)

> **Important**
>
> - If you specify a single quotation mark (`'`) as the enclosing character, enclose each in double quotation marks (`"`).
>
>   Example: If you specify a single quotation mark (`'`) as the enclosing character:
>
>   ```
>   adbdbstatus ... -k "'" ...
>   ```
>
> - If you specify as the enclosing character a special character used by the shell, such as the vertical bar (`|`), enclose each in either double quotation marks (`"`) or single quotation marks (`'`).
>
>   Example: If you specify the vertical bar (`|`) as the enclosing character:
>
>   ```
>   adbdbstatus ... -k "|" ...
>   ```
>
> - Do not use as the enclosing character any character that is used in the column titles and contents output by the `adbdbstatus` command. For details about the columns that are output by the `adbdbstatus` command, see the following:
>
>   - 13.3.2 List of items that are output as summary information for DB areas
>   - 13.4.2 List of items that are output as summary information for tables
>   - 13.5.2 List of items that are output as summary information for indexes
>   - 13.6.2 List of items that are output in the summary information of archived chunks
>   - 13.7.2 List of items that are output as usage information for DB areas, tables, and indexes
>   - 13.8.2 List of items that are output in the usage information of archived chunks

- `-s {`*delimiter*`|tab}`

  ~<character string>((1 byte))<<`,`>>

  Specifies the delimiter used in CSV format. To specify the tab character as the delimiter, specify `tab`. If this option is omitted, the comma (`,`) is assumed to be the delimiter.

  Only `tab` or a single-byte character can be specified as the delimiter (except for `tab`, a character string consisting of two or more bytes cannot be specified). None of the following characters can be specified as the delimiter:

  - Uppercase letters (`A` to `Z`), lowercase letters (`a` to `z`), numeric characters (`0` to `9`), underscore (_), double quotation mark (`"`), or asterisk (`*`)
  - The character that is used as the enclosing character (the value of or default value for the `-k` option)

> **Important**
>
> - If you specify a single quotation mark (`'`) as the delimiter, enclose it in double quotation marks (`"`).
>
>   Example: If you specify a single quotation mark (`'`) as the delimiter:
>
>   ```
>   adbdbstatus ... -s "'" ...
>   ```
>
> - If you specify as the delimiter a special character used by the shell, such as the vertical bar (`|`), enclose it in either double quotation marks (`"`) or single quotation marks (`'`).
>
>   Example: If you specify the vertical bar (`|`) as the delimiter:
>
>   ```
>   adbdbstatus ... -s "|" ...
>   ```

- Do not use as the delimiter any character that is used in the column titles and contents output by the `adbdbstatus` command. For details about the columns that are output by the `adbdbstatus` command, see the following:
  - 13.3.2 List of items that are output as summary information for DB areas
  - 13.4.2 List of items that are output as summary information for tables
  - 13.5.2 List of items that are output as summary information for indexes
  - 13.6.2 List of items that are output in the summary information of archived chunks
  - 13.7.2 List of items that are output as usage information for DB areas, tables, and indexes
  - 13.8.2 List of items that are output in the usage information of archived chunks

- `-S {K|M|G}`

Specifies that the output items for usage amounts are to be displayed in kilobytes, megabytes, or gigabytes, rather than in segments or pages.

When this option is omitted, the output items for usage amounts are displayed in segments or pages.

`K`: Kilobytes

Specifies that the output items for usage amounts are to be displayed in kilobytes.

`M`: Megabytes

Specifies that the output items for usage amounts are to be displayed in megabytes.

`G`: Gigabytes

Specifies that the output items for usage amounts are to be displayed in gigabytes.

When this option is specified, the column titles for the corresponding output items are also changed to match the units specified for the display information. The following table shows the relationships among the value of this option, the display units for output items, and the column titles.

Table 13-2: Display units and column titles for usage amount output items

No.	Information type	Output item	Value of this option	Display units for output items	Column title
1	Summary information for DB areas	Number of segments in use in the DB area	--	Segments	`Used_segments`
2			`K`	Kilobytes	`KB_Used_segments`
3			`M`	Megabytes	`MB_Used_segments`
4			`G`	Gigabytes	`GB_Used_segments`
5		Total number of segments in the DB area	--	Segments	`Total_segments`
6			`K`	Kilobytes	`KB_Total_segments`
7			`M`	Megabytes	`MB_Total_segments`
8			`G`	Gigabytes	`GB_Total_segments`
9	Summary information for tables	Number of segments in use by the table	--	Segments	`Used_segments`
10			`K`	Kilobytes	`KB_Used_segments`
11			`M`	Megabytes	`MB_Used_segments`
12			`G`	Gigabytes	`GB_Used_segments`
13		Number of pages in use by the table	--	Pages	`Used_pages`
14			`K`	Kilobytes	`KB_Used_pages`

No.	Information type	Output item	Value of this option	Display units for output items	Column title
15			`M`	Megabytes	`MB_Used_pages`
16			`G`	Gigabytes	`GB_Used_pages`
17	Summary information for indexes	Number of segments in use by the index	--	Segments	`Used_segments`
18			`K`	Kilobytes	`KB_Used_segments`
19			`M`	Megabytes	`MB_Used_segments`
20			`G`	Gigabytes	`GB_Used_segments`
21		Number of pages in use by the index	--	Pages	`Used_pages`
22			`K`	Kilobytes	`KB_Used_pages`
23			`M`	Megabytes	`MB_Used_pages`
24			`G`	Gigabytes	`GB_Used_pages`
25	Usage information for DB areas, tables, and indexes	Number of segments in use	--	Segments	`Used_segments`
26			`K`	Kilobytes	`KB_Used_segments`
27			`M`	Megabytes	`MB_Used_segments`
28			`G`	Gigabytes	`GB_Used_segments`
29		Number of pages in use	--	Pages	`Used_pages`
30			`K`	Kilobytes	`KB_Used_pages`
31			`M`	Megabytes	`MB_Used_pages`
32			`G`	Gigabytes	`GB_Used_pages`
33		Total number of segments in the DB area file	--	Segments	`Total_segments`
34			`K`	Kilobytes	`KB_Total_segments`
35			`M`	Megabytes	`MB_Total_segments`
36			`G`	Gigabytes	`GB_Total_segments`
37		Number of initialized pages in the DB area file	--	Pages	`Initialized_pages`
38			`K`	Kilobytes	`KB_Initialized_pages`
39			`M`	Megabytes	`MB_Initialized_pages`
40			`G`	Gigabytes	`GB_Initialized_pages`

Legend:

--: This option is omitted.

- `-r` *range-specification*

  If you want to output information about archived chunks, specify the range of chunks whose information you want to output. For details about the `-r` option, see (4) Explanation of the -r option (range specification).

- `--shared-lock`

  Specifies that the lock of the `adbdbstatus` command is to be changed. Specify this option if you want to execute the `adbdbstatus` command concurrently with the following commands and SQL statements:

- adbimport command (when background import is performed)
- adbmergechunk command
- Update SQL statements (INSERT statement, DELETE statement, and UPDATE statement)

For details about the lock when the --shared-lock option is specified, see *Locked resources that are reserved and their lock modes* in the *HADB Setup and Operation Guide*.

If the adbdbstatus command with the --shared-lock option specified is executed, some output items cannot be acquired. Also, if the adbdbstatus command with the --shared-lock option specified is executed concurrently with a transaction that updates the database, values of some output items might be inconsistent. For details, see the following.

Important

For details about the output items when the --shared-lock option is specified, see 13.4.2 List of items that are output as summary information for tables, 13.5.2 List of items that are output as summary information for indexes, and 13.7.2 List of items that are output as usage information for DB areas, tables, and indexes.

■ Output items that cannot be acquired if the --shared-lock option is specified

If the --shared-lock option is specified, the following output items cannot be acquired. In this case, no_check is always displayed.

- Summary information for tables
  Whether the table is non-updatable (Non-updatable), and command termination status (Command_status)
- Summary information for indexes
  Uniqueness constraint violation (Unique_constraint_violated), and whether the index is in unfinished status (Unfinished)

In addition, if the --shared-lock option is specified, information about deletion-pending chunks corresponding to the following tables or indexes is not output:

- System table (base table)
- Index defined for a system table (base table)
- Table for which the adbmergechunk command is being executed
- Index defined for a table for which the adbmergechunk command is being executed

■ Restrictions on output items when the --shared-lock option is specified

The adbdbstatus command with the --shared-lock option specified can be executed for a table for which one of the following commands was executed. In this case, the latest value might not be displayed for items relating to the number of segments, pages, or chunks. (The values that existed when the relevant command started might be displayed.) Therefore, a displayed value might be inconsistent with the values of other output items.

- adbimport command (when background import is performed)
- adbidxrebuild command (executed with the --create-temp-file option specified after background import is canceled)

- adbmergechunk command
- adbreorgsystemdata command

Note that if the --shared-lock option is specified when the following information is to be output, the adbdbstatus command causes an error:

- Summary information for DB areas
  If summary is specified for the -d option (or the -d option is omitted) and dbarea is specified for the -c option (or the -c option is omitted)
- Summary information of archived chunks
  If summary is specified for the -d option (or the -d option is omitted) and archivechunk is specified for the -c option
- Usage information of archived chunks
  If used is specified for the -d option and archivechunk is specified for the -c option

(4) Explanation of the -r option (range specification)

When you want to output information about archived chunks, use the -r option to specify the range of chunks whose information is to be output. If the specified range contains any part of the range of values in the archive range column (the column used to narrow down the search range when searching the archivable multi-chunk table), the corresponding chunks' information is output. If the corresponding chunk does not exist, no information is output.

The range specification is interpreted based on the data type in the archive range column.

The following table lists the types of range specifications and links to their detailed explanations.

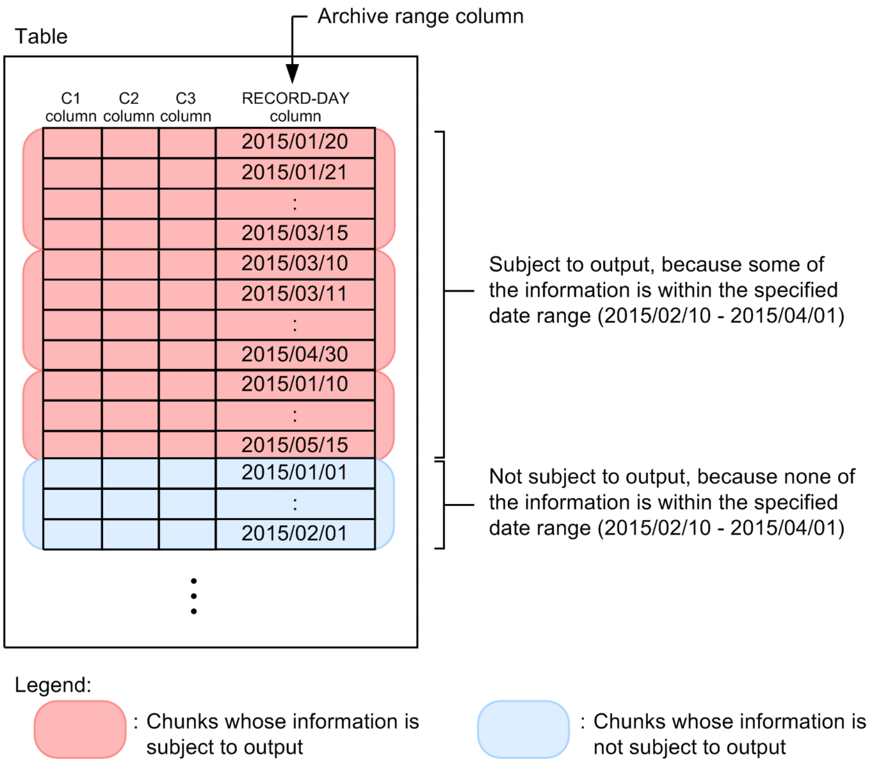
Table 13-3: Types of range specifications and their links

No.	Type of range specification	Link
1	Date range specification	(a) Specifying ranges of dates
2	Time range specification	(b) Specifying ranges of times
3	Time stamp range specification	(c) Specifying ranges of time stamps
4	Fixed-length character string range specification	(d) Specifying ranges of fixed-length character strings
5	Decimal range specification	(e) Specifying ranges of decimal numbers
6	Integer range specification	(f) Specifying ranges of integers
7	Floating point range specification	(g) Specifying ranges of floating point numbers

This section shows which chunks' information is to be output, using the following conditions as an example:

- The data type in the archive range column is DATE.
- The date range 2015/02/10 - 2015/04/01 is specified in the range specification.

Figure 13-2: Relationship between the content of the date range specification and the chunks whose information is to be output


Archive range column
Table
C1 column
C2 column
C3 column
RECORD-DAY column
2015/01/20
2015/01/21
:
2015/03/15
2015/03/10
2015/03/11
:
2015/04/30
2015/01/10
:
2015/05/15
2015/01/01
:
2015/02/01
Subject to output, because some of the information is within the specified date range (2015/02/10 - 2015/04/01)
Not subject to output, because none of the information is within the specified date range (2015/02/10 - 2015/04/01)
Legend:
: Chunks whose information is subject to output
: Chunks whose information is not subject to output

The rules for specifying the `-r` option are as follows:

- When this option is specified, `archivechunk` must be specified in the `-c` option. If `archivechunk` is not specified in the `-c` option, `KFAA90002-E` message is output. In addition, the table name of the archivable multi-chunk table must be specified in the `-n` option. If the specified table name is not the table name of the archivable multi-chunk table, the `KFAA50280-E` message is output.
- You can only specify one range. Multiple ranges cannot be specified.
- If 1,024 bytes or more are specified in the range specification, the `KFAA90003-E` message is output.
- In this option, specify the value that results after characters such as shell enclosing characters have been parsed. The specified value and the actual interpreted value might differ depending on how the shell parses the characters. The following is an example of specifying a range of fixed-length character strings.

  Option specification value: `-r 'abc'-'def'`

  Actual interpreted value: `-r abc-def`

  Single quotation marks (`'`) have been omitted from the actual interpreted value.

(a) Specifying ranges of dates

Specify a range of dates for the chunks whose information you want to output.

Format

```
-r {YYYY-MM-DD|YYYY/MM/DD}-{YYYY-MM-DD|YYYY/MM/DD}
```

Connect the minimum value (date specification) and maximum value (date specification) by using a hyphen (-). There must be no spaces between the minimum value and the hyphen (-), or between the maximum value and the hyphen (-).

Explanation

YYYY: Year (`0001-9999`)

MM: Month (`01-12`)

DD: Day (`01` to the last day of the month specified in *MM*)

Specification example

Specification example 1: `-r 2016/01/01-2016/12/31`

Specification example 2: `-r 2016-01-01-2016-12-31`

Specify each date in either of the following formats: *YYYY*`-`*MM*`-`*DD* or *YYYY*`/`*MM*`/`*DD* (delimit *YYYY*, *MM*, and *DD* using forward slashes (`/`) or hyphens (`-`)).

In the case of specification example 1, `2016/01/01` and `2016/12/31` are treated as date specifications.

In the case of specification example 2, `2016-01-01` and `2016-12-31` are treated as date specifications.

The rules for specifying ranges of dates are as follows:

- If there are not enough digits in the values you want to specify in *YYYY*, *MM*, or *DD*, pad the left side of the value with zeros (`0`).
- The data types in the archive range column for which you can specify a date range are as follows:
  - `DATE`
  - `TIMESTAMP`
- Only dates can be specified in a date range specification. If the data type in the archive range column is `TIMESTAMP`, specified values are treated as if the time has been omitted. Therefore, the start time and end time in a specified value are corrected as shown in the following table. As a result, all chunks containing data in the specified date range are subject to information output.

Table 13-4: Correction of the start time and end time

No.	Number of digits in the fractional seconds defined in the `TIMESTAMP`-type column	Start time	End time
1	`0`	0 hours 0 minutes 0 seconds	23 hours 59 minutes 59 seconds
2	`3`	0 hours 0 minutes 0.000 seconds	23 hours 59 minutes 59.999 seconds
3	`6`	0 hours 0 minutes 0.000000 seconds	23 hours 59 minutes 59.999999 seconds
4	`9`	0 hours 0 minutes 0.000000000 seconds	23 hours 59 minutes 59.999999999 seconds
5	`12`	0 hours 0 minutes 0.000000000000 seconds	23 hours 59 minutes 59.999999999999 seconds

- The following table lists the message IDs that are displayed when the specified date range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then execute the `adbdbstatus` command.

Table 13-5: Message IDs that are displayed when the specified date range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	`KFAA50225-E`	• Too many digits in the date • Not enough digits in the date	`-r 2015/7/1-2015/008/031`	`-r 2015/07/01-2015/08/31`

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
2		A date outside the range is specified.	`-r 2015/01/01-2015/06/31`	`-r 2015/01/01-2015/06/30`
3	• `KFAA90002-E` • `KFAA96720-I`	The range specification format is invalid.	**Specification example** `-r 2015/01/11Δ-Δ2015/7/31` **Explanation** Because there is a space between the date specification and the hyphen (`-`), the shell erroneously uses spaces to delimit the option arguments.	`-r 2015/01/11-2015/07/31`

Legend:

Δ: Space

(b) Specifying ranges of times

Specify a range of times for the chunks whose information you want to output.

Format

```
-r hh:mm:ss[.[nn...n]]-hh:mm:ss[.[nn...n]]
```

Connect the minimum value (time specification) and maximum value (time specification) by using a hyphen (`-`). There must be no spaces between the minimum value and the hyphen (`-`), or between the maximum value and the hyphen (`-`).

Explanation

hh:: Hours (`00`-`23`)

mm: Minutes (`00`-`59`)

ss: Seconds (`00`-`59`)

`.`: Decimal point

nn...n: Fractional seconds (*n* is `0`-`9`)

Specification example

Specification example 1: `-r 00:00:00-12:59:59`

Specification example 2: `-r 00:00:00.000-12:59:59.999`

The *hh*, *mm*, and *ss* values of each time are delimited by using colons (`:`).

When specifying fractional seconds, connect the *ss* and *nn...n* values by using a decimal point (`.`).

In the case of specification example 1, `00:00:00` and `12:59:59` are treated as time specifications.

In the case of specification example 2, `00:00:00.000` and `12:59:59.999` are treated as time specifications.

The rules for specifying ranges of times are as follows:

- If there are not enough digits in the values you want to specify in *hh*, *mm*, or *ss*, pad the left side of the value with zeros (`0`).
- The data type in the archive range column for which you can specify a time range is as follows:
  - `TIME`

- If the fractional seconds have more digits than the fractional seconds defined in the `TIME`-type column, the excess digits are truncated.
- If fractional seconds are omitted, the start time and end time are corrected to reflect the number of digits in the fractional seconds defined in the `TIME`-type column. The following table lists details about how this correction is made.

Table 13-6: Correction of the start time and end time when fractional seconds are omitted

No.	Number of digits in the fractional seconds defined in the TIME-type column	Start time	End time
1	`3`	.000 seconds	.999 seconds
2	`6`	.000000 seconds	.999999 seconds
3	`9`	.000000000 seconds	.999999999 seconds
4	`12`	.000000000000 seconds	.999999999999 seconds

- For details about comparing the minimum and maximum values for time specifications, see *Comparing datetime data* in *Data types that can be compared* in *Data types that can be converted, assigned, and compared* in the manual *HADB SQL Reference*.
- The following table lists the message IDs that are displayed when the specified time range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then execute the `adbdbstatus` command.

Table 13-7: Message IDs that are displayed when the specified time range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	`KFAA50225-E`	• Too many digits in the time • Not enough digits in the time	`-r 0:0:0-012:59:59`	`-r 00:00:00-12:59:59`
2		A time outside the range is specified.	`-r 00:00:00-12:60:00`	`-r 00:00:00-12:59:59`
3	• `KFAA90002-E` • `KFAA96720-I`	The range specification format is invalid.	**Specification example** `-r 00:00:00Δ-Δ12:59:59` **Explanation** Because there is a space between the time specification and the hyphen (`-`), the shell erroneously uses spaces to delimit the option arguments.	`-r 00:00:00-12:59:59`

Legend:

Δ: Space

(c) Specifying ranges of time stamps

Specify a range of time stamps for the chunks whose information you want to output.

Format

```
-r '{YYYY-MM-DD|YYYY/MM/DD}Δhh:mm:ss[.[nn...n]]'
    -'{YYYY-MM-DD|YYYY/MM/DD}Δhh:mm:ss[.[nn...n]]'
```

Connect the minimum value (time stamp specification) and maximum value (time stamp specification) by using a hyphen (`-`). There must be no spaces between the minimum value and the hyphen (`-`), or between the maximum value and the hyphen (`-`).

Explanation

YYYY: Year (`0001`-`9999`)

MM: Month (`01`-`12`)

DD: Day (`01` to the last day of the month specified in *MM*)

Δ: Halfwidth space or tab

hh:: Hours (`00`-`23`)

mm: Minutes (`00`-`59`)

ss: Seconds (`00`-`59`)

`.`: Decimal point

nn...n: Fractional seconds (*n* is `0`-`9`)

Specification example

Specification example 1: `-r '2016/01/01 00:00:00'-'2016/12/31 12:59:59'`

Specification example 2: `-r '2016/01/01 00:00:00.000'-'2016/12/31 12:59:59.999'`

Connect the date and time of each specified time stamp by using a halfwidth space, and enclose entire date or time values in single quotation marks (`'`). When specifying dates, use either of the following formats: *YYYY*`-`*MM*`-`*DD* or *YYYY*`/`*MM*`/`*DD* (delimit *YYYY*, *MM*, and *DD* using forward slashes (`/`) or hyphens (`-`)). When specifying times, delimit the *hh*, *mm*, and *ss* values by using colons (`:`). When specifying fractional seconds, connect the *ss* and *nn...n* values by using a decimal point (`.`).

In the case of specification example 1, `2016/01/01 00:00:00` and `2016/12/31 12:59:59` are treated as time stamp specifications.

In the case of specification example 2, `2016/01/01 00:00:00.000` and `2016/12/31 12:59:59.999` are treated as time stamp specifications.

The rules for specifying ranges of time stamps are as follows:

- If there are not enough digits in the values you want to specify in *YYYY*, *MM*, *DD*, *hh*, *mm*, or *ss*, pad the left side of the value with zeros (`0`).
- The data type in the archive range column for which you can specify a time stamp range is as follows:
  - `TIMESTAMP`
- If the fractional seconds have more digits than the fractional seconds defined in the `TIMESTAMP`-type column, the excess digits are truncated.
- If fractional seconds are omitted, the start time and end time are corrected to reflect the number of digits in the fractional seconds defined in the `TIMESTAMP`-type column. The following table lists details about how this correction is made.

Table 13-8: Correction of the start time and end time when fractional seconds are omitted

No.	Number of digits in the fractional seconds defined in the TIMESTAMP-type column	Start time	End time
1	3	.000 seconds	.999 seconds
2	6	.000000 seconds	.999999 seconds
3	9	.000000000 seconds	.999999999 seconds

No.	Number of digits in the fractional seconds defined in the TIMESTAMP-type column	Start time	End time
4	12	.000000000000 seconds	.999999999999 seconds

- For details about comparing the minimum and maximum values for time stamp specifications, see *Comparing datetime data* in *Data types that can be compared* in *Data types that can be converted, assigned, and compared* in the manual *HADB SQL Reference*.
- The following table lists the message IDs that are displayed when the specified time stamp range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then execute the `adbdbstatus` command.

Table 13-9: Message IDs that are displayed when the specified time stamp range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	KFAA50225-E	• Too many digits in the time stamp • Not enough digits in the time stamp	-r '2015/7/1Δ00:00:00'-'2015/008/031Δ12:59:59'	-r '2015/07/01Δ00:00:00'-'2015/08/31Δ12:59:59'
2		A time stamp outside the range is specified.	-r '2015/01/01Δ00:00:00'-'2015/06/31Δ12:60:00'	-r '2015/01/01Δ00:00:00'-'2015/06/30Δ12:59:59'
3	• KFAA90002-E • KFAA96720-I	The range specification format is invalid.	**Specification example** -r '2016/04/01Δ00:00:00'Δ-Δ'2016/06/30Δ12:59:59' **Explanation** Because there is a space between the time stamp specification and the hyphen (-), the shell erroneously uses spaces to delimit the option arguments.	-r '2016/04/01Δ00:00:00'-'2016/06/30Δ12:59:59'
4			**Specification example** -r 2016/04/01Δ00:00:00-2016/06/30Δ12:59:59 **Explanation** Because the time stamp specification is not enclosed in single quotation marks ('), the shell sees the space between the date and time values, and erroneously uses spaces to delimit the option arguments.	

Legend:

Δ: Space

(d) Specifying ranges of fixed-length character strings

Specify a range of fixed-length character strings for the chunks whose information you want to output.

Format

```
-r a...a-a...a
```

Connect the minimum value (fixed-length character string specification) and maximum value (fixed-length character string specification) by using a hyphen (`-`). There must be no spaces between the minimum value and the hyphen (`-`), or between the maximum value and the hyphen (`-`).

Explanation

a...a: String of one or more characters

Specification example

`-r 20160101-20161231`

`20160101` and `20161231` are treated as fixed-length character string specifications.

The rules for specifying ranges of fixed-length character strings are as follows:

- Specify a fixed-length character string of one or more characters.
- Fixed-length character string specifications are case sensitive.
- The data type in the archive range column for which you can specify a fixed-length character string range is as follows:
  - `CHARACTER`
- If the fixed-length character string includes a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), specify the escape character (`\`) immediately before that character. Alternatively, enclose the entire fixed-length character string in single quotation marks (`'`).

  Specification example

  Specification example 1: `-r a\|c-def`

  Specification example 2: `-r 'a|c'-def`

  In both examples, `a|c` and `def` are treated as fixed-length character string specifications.
- If a character string is enclosed in double quotation marks (`"`) and then the entire string is enclosed in single quotation marks (`'`), both the character string and the double quotation marks (`"`) are treated as a fixed-length character string.

  Specification example

  `-r '"abc"'-def`

  `"abc"` and `def` are treated as fixed-length character string specifications.
- If you specify a fixed-length character string that includes any hyphens (`-`), enclose the entire fixed-length character string in single quotation marks (`'`). This is to differentiate hyphens (`-`) in the specification from hyphens used to separate the minimum and maximum values.

  Specification example

  - When enclosing a specification in `\'` (backslashes and single quotation marks)

    `-r \'a-c\'-def`
  - When first enclosing a specification in single quotation marks (`'`), and then enclosing the entire string in double quotation marks (`"`)

    `-r "'a-c'"-def`

  In both examples, `a-c` and `def` are treated as fixed-length character string specifications.

  In addition, even if a specified fixed-length character string does not include any hyphens (`-`), if you enclose the specification in single quotation marks (`'`), the enclosed value is treated as the fixed-length character string specification, with the single quotation marks (`'`) omitted.

Specification example

- When enclosing a specification in `\'` (backslashes and single quotation marks)
  `-r \'abc\'-def`
  `abc` and `def` are treated as fixed-length character string specifications.

- If a specified fixed-length character string includes any single quotation marks (`'`), specify two single quotation marks (`'`) in succession to express one single quotation mark (`'`).

  Specification example

  - When specifying two `\'` (backslashes and single quotation marks)
    `-r a\'\'c-def`
  - When enclosing a specification that includes a single quotation mark (`'`), in double quotation marks (`"`)
    `-r "a''c"-def`

  In both examples, `a'c` and `def` are treated as fixed-length character string specifications.

- For details about comparing the minimum and maximum values for fixed-length character string specifications, see *Comparing character string data* in *Data types that can be compared* in *Data types that can be converted, assigned, and compared* in the manual *HADB SQL Reference*.

- The following table lists the message IDs that are displayed when the specified fixed-length character string range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then execute the `adbdbstatus` command.

Table 13-10: Message IDs that are displayed when the specified fixed-length character string range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification

1	`KFAA50225-E`	The fixed-length character string specification includes multiple hyphens (-) that are used to separate the minimum and maximum values.	`-r a-c-def`
			`-r 'a-c-def'`
				`-r \'a-c\'-def`

2	• `KFAA90002-E` • `KFAA96720-I`	The range specification format is invalid.	**Specification example** `-r abcΔ-Δdef` **Explanation** Because there is a space between the fixed-length character string specification and the hyphen (-), the shell erroneously uses spaces to delimit the option arguments.	`-r abc-def`

Legend:

Δ: Space

(e) Specifying ranges of decimal numbers

Specify a range of decimal numbers for the chunks whose information you want to output.

Format

```
-r [{±|-}]{a...a[.[b...b]]|.b...b}
    -[{±|-}]{a...a[.[b...b]]|.b...b}
```

Connect the minimum value (decimal specification) and maximum value (decimal specification) by using a hyphen (`-`). There must be no spaces between the minimum value and the hyphen (`-`), or between the maximum value and the hyphen (`-`).

Explanation

`+`, `-`: Signs (the "+" sign can be omitted)

a...a: Integer part (*a* is `0`-`9`)

`.`: Decimal point

b...b: Fractional part (*b* is `0`-`9`)

Specification examples in the case of DECIMAL(5,2)

Specification example 1: `-r -100-100`

Specification example 2: `-r -100.00-+100.00`

`-100.00` and `+100.00` are treated as decimal specifications.

The rules for specifying ranges of decimal numbers are as follows:

- Either the integer part or the fractional part must be specified. If there is no fractional part, the decimal point can be omitted.
- The data type in the archive range column for which you can specify a decimal range is as follows:
  - `DECIMAL`
- If the fractional parts have more digits than the fractional parts defined in the `DECIMAL`-type column, the excess digits are truncated.
- The following table lists the message IDs that are displayed when the specified decimal range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then execute the `adbdbstatus` command.

Table 13-11: Message IDs that are displayed when the specified decimal range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	`KFAA50225-E`	A numerical value has been specified that cannot be used with the data type in the archive range column.	**Specification example** `-r 0-123456789012345678901234567890123456789` **Explanation** A decimal specification contains 39 digits.	**Specification example** `-r 0-12345678901234567890123456789012345678` **Explanation** Specify a decimal using 38 digits.
2	• `KFAA90002-E` • `KFAA96720-I`	The range specification format is invalid.	**Specification example** `-r 0Δ-Δ1` **Explanation** Because there is a space between the decimal specification and the hyphen (`-`), the shell erroneously uses spaces to delimit the option arguments.	`-r 0-1`

Legend:

Δ: Space

(f) Specifying ranges of integers

Specify a range of integers for the chunks whose information you want to output.

Format

```
-r [{+|-}]a...a-[{+|-}]a...a
```

Connect the minimum value (integer specification) and maximum value (integer specification) by using a hyphen (-). There must be no spaces between the minimum value and the hyphen (-), or between the maximum value and the hyphen (-).

Explanation

+, -: Signs (the "+" sign can be omitted)

a...a: Numerical value (*a* is 0-9)

Specification example

Specification example 1: `-r -100-100`

Specification example 2: `-r -100-+100`

`-100` and `+100` are treated as integer specifications.

The rules for specifying ranges of integers are as follows:

- The data types in the archive range column for which you can specify an integer range are as follows:
  - `INTEGER`
  - `SMALLINT`
- The following table lists the message IDs that are displayed when the specified integer range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then execute the `adbdbstatus` command.

Table 13-12: Message IDs that are displayed when the specified integer range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	KFAA50225-E	A numerical value has been specified that cannot be used with the data type in the archive range column.	**In the case of INTEGER** -r 0-9223372036854775808 **In the case of SMALLINT** -r 0-2147483648	**In the case of INTEGER** -r 0-9223372036854775807 **In the case of SMALLINT** -r 0-2147483647
2	• KFAA90002-E • KFAA96720-I	The range specification format is invalid.	**Specification example** -r 0Δ-Δ1 **Explanation** Because there is a space between the integer specification and the hyphen (-), the shell erroneously uses spaces to delimit the option arguments.	-r 0-1

Legend:

Δ: Space

(g) Specifying ranges of floating point numbers

Specify a range of floating point numbers for the chunks whose information you want to output.

Format

```
-r [{+|-}]{a...a[.[b...b]]|.b...b}[{E|e}[{+|-}]c...c]
    - [{+|-}]{a...a[.[b...b]]|.b...b}[{E|e}[{+|-}]c...c]
```

Connect the minimum value (floating point specification) and maximum value (floating point specification) by using a hyphen (-). There must be no spaces between the minimum value and the hyphen (-), or between the maximum value and the hyphen (-).

Explanation

+, -: Signs (the "+" sign can be omitted)

a...a: Integer part of the mantissa (*a* is 0-9)

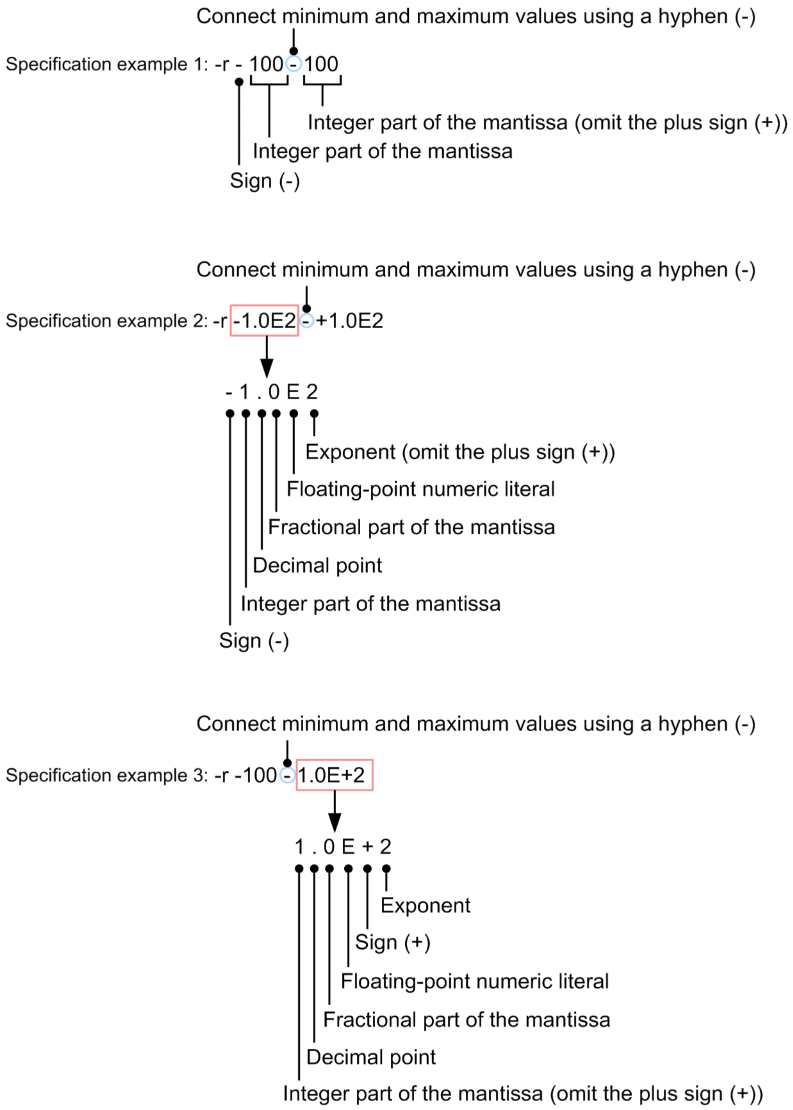
.: Decimal point

b...b: Fractional part of the mantissa (*b* is 0-9)

E, e: Floating-point numeric literal

c...c: Exponent (*c* is 0-9)

Figure 13-3: Example of specifying a floating point range by using the -r option


Connect minimum and maximum values using a hyphen (-)
Specification example 1: -r - 100-100
Integer part of the mantissa (omit the plus sign (+))
Integer part of the mantissa
Sign (-)
Connect minimum and maximum values using a hyphen (-)
Specification example 2: -r -1.0E2-+1.0E2
- 1 . 0 E 2
Exponent (omit the plus sign (+))
Floating-point numeric literal
Fractional part of the mantissa
Decimal point
Integer part of the mantissa
Sign (-)
Connect minimum and maximum values using a hyphen (-)
Specification example 3: -r -100-1.0E+2
1 . 0 E + 2
Exponent
Sign (+)
Floating-point numeric literal
Fractional part of the mantissa
Decimal point
Integer part of the mantissa (omit the plus sign (+))

In the case of specification example 1, -100 and 100 are treated as floating point specifications.

In the case of specification example 2, -1.0E2 and +1.0E2 are treated as floating point specifications.

In the case of specification example 3, `-100` and `1.0E+2` are treated as floating point specifications.

The rules for specifying ranges of floating point numbers are as follows:

- The data type in the archive range column for which you can specify a floating point range is as follows:
  - `DOUBLE PRECISION`
- The following table lists the message IDs that are displayed when the specified floating point range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then execute the `adbdbstatus` command.

Table 13-13: Message IDs that are displayed when the specified floating point range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	`KFAA50225-E`	The number of specifiable characters is exceeded.	**Specification example** `-r 0-123...(omitted)...789` **Explanation** A floating point specification contains 510 characters.	**Specification example** `-r 0-123...(omitted)...78` **Explanation** Specify a floating point specification using 509 characters.
2	• `KFAA90002-E` • `KFAA96720-I`	The range specification format is invalid.	**Specification example** `-r 0Δ-Δ1` **Explanation** Because there is a space between the floating point specification and the hyphen (-), the shell erroneously uses spaces to delimit the option arguments.	`-r 0-1`

Legend:

Δ: Space

13.2.2 Explanation of the specification format and options (outputting information about the need for reorganization of column store tables)

(1) Specification format

The following indicates the specification format of the `adbdbstatus` command for outputting *information about the need for reorganization* of column store tables.

For details about the specification format to output *summary information* or *usage information*, see (1) Specification format in 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).

```
adbdbstatus
    -d reorginfo
    -n table-name
  [-c chunk-ID-list-to-be-analyzed]
  [-t]
  [-k enclosing-character]
```

```
[-s {delimiter|tab}]
[-S {K|M|G}]
```

(2) Explanation of options

The following describes the options of the `adbdbstatus` command used to output *information about the need for reorganization* of column store tables. For details about the options used to output *summary information* or *usage information*, see (3) Explanation of options in 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).

- `-d reorginfo`

  This option must be specified.

  Specify `reorginfo` as the type of output results to output information about the need for reorganization.

- `-n` *table-name*

  ~<character string>

  This option must be specified.

  Specifies the table name of a column store table for which information is to be collected, in the format *schema-name*`.`*table-identifier*. The schema name cannot be omitted. Specifying a viewed table or row store table results in an error.

  For details about the specification rules that apply when a schema name or table identifier includes any lowercase letter or backslash (`\`), see 1.4.3 Table name specification rules.

- `-c` *chunk-ID-list-to-be-analyzed*

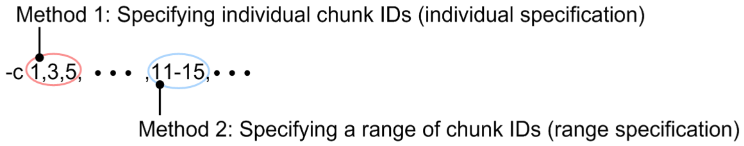
  ~<character string>((1 to 32,768 bytes))

  Specify this option if you want to collect information about specific chunks in the target table specified by the `-n` option.

  If this option is omitted, information about all chunks in the target table except for deletion-pending chunks is collected.

  You can specify *chunk-ID-list-to-be-analyzed* by using the following methods. Note that you can combine multiple methods of specifying chunk IDs.

  Figure 13-4: Example of a chunk ID list to be analyzed


Method 1: Specifying individual chunk IDs (individual specification)
-c 1,3,5, • • • ,11-15,• • •
Method 2: Specifying a range of chunk IDs (range specification)

  - **Specifying individual chunk IDs (individual specification)**

    If you want to specify individual chunk IDs for which information is to be collected, delimit the chunk IDs by using commas (`,`).

    Important

    There must be no spaces before or after a comma (`,`).

  - **Specifying a range of chunk IDs (range specification)**

    If you specify a range of chunk IDs for which information is to be collected, use the format *smallest-chunk-ID-in-the-range*`-`*largest-chunk-ID-in-the-range* (connect the two chunk IDs with a hyphen (`-`)).

Important

There must be no spaces before or after the hyphen (`-`). The chunk ID specified to the right of the hyphen must be greater than the chunk ID specified to the left of the hyphen.

Among the chunks whose IDs are specified, information for the following chunks will not be collected:

- Chunks that are not in the table to be processed
- Deletion-pending chunks

The rules for specifying this option are as follows:

- You can specify maximum of 30,000 chunk IDs. The following explains how chunk IDs are counted:

  • For a range specification, all chunk IDs in the specified range are counted.

  • If any duplicate chunk IDs are specified, the duplicate chunk IDs are counted.

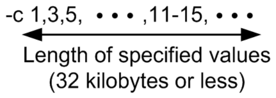
  Examples of chunk ID lists to be analyzed, and how to count chunk IDs

  Example 1: `-c 1,3,5`: It is assumed that three chunk IDs (`1`, `3`, and `5`) are specified .

  Example 2: `-c 1,3,5-8,10`: It is assumed that seven chunk IDs (`1`, `3`, `5`, `6`, `7`, `8`, and `10`) are specified.

  Example 3: `-c 1,3,5,5-8,10`: It is assumed that eight chunk IDs (`1`, `3`, `5`, `5`, `6`, `7`, `8`, and `10`) are specified.

- The maximum permitted length of the entire value specified in the `-c` option is 32 kilobytes. If the specified length exceeds 32 kilobytes, use range specifications instead of individual specifications, or execute the `adbdbstatus` command more than once.



- For details about specifications that result in an error and specification examples, see (3) Errors that can be caused by the -c option.

▪ **Searching for chunk IDs of chunks that are in a table to be processed**

Search system tables for `STATUS_CHUNKS`, and check the chunk IDs.

For details, see *Checking the information about all chunks in a table based on a table name* in *Searching system tables* in *System Tables* in the *HADB Setup and Operation Guide*.

- `-t`

  Specifies that title lines (column headers) are not to be output.

- `-k` *enclosing-character*

  ~<character string>((1 byte))<<">>

  Specifies an enclosing character to be used for the character strings in CSV format. If this option is omitted, the double quotation mark (`"`) is assumed to be the enclosing character.

  The enclosing character must be a single-byte character. A character string of 2 or more bytes cannot be specified. None of the following characters are permitted to be the enclosing character:

  - Space, tab, asterisk (`*`)
  - The character that is used as the delimiter (the value of or default value for the `-s` option)

Important

- If you specify a single quotation mark (`'`) as the enclosing character, enclose each in double quotation marks (`"`).

Example: If you specify a single quotation mark (') as the enclosing character:

```
adbdbstatus ... -k "'" ...
```

- If you specify as the enclosing character a special character used by the shell, such as the vertical bar (|), enclose each in either double quotation marks (") or single quotation marks (').

  Example: If you specify the vertical bar (|) as the enclosing character:

  ```
  adbdbstatus ... -k "|" ...
  ```

- For the enclosing character, do not use one that is already in any column titles or contents output in information about the need for reorganization. For details about the columns that are output in the information about the need for reorganization, see 13.9.2 List of items that are output as information about the need for reorganization.

- `-s {`*delimiter*`|tab}`

  ~<character string>((1 byte))<<,>>

  Specifies the delimiter used in CSV format. To specify the tab character as the delimiter, specify `tab`. If this option is omitted, the comma (,) is assumed to be the delimiter.

  Only the `tab` character or a single-byte character can be specified as a delimiter. Except for the `tab` character, a character string consisting of two or more bytes cannot be specified. None of the following characters can be specified as the delimiter:

  - Uppercase letters (A to Z), lowercase letters (a to z), numeric characters (0 to 9), underscore (_), double quotation mark ("), or asterisk (*)
  - The character that is used as the enclosing character (the value of or default value for the `-k` option)

Important

- If you specify a single quotation mark (') as the delimiter, enclose it in double quotation marks (").

  Example: If you specify a single quotation mark (') as the delimiter:

  ```
  adbdbstatus ... -s "'" ...
  ```

- If you specify as the delimiter a special character used by the shell, such as the vertical bar (|), enclose it in either double quotation marks (") or single quotation marks (').

  Example: If you specify the vertical bar (|) as the delimiter:

  ```
  adbdbstatus ... -s "|" ...
  ```

- For the delimiter, do not use one that is already in any column titles or contents output in information about the need for reorganization. For details about the columns that are output in the information about the need for reorganization, see 13.9.2 List of items that are output as information about the need for reorganization.

- `-S {K|M|G}`

  Specifies that output items for necessity of reorganization are to be displayed in kilobytes, megabytes, or gigabytes, rather than in pages.

  When this option is omitted, output items are displayed in pages.

  `K`: Kilobytes

  Specifies that the output items are to be displayed in kilobytes.

  `M`: Megabytes

  Specifies that the output items are to be displayed in megabytes.

G: Gigabytes

Specifies that the output items are to be displayed in gigabytes.

When this option is specified, the column titles for the corresponding output items are also changed to match the units specified for the display information. The following table shows the relationships among the value of this option, the display units for output items, and the column titles.

Table 13-14: Display units and column titles for the output items for the necessity of reorganization

No.	Output item	Value of this option	Display units for output items	Column title
1	Number of basic row pages contained in the row-data segment	--	Pages	`Base_row_pages`
2		K	Kilobytes	`KB_Base_row_pages`
3		M	Megabytes	`MB_Base_row_pages`
4		G	Gigabytes	`GB_Base_row_pages`
5	Number of branch row pages contained in the row-data segment	--	Pages	`Branch_row_pages`
6		K	Kilobytes	`KB_Branch_row_pages`
7		M	Megabytes	`MB_Branch_row_pages`
8		G	Gigabytes	`GB_Branch_row_pages`
9	Number of invalid row information pages contained in the row-data segment	--	Pages	`Invalid_row_information_pages`
10		K	Kilobytes	`KB_Invalid_row_information_pages`
11		M	Megabytes	`MB_Invalid_row_information_pages`
12		G	Gigabytes	`GB_Invalid_row_information_pages`

Legend:

--: This option is omitted.

(3) Errors that can be caused by the -c option

The following table shows cases where the `adbdbstatus` command fails due to the specification of the `-c` option.

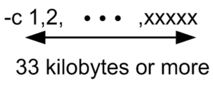
In any of these error cases, correct the specification of the `-c` option, and then re-execute the `adbdbstatus` command. For details about the `-c` option, see -c in (2) Explanation of options.

Table 13-15: Message ID displayed when the specification of the -c option is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	`KFAA50205-E`	The number of specified chunk IDs exceeds 30,000.	`-c 1-30001`	`-c 1-30000`
2		The same chunk ID is specified on both sides of a hyphen (-) in the range specification.	`-c 3-3,7`	`-c 3-5,7`

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
3		In the range specification, the chunk ID specified to the right of the hyphen (-) is smaller than the chunk ID specified to the left of the hyphen.	`-c 10-3`	`-c 3-10`

4		Any one of the following invalid values is specified for the chunk ID:
		• Values other than integers
		• Values outside the range from 1 to 9,223,372,036,854,775,807
			`-c A`
			`-c 0`
			`-c A-B`
			`-c 0-3`
				Specify a correct chunk ID.

5	• `KFAA90002-E` • `KFAA96720-I`	There is a space between a chunk ID and a comma.	`-c 1,3,Δ10`	`-c 1,3,10`
6		There is a space between a chunk ID and the hyphen (-) in the range specification.	`-c 1,3Δ-Δ10`	`-c 1,3-10`
7	`KFAA90003-E`	The length of the specified value exceeds 32 kilobytes.		• If individual specification is used, change it to range specification (use a hyphen (-), not a comma (`,`)) `-c 1-`*xxxxx* • Divide the specification of the `adbdbstatus` command, and then execute the command more than once.

Legend:

Δ: Space

13.3 Items that are output as summary information for DB areas

This section provides the details of the items that are output as summary information for DB areas.

To output summary information for DB areas, specify the following two options:

- `-d` option
  Specify `summary` in the `-d` option. Alternatively, omit the `-d` option.
- `-c` option
  Specify `dbarea` in the `-c` option. Alternatively, omit the `-c` option.

You can use the `-n` option to specify a specific DB area for which information is to be collected.

For details about the specification format and options of the `adbdbstatus` command, see 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).

13.3.1 Number of rows and their output order in the summary information for DB areas

One row of summary information is output for each DB area. The number of rows output equals the number of DB areas for which summary information has been obtained. Title lines (column header rows) are not included in the row count.

The rows are output in ascending order of the DB area IDs.

13.3.2 List of items that are output as summary information for DB areas

The table below lists and describes the titles and contents of the items (columns) that are output as summary information for DB areas. The columns are output in the order listed in this table. For details about the output rules for summary information for DB areas, see 13.12 Output rules for the adbdbstatus command.

Table 13-16: Columns output as summary information for DB areas

No.	Column title	Information that is output	Enclosing character
1	`Type`	Type of information that is output: `DBarea`: Summary information for a DB area	N
2	`Timestamp`	Timestamp when the `adbdbstatus` command was executed The date and time the `adbdbstatus` was executed are output in the following format: `"`*YYYY*`-`*MM*`-`*DD*Δ*hh*`:`*mm*`:`*ss*`"`#1	Y
3	`DBarea_name`	DB area name	Y
4	`DBarea_ID`	DB area ID	N
5	`DBarea_type`	Type of DB area: • `data`: Data DB area	N

No.	Column title	Information that is output	Enclosing character
		- `dictionary`: Dictionary DB area - `system_table`: System-table DB area	
6	`DBarea_created_time`	Timestamp when the DB area was created The date and time the DB area was created by the `adbinit` command or was added by the `adbmodarea` command are output in the following format: "*YYYY*-*MM*-*DD*Δ*hh*:*mm*:*ss*"#1	Y
7	`DBarea_modified_time`	Timestamp when the DB area was expanded The date and time DB area files were added to the DB area by the `adbmodarea` command are output in the following format: "*YYYY*-*MM*-*DD*Δ*hh*:*mm*:*ss*"#1 If no DB area file has been added, no value is output for this item.	Y
8	`Rsv`	--	--
9	`Page_size`	Page size (bytes)	N
10	`Segment_size`	Segment size (pages)	N
11	`Top_DBarea_filename`	File path of the first DB area file This is the file path of the first DB area file allocated in the DB area. - For a regular file The absolute path of the DB area file is output. - For a block special file The name of the block special file allocated to the DB area file is output. For a symbolic link, the target of the symbolic link is output.	Y
12	`Number_of_files`	Number of DB area files This is the number of DB area files that make up the DB area.	N
13	`Creatable_chunks`	Number of chunks that can be created in the DB area	N
14	`Created_chunks`	Number of chunks that have been created in the DB area	N
15	`Rsv`	--	--
16	`Used_ratio`#2	Segment usage rate of the DB area (%) This is the percentage of segments currently in use out of the total number of segments in the DB area. The formula is as follows: ↑ (*number of segments in use in the DB area* ÷ *total number of segments in the DB area*) × 100 ↑	N
17	`Used_segments`#2, #3	Number of segments in use in the DB area#4 The number of segments being used in the DB area#4 is output.	N
18	`Total_segments`#2, #3	Total number of segments in the DB area#4 The total number of segments that can be managed, based on the size of the DB area file at the time the `adbdbstatus` command was executed#4, is output.	N
19	`Rsv`	--	--

Legend:

--: No value is output.

Y: The output value is enclosed in a pair of enclosing characters.

N: The output value is not enclosed in the enclosing character.

#1

The year (*YYYY*) is output as four digits. The month (*MM*), day (*DD*), hour (*hh*), minute (*mm*), and second (*ss*) are each output as two digits. If there are not enough digits, leading zeros are added. Δ indicates a single-byte space.

#2

If a transaction that updates the DB area that is being analyzed is executed concurrently with execution of the `adbdbstatus` command, the displayed value might differ from the actual value.

#3

When the display units for usage amounts are changed by specifying the `-S` option, the column titles for the usage amount output items are also changed to match the selected display units. For details about the display units and column titles, see Table 13-2: Display units and column titles for usage amount output items in (3) Explanation of options in 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).

#4

The `-S` option enables you to change the display units for usage amount output items to kilobytes, megabytes, or gigabytes.

13.3.3 Output example of summary information for DB areas

An output example of the summary information for DB areas is shown below. Because the rows in the actual output results are wide, they are shown in sections in this manual.

Figure 13-5: Output example of summary information for DB areas

Output example of summary information for DB areas (cols. 1 to 6)

```
Type  ,Timestamp            ,DBarea_name  ,DBarea_ID,DBarea_type ,DBarea_created_time  ,
DBarea,"2013-07-03 10:29:46","ADBDIC"     ,        2,dictionary  ,"2013-05-30 08:45:28",
DBarea,"2013-07-03 10:29:46","ADBSTBL"    ,        4,system_table,"2013-05-30 08:45:28",
DBarea,"2013-07-03 10:29:46","DBAREATBL01",       11,data        ,"2013-05-30 08:45:28",
DBarea,"2013-07-03 10:29:46","DBAREATBL02",       12,data        ,"2013-05-30 08:45:28",
DBarea,"2013-07-03 10:29:46","DBAREATBL03",       13,data        ,"2013-05-30 08:45:28",
```

Output example of summary information for DB areas (cols. 7 to 11)

```
DBarea_modified_time ,Rsv,Page_size,Segment_size,Top_DBarea_filename               ,
                     ,   ,     4096,           8,"/home/osuser01/DBDIR/ADBDIC"     ,
                     ,   ,     4096,          16,"/home/osuser01/DBDIR/ADBSTBL"    ,
"2013-08-23 14:11:07",   ,     4096,        1024,"/home/osuser01/DBDIR/DBAREATBL01",
                     ,   ,     4096,        1024,"/dev/ode/DBAREA_159"             ,
                     ,   ,     8192,         512,"/dev/ode/DBAREA_216"             ,
```

Output example of summary information for DB areas (cols. 12 to 17)

```
Number_of_files,Creatable_chunks,Created_chunks,Rsv,Used_ratio,MB_Used_segments#,
              1,           29963,            37,   ,       100,               4,
              1,           29991,             9,   ,       100,               2,
             10,           29743,           257,   ,        94,             268,
              3,           29488,           512,   ,         1,             704,
              1,           29999,             1,   ,         1,              88,
```

Output example of summary information for DB areas (cols. 18 and 19)

```
MB_Total_segments#,Rsv
                4,
                2,
              288,
           153624,
            51208,
```

#

The display units for usage amount output items and the column titles are changed based on the specified -S option. For details about the -S option, see (3) Explanation of options in 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).

13.4 Items that are output as summary information for tables

This section provides the details of the items that are output as summary information for tables.

To output summary information for tables, specify the following two options:

- `-d option`
  Specify `summary` in the `-d` option. Alternatively, omit the `-d` option.
- `-c option`
  Specify `table` in the `-c` option.

You can use the `-n` option to specify a specific table for which information is to be collected.

For details about the specification format and options of the `adbdbstatus` command, see 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).

13.4.1 Number of rows and their output order in the summary information for tables

One row of summary information is output for each table. The number of rows output equals the number of tables for which summary information has been obtained. Title lines (column header rows) are not included in the row count.

The rows are output in ascending order of the DB area IDs. Rows with the same DB area ID are output in ascending order of the table names (*schema-name*.*table-identifier*).

13.4.2 List of items that are output as summary information for tables

The table below lists and describes the titles and contents of the items (columns) that are output as summary information for tables. The columns are output in the order listed in this table. For details about the output rules for summary information for tables, see 13.12 Output rules for the adbdbstatus command.

Table 13-17: Columns output as summary information for tables

No.	Column title	Information that is output	Enclosing character
1	`Type`	Type of information that is output: `Table`: Summary information for a table `Table(Shared_lock)`: Summary information for a table (if the `--shared-lock` option is specified)	N
2	`Timestamp`	Timestamp when the `adbdbstatus` command was executed The date and time the `adbdbstatus` was executed are output in the following format: "*YYYY*-*MM*-*DD*Δ*hh*:*mm*:*ss*"[#1]	Y
3	`DBarea_name`	Name of the DB area where the table is stored	Y
4	`DBarea_ID`	DB area ID of the DB area where the table is stored	N

No.	Column title	Information that is output	Enclosing character
5	`Schema_name`	Schema name of the table The schema name for a dictionary table (base table) or a system table (base table) is always `HADB`.	Y
6	`Table_identifier`	Table identifier	Y
7	`Table_ID`	Table ID	N
8	`Table_type`	Table type: • `user`: User-defined table • `dictionary`: Dictionary table (base table) • `system`: System table (base table)	N
9	`Rsv`	--	--
10	`Page_size`	Page size (bytes)	N
11	`Segment_size`	Segment size (pages)	N
12	`Creatable_chunks`	Number of chunks that can be created for the table#4 The number of times background import can be performed is output. For a single-chunk table, `0` is output.	N
13	`Created_chunks`	Number of chunks created for the table#4 The value that is output as the number of chunks created for the table also includes the following chunks: • Delete-pending chunks • System chunks used temporarily by the `adbmergechunk` command This includes the number of chunks in wait status. For a single-chunk table, `1` is output. For a system table (base table), `1` or `2` is output.	N
14	`Current_chunk_ID`	Chunk ID of the current chunk for the table#4 For a single-chunk table, `1` is output. For a system table (base table), "1 plus the number of times the `adbreorgsystemdata` command is executed" is output.	N
15	`Pending_delete_chunks`	Number of delete-pending chunks for the table#4 For a multi-chunk table, the number of merge-source chunks whose deletion has not been completed by the `adbmergechunk` command is output. For a single-chunk table, `0` is output. For a system table (base table), `0` or `1` is output (if deletion of pre-reorganization data has not been completed, `1` is output).	N
16	`Non-updatable`	Whether the table is non-updatable • `non-updatable`: This is output if the table is non-updatable. • `no_check`: This is output if the `--shared-lock` option is specified. Information about whether the table is non-updatable cannot be acquired. In any other case, no value is output.	N
17	`Command_status`	Command termination status (whether the command needs to be re-executed) `abort` is output in one of the following cases: • If the command indicated in the `Rerun_command` column terminated abnormally during table or index creation	N

No.	Column title	Information that is output	Enclosing character
		Re-execute the command indicated in the `Rerun_command` column. Until that command is re-executed, other commands that create tables and indexes cannot be executed. The commands that create tables and indexes are as follows: - `adbimport` command - `adbidxrebuild` command - `adbunarchivechunk` command • If the command indicated in the `Rerun_command` column terminated abnormally before deletion of pre-reorganization data was completed. Re-execute the command indicated in the `Rerun_command` column. This applies to the following command: - `adbreorgsystemdata` command If the `--shared-lock` option is specified, `no_check` is output because information about the command termination status cannot be acquired. In any other case, no value is output.	
18	`Rerun_command`	Name of the command to be re-executed If the value in the `Command_status` column is `abort`, the name of the command that needs be re-executed is output. • `adbimport`: `adbimport` command • `adbidxrebuild`: `adbidxrebuild` command • `adbunarchivechunk`: `adbunarchivechunk` command • `adbreorgsystemdata`: `adbreorgsystemdata` command If no command needs to be re-executed, no value is output.	N
19	`Rsv`	--	--
20	`Used_segments`#2	Number of segments in use by the table#3, #5 The number of segments being used by the table#3, #5 is output.	N
21	`Used_pages`#2, #6	Number of pages in use by the table#3, #5 The number of pages being used by the table#3, #5 is output.	N
22	`Rsv`	--	--
23	`Wait_chunks`	Number of chunks in wait status#4 For a single-chunk table, `0` is output.	N
24	`Archive_chunks`	Number of archived chunks For a single-chunk table and regular multi-chunk table, `0` is output.	N
25	`Storage_format`	Table-data storage format • `row`: Row store format • `column`: Column store format	N

Legend:

--: No value is output.

Y: The output value is enclosed in a pair of enclosing characters.

N: The output value is not enclosed in the enclosing character.

#1

The year (*YYYY*) is output as four digits. The month (*MM*), day (*DD*), hour (*hh*), minute (*mm*), and second (*ss*) are each output as two digits. If there are not enough digits, leading zeros are added. Δ indicates a single-byte space.

#2

When the display units for usage amounts are changed by specifying the -S option, the column titles for the usage amount output items are also changed to match the selected display units. For details about the display units and column titles, see Table 13-2: Display units and column titles for usage amount output items in (3) Explanation of options in 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).

#3

The -S option enables you to change the display units for usage amount output items to kilobytes, megabytes, or gigabytes.

#4

If the adbdbstatus command with the --shared-lock option specified is executed at the same time as a transaction that updates the table to be processed, values being updated are output in the results of the adbdbstatus command. Therefore, a displayed value might be inconsistent with the values of other items. The following shows an example.

Example:

In the table to be processed, the value of Wait_chunks (number of chunks in wait status) is greater than the value of Created_chunks (number of chunks created for the table).

#5

If the --shared-lock option is specified, the values output for Used_segments and Used_pages (number of segments and number of pages in use by the table) of the following tables do not include the values relating to deletion-pending chunks:

- System table (base table)
- Table for which the adbmergechunk command is being executed

#6

This value includes the number of basic row pages that are made reusable by the updated-row columnizing facility. For details about the updated-row columnizing facility, see *Using the updated-row columnizing facility (maintaining the retrieval performance for column store tables)* in the *HADB Setup and Operation Guide*.

13.4.3 Output example of summary information for tables

An output example of the summary information for tables is shown below. Because the rows in the actual output results are wide, they are shown in sections in this manual.

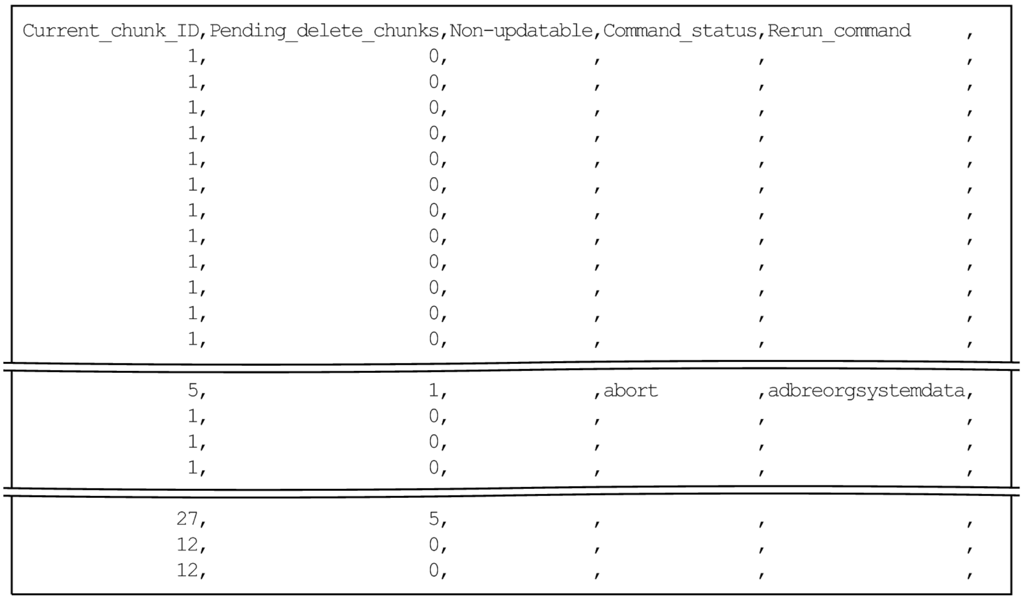
Figure 13-6: Output example of summary information for tables (columns 1 through 6)

```
Type ,Timestamp            ,DBarea_name  ,DBarea_ID,Schema_name,Table_identifier        ,
Table,"2013-07-03 10:29:47","ADBDIC"     ,        2,"HADB"     ,"SQL_COLUMNS"           ,
Table,"2013-07-03 10:29:47","ADBDIC"     ,        2,"HADB"     ,"SQL_DBAREAS"           ,
Table,"2013-07-03 10:29:47","ADBDIC"     ,        2,"HADB"     ,"SQL_DEFINE_ENVIRONMENT",
Table,"2013-07-03 10:29:47","ADBDIC"     ,        2,"HADB"     ,"SQL_DEFINE_SOURCE"     ,
Table,"2013-07-03 10:29:47","ADBDIC"     ,        2,"HADB"     ,"SQL_DIV_INDEX"         ,
Table,"2013-07-03 10:29:47","ADBDIC"     ,        2,"HADB"     ,"SQL_DIV_TABLE"         ,
Table,"2013-07-03 10:29:47","ADBDIC"     ,        2,"HADB"     ,"SQL_INDEXES"           ,
Table,"2013-07-03 10:29:47","ADBDIC"     ,        2,"HADB"     ,"SQL_SCHEMATA"          ,
Table,"2013-07-03 10:29:47","ADBDIC"     ,        2,"HADB"     ,"SQL_TABLES"            ,
Table,"2013-07-03 10:29:47","ADBDIC"     ,        2,"HADB"     ,"SQL_VIEWS"             ,
Table,"2013-07-03 10:29:47","ADBDIC"     ,        2,"HADB"     ,"SQL_VIEW_OBJECT"       ,
Table,"2013-07-03 10:29:47","ADBDIC"     ,        2,"HADB"     ,"SQL_VIEW_TABLE_USAGE"  ,

Table,"2013-07-03 10:29:47","ADBSTBL"    ,        4,"HADB"     ,"STATUS_CHUNKS"         ,
Table,"2013-07-03 10:29:47","ADBSTBL"    ,        4,"HADB"     ,"STATUS_COLUMNS"        ,
Table,"2013-07-03 10:29:47","ADBSTBL"    ,        4,"HADB"     ,"STATUS_INDEXES"        ,
Table,"2013-07-03 10:29:47","ADBSTBL"    ,        4,"HADB"     ,"STATUS_TABLES"         ,

Table,"2013-07-03 10:29:47","DBAREATBL02",       12,"ADBUSER01","TABLE01"               ,
Table,"2013-07-03 10:29:47","DBAREATBL03",       13,"ADBUSER01","TABLE02"               ,
Table,"2013-07-03 10:29:47","DBAREATBL03",       13,"ADBUSER01","TABLE03"               ,
```

Figure 13-7: Output example of summary information for tables (columns 7 through 13)

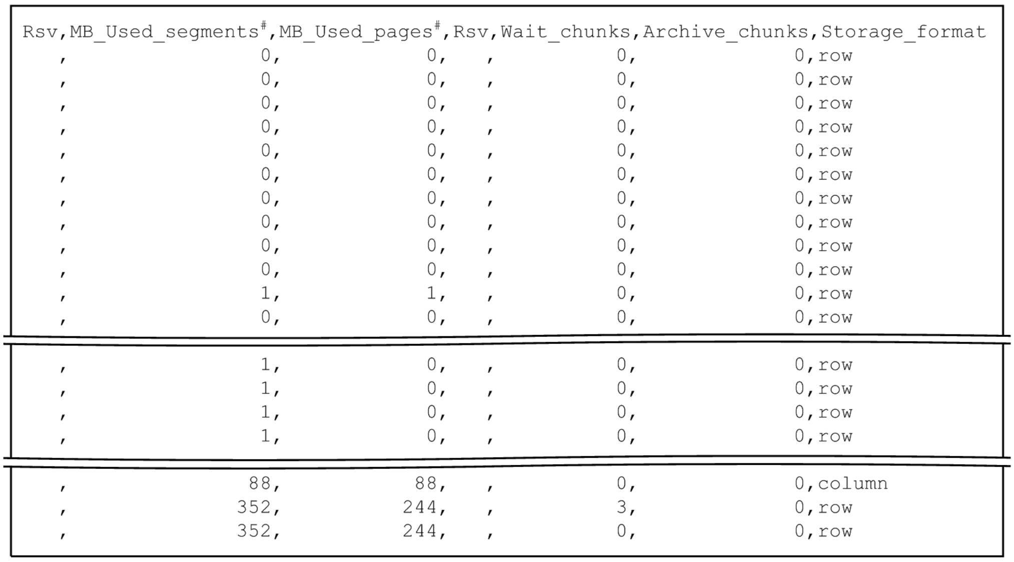
```
Table_ID,Table_type,Rsv,Page_size,Segment_size,Creatable_chunks,Created_chunks,
  131074,dictionary,   ,     4096,           8,               0,             1,
  131078,dictionary,   ,     4096,           8,               0,             1,
  131084,dictionary,   ,     4096,           8,               0,             1,
  131083,dictionary,   ,     4096,           8,               0,             1,
  131077,dictionary,   ,     4096,           8,               0,             1,
  131076,dictionary,   ,     4096,           8,               0,             1,
  131075,dictionary,   ,     4096,           8,               0,             1,
  131079,dictionary,   ,     4096,           8,               0,             1,
  131073,dictionary,   ,     4096,           8,               0,             1,
  131080,dictionary,   ,     4096,           8,               0,             1,
  131082,dictionary,   ,     4096,           8,               0,             1,
  131081,dictionary,   ,     4096,           8,               0,             1,

  131276,system    ,   ,     4096,          16,               0,             2,
  131274,system    ,   ,     4096,          16,               0,             1,
  131275,system    ,   ,     4096,          16,               0,             1,
  131273,system    ,   ,     4096,          16,               0,             1,

  131473,user      ,   ,     8192,         512,             200,            11,
  131474,user      ,   ,    32768,         128,             244,            12,
  131475,user      ,   ,    32768,         128,             244,            12,
```

Figure 13-8: Output example of summary information for tables (columns 14 through 18)



```
Current_chunk_ID,Pending_delete_chunks,Non-updatable,Command_status,Rerun_command     ,
              1,                   0,            ,             ,                 ,
              1,                   0,            ,             ,                 ,
              1,                   0,            ,             ,                 ,
              1,                   0,            ,             ,                 ,
              1,                   0,            ,             ,                 ,
              1,                   0,            ,             ,                 ,
              1,                   0,            ,             ,                 ,
              1,                   0,            ,             ,                 ,
              1,                   0,            ,             ,                 ,
              1,                   0,            ,             ,                 ,
              1,                   0,            ,             ,                 ,
              1,                   0,            ,             ,                 ,

              5,                   1,            ,abort        ,adbreorgsystemdata,
              1,                   0,            ,             ,                 ,
              1,                   0,            ,             ,                 ,
              1,                   0,            ,             ,                 ,

             27,                   5,            ,             ,                 ,
             12,                   0,            ,             ,                 ,
             12,                   0,            ,             ,                 ,
```

Figure 13-9: Output example of summary information for tables (columns 19 through 25)



```
Rsv,MB_Used_segments#,MB_Used_pages#,Rsv,Wait_chunks,Archive_chunks,Storage_format
   ,                0,             0,   ,          0,             0,row
   ,                0,             0,   ,          0,             0,row
   ,                0,             0,   ,          0,             0,row
   ,                0,             0,   ,          0,             0,row
   ,                0,             0,   ,          0,             0,row
   ,                0,             0,   ,          0,             0,row
   ,                0,             0,   ,          0,             0,row
   ,                0,             0,   ,          0,             0,row
   ,                0,             0,   ,          0,             0,row
   ,                0,             0,   ,          0,             0,row
   ,                1,             1,   ,          0,             0,row
   ,                0,             0,   ,          0,             0,row

   ,                1,             0,   ,          0,             0,row
   ,                1,             0,   ,          0,             0,row
   ,                1,             0,   ,          0,             0,row
   ,                1,             0,   ,          0,             0,row

   ,               88,            88,   ,          0,             0,column
   ,              352,           244,   ,          3,             0,row
   ,              352,           244,   ,          0,             0,row
```

#

The display units for usage amount output items and the column titles are changed based on the specified -S option. For details about the -S option, see (3) Explanation of options in 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).

13.5 Items that are output as summary information for indexes

This section provides the details of the items that are output as summary information for indexes.

To output summary information for indexes, specify the following two options:

- `-d` option
  Specify `summary` in the `-d` option. Alternatively, omit the `-d` option.
- `-c` option
  Specify `index` in the `-c` option.

You can use the `-n` option to specify a specific index for which information is to be collected.

For details about the specification format and options of the `adbdbstatus` command, see 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).

13.5.1 Number of rows and their output order in the summary information for indexes

One row of summary information is output for each index. The number of rows output equals the number of indexes for which summary information has been obtained. Title lines (column header rows) are not included in the row count.

The rows are output in ascending order of the DB area IDs. Rows with the same DB area ID are output in ascending order of the index names (*schema-name*.*index-identifier*).

13.5.2 List of items that are output as summary information for indexes

The table below lists and describes the titles and contents of the items (columns) that are output as summary information for indexes. The columns are output in the order listed in this table. For details about the output rules for summary information for indexes, see 13.12 Output rules for the adbdbstatus command.

Table 13-18: Columns output as summary information for indexes

No.	Column title	Information that is output	Enclosing character
1	`Type`	Type of information that is output: `Index`: Summary information for an index `Index(Shared_lock)`: Summary information for an index (if the `--shared-lock` option is specified)	N
2	`Timestamp`	Timestamp when the `adbdbstatus` command was executed: The date and time the `adbdbstatus` was executed are output in the following format: "*YYYY*-*MM*-*DD*Δ*hh*:*mm*:*ss*"#1	Y
3	`DBarea_name`	Name of the DB area where the index is stored	Y
4	`DBarea_ID`	DB area ID of the DB area where the index is stored	N

No.	Column title	Information that is output	Enclosing character
5	`Schema_name`	Schema name of the index: The schema name for an index defined for a dictionary table (base table) or system table (base table) is always `HADB`.	Y
6	`Index_identifier`	Index identifier	Y
7	`Index_ID`	Index ID	N
8	`Index_type`	Index type: • `B-tree`: B-tree index • `text`: Text index • `range`: Range index	N
9	`Schema_name`	Schema name of the table for which the index is defined	Y
10	`Table_identifier`	Table identifier of the table for which the index is defined	Y
11	`Table_ID`	Table ID of the table for which the index is defined	N
12	`Unique_constraint_violated`	Uniqueness constraint violation: • `unique_constraint_violated`: A uniqueness constraint violation occurred on a unique index. • `unknown`: Whether a uniqueness constraint violation occurred on a unique index cannot be determined. • `no_check`: This is output if the `--shared-lock` option is specified. Information about uniqueness constraint violation cannot be acquired. In any other case, no value is output.	N
13	`Page_size`	Page size (bytes)	N
14	`Segment_size`	Segment size (pages)	N
15	`Unfinished`	Whether the index is in unfinished status: • `unfinished`: This is output if the index is in unfinished status. • `no_check`: This is output if the `--shared-lock` option is specified. Information about whether the index is in unfinished status cannot be acquired. In any other case, no value is output.	N
16	`Pagegroup_size`	Page group size for a range index (pages): • No value is output for B-tree indexes or text indexes. • For a range index that was defined by a version earlier than 02-02, the segment size is output.	N
17	`Used_segments`#2	Number of segments in use by the index#3, #4: The number of segments being used by the index#3, #4 is output.	N
18	`Used_pages`#2	Number of pages in use by the index#3, #4: The number of pages being used by the index#3, #4 is output.	N
19	`Rsv`	--	--

Legend:

--: No value is output.

Y: The output value is enclosed in a pair of enclosing characters.

N: The output value is not enclosed in the enclosing character.

#1

The year (*YYYY*) is output as four digits. The month (*MM*), day (*DD*), hour (*hh*), minute (*mm*), and second (*ss*) are each output as two digits. If there are not enough digits, leading zeros are added. Δ indicates a single-byte space.

#2

When the display units for usage amounts are changed by specifying the `-S` option, the column titles for the usage amount output items are also changed to match the selected display units. For details about the display units and column titles, see Table 13-2: Display units and column titles for usage amount output items in (3) Explanation of options in 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).

#3

The `-S` option enables you to change the display units for usage amount output items to kilobytes, megabytes, or gigabytes.

#4

If the `--shared-lock` option is specified, the values output for `Used_segments` and `Used_pages` (number of segments and number of pages in use by the index) of the following tables do not include the values relating to deletion-pending chunks:

- Index defined for a system table (base table)
- Index defined for a table for which the `adbmergechunk` command is being executed

13.5.3 Output example of summary information for indexes

An output example of the summary information for indexes is shown below. Because the rows in the actual output results are wide, they are shown in sections in this manual.

Figure 13-10: Output example of summary information for indexes (columns 1 through 7)

```
Type ,Timestamp            ,DBarea_name  ,DBarea_ID,Schema_name,Index_identifier   ,Index_ID,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXM01"      ,  196609,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXM02"      ,  196611,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXM03"      ,  196613,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXM04"      ,  196614,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXM05"      ,  196616,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXM06"      ,  196618,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXM07"      ,  196619,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXM08"      ,  196623,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXM09"      ,  196626,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXM10"      ,  196628,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXM11"      ,  196629,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXM12"      ,  196631,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXM13"      ,  196633,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXS01"      ,  196610,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXS02"      ,  196612,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXS03"      ,  196615,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXS04"      ,  196617,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXS05"      ,  196620,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXS06"      ,  196621,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXS07"      ,  196622,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXS08"      ,  196624,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXS09"      ,  196625,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXS10"      ,  196627,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXS11"      ,  196630,
Index,"2013-07-03 10:29:48","ADBDIC"     ,        2,"HADB"     ,"SQLINDEXS12"      ,  196632,
Index,"2013-07-03 10:29:48","ADBSTBL"    ,        4,"HADB"     ,"STATUSINDEXM01"   ,  197009,
Index,"2013-07-03 10:29:48","ADBSTBL"    ,        4,"HADB"     ,"STATUSINDEXM02"   ,  197010,
Index,"2013-07-03 10:29:48","ADBSTBL"    ,        4,"HADB"     ,"STATUSINDEXM03"   ,  197011,
Index,"2013-07-03 10:29:48","ADBSTBL"    ,        4,"HADB"     ,"STATUSINDEXM04"   ,  197012,
Index,"2013-07-03 10:29:48","ADBSTBL"    ,        4,"HADB"     ,"STATUSINDEXM05"   ,  197013,
Index,"2013-07-03 10:29:48","DBAREAIDX01",       22,"ADBUSER01","TABLE04INDX1"     ,  197413,
Index,"2013-07-03 10:29:48","DBAREAIDX01",       22,"ADBUSER01","TABLE04INDX2"     ,  197414,
Index,"2013-07-03 10:29:48","DBAREAIDX01",       22,"ADBUSER01","TABLE04RANGINDX1" ,  197415,
Index,"2013-07-03 10:29:48","DBAREAIDX01",       22,"ADBUSER01","TABLE04RANGINDX2" ,  197416,
```

Figure 13-11: Output example of summary information for indexes (columns 8 through 13)

```
Index_type,Schema_name,Table_identifier        ,Table_ID,Unique_constraint_violated,Page_size,
B-tree    ,"HADB"     ,"SQL_TABLES"            ,  131073,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_COLUMNS"           ,  131074,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_INDEXES"           ,  131075,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_INDEXES"           ,  131075,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_DIV_TABLE"         ,  131076,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_DIV_INDEX"         ,  131077,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_DIV_INDEX"         ,  131077,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_DBAREAS"           ,  131078,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_VIEWS"             ,  131080,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_VIEW_TABLE_USAGE"  ,  131081,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_VIEW_TABLE_USAGE"  ,  131081,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_VIEW_OBJECT"       ,  131082,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_DEFINE_ENVIRONMENT",  131084,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_TABLES"            ,  131073,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_COLUMNS"           ,  131074,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_INDEXES"           ,  131075,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_DIV_TABLE"         ,  131076,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_DIV_INDEX"         ,  131077,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_DBAREAS"           ,  131078,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_DBAREAS"           ,  131078,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_SCHEMATA"          ,  131079,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_SCHEMATA"          ,  131079,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_VIEWS"             ,  131080,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_VIEW_TABLE_USAGE"  ,  131081,                          ,     4096,
B-tree    ,"HADB"     ,"SQL_DEFINE_SOURCE"     ,  131083,                          ,     4096,
B-tree    ,"HADB"     ,"STATUS_TABLES"         ,  131273,                          ,     4096,
B-tree    ,"HADB"     ,"STATUS_COLUMNS"        ,  131274,                          ,     4096,
B-tree    ,"HADB"     ,"STATUS_INDEXES"        ,  131275,                          ,     4096,
B-tree    ,"HADB"     ,"STATUS_INDEXES"        ,  131275,                          ,     4096,
B-tree    ,"HADB"     ,"STATUS_CHUNKS"         ,  131276,                          ,     4096,
B-tree    ,"ADBUSER01","TABLE04"               ,  131476,unique_constraint_violated,     8192,
B-tree    ,"ADBUSER01","TABLE04"               ,  131476,unknown                   ,     8192,
range     ,"ADBUSER01","TABLE04"               ,  131476,                          ,     8192,
range     ,"ADBUSER01","TABLE04"               ,  131476,                          ,     8192,
```

Figure 13-12: Output example of summary information for indexes (columns 14 through 19)

```
Segment_size ,Unfinished,Pagegroup_size,MB_Used_segments#,MB_Used_pages#,Rsv
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
            8,          ,              ,               0,             0,
           16,          ,              ,               1,             0,
           16,          ,              ,               1,             0,
           16,          ,              ,               1,             0,
           16,          ,              ,               1,             0,
           16,          ,              ,               1,             0,
          512,          ,              ,             668,           359,
          512,          ,              ,             668,           359,
          512,          ,             4,             488,           488,
          512,          ,             4,             488,           488,
```

#

The display units for usage amount output items and the column titles are changed based on the specified -S option. For details about the -S option, see (3) Explanation of options in 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).

13.6 Items that are output in the summary information of archived chunks

This section explains in detail about items that are output in the summary information of archived chunks.

When you want to output summary information about archived chunks, specify the following three options:

- `-d` option
  Specify `summary` in the `-d` option. Alternatively, omit the `-d` option.
- `-c` option
  Specify `archivechunk` in the `-c` option.
- `-n` option
  Specify the table name of the archivable multi-chunk table in the `-n` option.

For details about the specification format and options of the `adbdbstatus` command, see 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).

13.6.1 Number of rows that are output, and order of output, for the summary information of archived chunks

For each chunk created in the table, one row is output in the summary information of archived chunks. Accordingly, the number of rows that are output in the summary information of archived chunks is the same as the number of chunks created in the table. The column title is not included in the number of rows.

Note that there is no particular output order for these rows (information is not necessarily output in increasing or decreasing order of chunk ID).

13.6.2 List of items that are output in the summary information of archived chunks

The following table lists the titles and output content of items (columns) that are output in the summary information of archived chunks. Each column's information is output in the order of the item numbers in the following table. For the rules for outputting archived chunks' summary information, see 13.12 Output rules for the adbdbstatus command.

Table 13-19: Columns output as summary information for archived chunks

No.	Column title	Information that is output	Enclosing character
1	`Type`	Type of output information `Archive_chunk`: Summary information of archived chunk	N
2	`Timestamp`	Time stamp of `adbdbstatus` command execution The date and time the `adbdbstatus` command was executed is output in the following format: "*YYYY*-*MM*-*DD*Δ*hh*:*mm*:*ss*"#	Y
3	`Schema_name`	Table schema name	Y
4	`Table_identifier`	Table identifier	Y

No.	Column title	Information that is output	Enclosing character
5	`Table_ID`	Table ID	N
6	`Chunk_ID`	Chunk ID	N
7	`Chunk_status`	Chunk status • `Normal`: Normal status • `Wait`: Wait status • `Pending_delete`: Deletion-pending status	N
8	`Archive_status`	Chunk archived state `Archived`: Archived state If a chunk is not in archived state, nothing is output.	N
9	`Range_min`	Minimum value of the archive range column Information is output as the data type in the archive range column. However, if any of the following apply, nothing is output: • No data is stored in the chunk. • The chunk is pending deletion. • The range index defined in the archive range column is in unfinished status, and the chunk has not been archived.	Y
10	`Range_min_raw`	Minimum value (hexadecimal) of the archive range column Information is output as hexadecimal-format binary data only when the data type in the archive range column is `CHARACTER`. However, if any of the following apply, nothing is output: • No data is stored in the chunk. • The chunk is pending deletion. • The range index defined in the archive range column is in unfinished status, and the chunk has not been archived.	Y
11	`Range_max`	Maximum value of the archive range column Information is output as the data type in the archive range column. However, if any of the following apply, nothing is output: • No data is stored in the chunk. • The chunk is pending deletion. • The range index defined in the archive range column is in unfinished status, and the chunk has not been archived.	Y
12	`Range_max_raw`	Maximum value (hexadecimal) of the archive range column Information is output as hexadecimal-format binary data only when the data type in the archive range column is `CHARACTER`. However, if any of the following apply, nothing is output: • No data is stored in the chunk. • The chunk is pending deletion. • The range index defined in the archive range column is in unfinished status, and the chunk has not been archived.	Y
13	`Rows`	Number of rows stored in the archived chunk This is only output if the chunk is in archived state.	--
14	`Archive_file_num`	Total number of archive files corresponding to the archived chunk If a chunk is not in archived state, `0` is output.	--
15	`Archive_file_size`	Total size (units: bytes) of all archive files corresponding to the archived chunk If a chunk is not in archived state, `0` is output.	--

No.	Column title	Information that is output	Enclosing character
16	`Unarchive_table_size`	Size (units: bytes) of segment used to store the table of chunks that have not yet been archived If a chunk is not in archived state, `0` is output.	--
17	`Unarchive_index_size`	Size of segment (units: bytes) used to store the indexes of chunks that have not yet been archived If any indexes are defined in the target table, the size of the segment used to store the indexes of chunks that have not yet been archived is output. If a chunk is not in archived state, `0` is output.	--
18	`Compression_ratio`	Compression rate (units: percent) of archived chunks This is only output if the chunk is in archived state. However, even if a chunk is in archived state, if there are 0 rows stored in the chunk, nothing is output.	--
19	`Chunk_comment`	Comment specified in a chunk This is output only when a chunk comment is specified in a chunk.	Y
20	`Rsv`	--	--

Legend:

--: No value is output.

Y: The output value is enclosed in enclosing characters.

N: The output value is not enclosed in enclosing characters.

#

The year (*YYYY*) is output in 4 digits. The month (*MM*), date (*DD*), hours (*hh*), minutes (*mm*), and seconds (*ss*) are each output in 2 digits. If there are not enough digits in a value, the left side of the value is padded with zeros (`0`). Δ represents a halfwidth space.

13.6.3 Examples of outputting the summary information of archived chunks

The following are examples of outputting the summary information of archived chunks. Note that the actual output results are shown here in parts due to their horizontal length.

Figure 13-13: Examples of outputting the summary information of archived chunks (columns 1-6)

```
Type         ,Timestamp            ,Schema_name,Table_identifier,Table_ID ,Chunk_ID,
Archive_chunk,"2015-10-19 10:14:32","USER01"   ,"TABLE01"       ,   131473,       3,
Archive_chunk,"2015-10-19 10:14:32","USER01"   ,"TABLE01"       ,   131473,       4,
Archive_chunk,"2015-10-19 10:14:32","USER01"   ,"TABLE01"       ,   131473,       5,
Archive_chunk,"2015-10-19 10:14:32","USER01"   ,"TABLE01"       ,   131473,       7,
Archive_chunk,"2015-10-19 10:14:32","USER01"   ,"TABLE01"       ,   131473,       8,
Archive_chunk,"2015-10-19 10:14:32","USER01"   ,"TABLE01"       ,   131473,       9,
Archive_chunk,"2015-10-19 10:14:32","USER01"   ,"TABLE01"       ,   131473,      10,
Archive_chunk,"2015-10-19 10:14:32","USER01"   ,"TABLE01"       ,   131473,     108,
Archive_chunk,"2016-04-20 14:04:32","USER01"   ,"TABLE02"       ,   131475,       3,
Archive_chunk,"2016-04-20 14:04:32","USER01"   ,"TABLE02"       ,   131475,       4,
```

Figure 13-14: Examples of outputting the summary information of archived chunks (columns 7-12)

```
Chunk_status,Archive_status,Range_min   ,Range_min_raw     ,Range_max   ,Range_max_raw     ,
Normal      ,Archived      ,"2007-03-01",                  ,"2007-03-31",                  ,
Normal      ,Archived      ,"2007-04-01",                  ,"2007-04-30",                  ,
Normal      ,              ,"2007-05-01",                  ,"2007-05-31",                  ,
Normal      ,              ,"2007-07-01",                  ,"2007-07-31",                  ,
Normal      ,              ,"2007-08-01",                  ,"2007-08-31",                  ,
Normal      ,              ,"2007-09-01",                  ,"2007-09-30",                  ,
Normal      ,              ,"2007-10-01",                  ,"2007-10-30",                  ,
Wait        ,Archived      ,"2015-12-01",                  ,"2015-12-31",                  ,
Normal      ,Archived      ,"20160401"  ,"3230313630343031","20160430"  ,"3230313630343330",
Normal      ,              ,"20160501"  ,"3230313630353031","20160531"  ,"3230313630353331",
```

Figure 13-15: Examples of outputting the summary information of archived chunks (columns 13-17)

```
Rows ,Archive_file_num,Archive_file_size,Unarchive_table_size,Unarchive_index_size,
12345,               2,           874081,             4194304,             8388608,
24680,               4,          1797376,             8388608,            16777216,
     ,               0,                0,                   0,                   0,
     ,               0,                0,                   0,                   0,
     ,               0,                0,                   0,                   0,
     ,               0,                0,                   0,                   0,
     ,               0,                0,                   0,                   0,
12354,               2,           875434,             4194304,             8388608,
12345,               2,           874081,             4194304,             8388608,
24680,               0,                0,                   0,                   0,
```

Figure 13-16: Examples of outputting the summary information of archived chunks (columns 18-20)

```
Compression_ratio,Chunk_comment,Rsv
               20,"2007-3"     ,
               21,"2007-4"     ,
                 ,"2007-5"     ,
                 ,"2007-7"     ,
                 ,"2007-8"     ,
                 ,"2007-9"     ,
                 ,"2007-10"    ,
               21,"2015-12"    ,
               20,"2016-4"     ,
                 ,"2016-5"     ,
```

13.7 Items that are output as usage information for DB areas, tables, and indexes

This section provides details about the items that are output as usage information for DB areas, tables, and indexes.

To output usage information for DB areas, tables, and indexes, specify `used` in the `-d` option of the `adbdbstatus` command.

You can use the `-c` option to specify a range of information to be collected (DB area, table, or index) and the `-n` option to specify the name of the target that is to be subject to information collection.

For details about the specification format and options of the `adbdbstatus` command, see 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).

13.7.1 Output order of usage information for DB areas, tables, and indexes

The rows containing usage information for DB areas, tables, and indexes are output in ascending order of the DB area IDs.

When multiple DB areas have the same DB area ID, the information to be output differs depending on whether tables and indexes are stored in the applicable DB area.

The rules applicable to the information output order are explained separately below for the following cases:

- For DB areas that store tables and indexes
- For DB areas that do not store tables and indexes

(1) DB areas that store tables and indexes

When multiple DB areas that have the same DB area ID store tables and indexes, information is output in the following order:

1. Table and index usage
2. DB area file usage

The following rules apply to the output order of each of these information items.

Output order of the usage values for tables and indexes

If one DB area stores both tables and indexes, the usage values for tables are output first and then the usage values for indexes are output.

Output order of the usage values for tables

The usage values for the tables stored in the same DB area are output according to the following rules:

1. When DB area files consist of multiple block special files, order in which the DB area files were specified when the `adbinit` or `adbmodarea` command was executed

   The DB area file names are output to the `DBarea_filename` column.

2. Ascending order of the table names (*schema-name* (`Schema_name` column)`.`*table-identifier* (`Object_identifier` column))

Output order of the usage values for indexes

The usage values for the indexes stored in the same DB area are output according to the following rules:

1. When DB area files consist of multiple block special files, order in which the DB area files were specified when the `adbinit` or `adbmodarea` command was executed

   The DB area file names are output to the `DBarea_filename` column.

2. Ascending order of the index names (*schema-name* (`Schema_name` column)`.`*index-identifier* (`Object_identifier` column))

Output order of the usage values for DB area files

If one DB area contains DB area files that consist of multiple block special files, the usage values for the DB area files are output in the order in which the DB area files were defined when the `adbinit` or `adbmodarea` command was executed (the DB area file names are output to the `DBarea_filename` column).

(2) DB areas that do not store tables and indexes

If a DB area does not store tables and indexes, only DB area file usage is output. Table and index usage is not output.

The following rule applies to the output order of DB area file usage.

Output order of DB area file usage

If a single DB area contains a DB area file that uses multiple block special files, DB area file usage is output according to the order in which DB area files were defined by executing the `adbinit` or `adbmodarea` command. (The DB area file name is output to the `DBarea_filename` column.)

13.7.2 List of items that are output as usage information for DB areas, tables, and indexes

The table below lists and describes the titles and contents of the items (columns) that are output as usage information for DB areas, tables, and indexes. The columns are output in the order listed in this table. For details about the output rules for usage information for DB areas, tables, and indexes, see 13.12 Output rules for the adbdbstatus command.

Table 13-20: Columns output as usage information for DB areas, tables, and indexes

No.	Column title	Information that is output	Enclosing character
1	`Type`	Type of information that is output[#6] `Used`: Usage information for DB areas, tables, and indexes `Used(Shared_lock)`: Usage information for DB areas, tables, and indexes (if the `--shared-lock` option is specified)	N
2	`Timestamp`	Timestamp when the `adbdbstatus` command was executed[#6] The date and time the `adbdbstatus` was executed are output in the following format: `"`*YYYY*`-`*MM*`-`*DD*Δ*hh*`:`*mm*`:`*ss*`"`[#1]	Y
3	`DBarea_name`	DB area name[#6]	Y
4	`DBarea_ID`	DB area ID[#6]	N
5	`DBarea_type`	Type of DB area[#6] • `dictionary`: Dictionary DB area • `system_table`: System-table DB area • `data`: Data DB area	N

No.	Column title	Information that is output	Enclosing character
6	`Rsv`	--	--
7	`Page_size`	Page size (bytes)#6	N
8	`Segment_size`	Segment size (pages)#6	N
9	`DBarea_filename`	DB area file name#6 File path of a DB area file that was allocated in the DB area is output. • For a regular file The absolute path of the DB area file is output. • For a block special file The name of the block special file allocated to the DB area file is output. For a symbolic link, the target of the symbolic link is output.	Y
10	`Rsv`	--	--
11	`Object_type`	Schema object type#2, #6 • `table`: Table • `index`: Index	N
12	`Schema_name`	Schema name#2, #6 The schema name for a dictionary table (base table), system table (base table), or an index defined for such a table is always `HADB`.	Y
13	`Object_identifier`	Schema object identifier#2, #6 The table or index identifier is output.	Y
14	`Object_ID`	Schema object ID#2, #6 The table or index ID is output.	N
15	`Chunk_ID`	Chunk ID#2, #6	N
16	`Chunk_create_time`	Timestamp when the chunk was created#2, #6 The date and time the chunk was created are output in the following format: `"`*YYYY*`-`*MM*`-`*DD*Δ*hh*`:`*mm*`:`*ss*`"`#1 Outputs are different for the following cases: • For merged chunks, the date and time on which the oldest merge-source chunk was created is output. • For chunks that were created via current chunk swapping caused by a chunk status change, the date and time when the `adbchgchunkstatus` command was executed is output. • For chunks whose status is changed to the normal status by using the `adbchgchunkstatus` command after having been created as wait status chunks via a background import, the date and time when the `adbchgchunkstatus` command was executed is output. • For chunks whose status is not changed to the normal status by using the `adbchgchunkstatus` command after having been created as wait status chunks via a background import, no value is output for this field. • For a system table (base table), the date and time the `adbinit` command was executed or the date and time the `adbreorgsystemdata` command was last executed are output. However, if `Pending_delete` is indicated in the `Chunk_status` column, the last execution date and time of the `adbreorgsystemdata` command that has completed	Y

No.	Column title	Information that is output	Enclosing character
		deletion of pre-reorganization data are output. The execution date and time of the adbreorgsystemdata command for which pre-reorganization data remains are not output.	
17	Chunk_swap_time	Timestamp when chunks were last swapped#2, #6, #7 The date and time a new chunk was created and became the current chunk are output in the following format: "*YYYY*-*MM*-*DD*Δ*hh*:*mm*:*ss*"#1 If chunks have not been swapped, no value is output. Outputs are different for the following cases: - For merged chunks, the date and time on which the newest current chunk swapping occurred in the merge-source chunks is output. - For chunks whose status is changed to the normal status by using the adbchgchunkstatus command after having been created as wait status chunks via a background import, and that did not become a current chunk when the status was changed to normal status, the date and time when the adbchgchunkstatus command was executed is output. - For a system table (base table), the following date and time are output only when Pending_delete is indicated in the Chunk_status column. - The last execution date and time of the adbreorgsystemdata command that has created post-reorganization data but that failed to delete pre-reorganization data	Y
18	Chunk_status	Chunk status#2, #6, #7 - Normal: Normal status - Wait: Wait status - Pending_delete: Delete-pending status	N
19	Segment_type	Segment type#2, #6 **For tables (row store format):** - Base_row: Basic row segment - Branch_row: Branch row segment **For tables (column store format):** - Column_data: Column-data segment - Row_data: Row-data segment **For B-tree indexes:** - Lower_page: Lower-page segment of a B-tree index - Upper_page: Upper-page segment of a B-tree index **For text indexes:** - String_page: String control segment - Position_page: Position control segment **For range indexes:** - Range_control: DB area file control segment - Range_index_control: Range control segment	N
20	Used_segments#3	Number of segments in use#4, #6, #7 The number of segments being used for each segment type or DB area file#4 is output.	N
21	Used_pages#3, #8	Number of pages in use#2, #4, #6 The number of pages being used for each segment type#4 is output.	N

No.	Column title	Information that is output	Enclosing character
22	`Total_segments`[#3]	Total number of segments in the DB area file[#4, #5, #7] The number of segments that can be managed, based on the size of the DB area file at the time the `adbdbstatus` command was executed[#4], is output.	N
23	`Initialized_pages`[#3]	Number of initialized pages in the DB area file[#4, #5, #7] This includes the number of directory pages. For details about directory pages, see *Page types* in *Pages* in *DB area structure (segments and pages)* in the *HADB Setup and Operation Guide*.	N
24	`Storage_format`	Chunk-data storage format[#2] **For tables:** • `row`: Row store format • `column`: Column store format **Other than tables** No value is output.	N

Legend:

--: No value is output.

Y: The output value is enclosed in a pair of enclosing characters.

N: The output value is not enclosed in the enclosing character.

#1

The year (*YYYY*) is output as four digits. The month (*MM*), day (*DD*), hour (*hh*), minute (*mm*), and second (*ss*) are each output as two digits. If there are not enough digits, leading zeros are added. Δ indicates a single-byte space.

#2

This value is output on a row for which usage information for a table or index is output. The following shows the number of rows for which table and index usage is output:

segment type × *number of DB area files* × *number of chunks*

#3

When the display units for usage amounts are changed by specifying the `-S` option, the column titles for the usage amount output items are also changed to match the selected display units. For details about the display units and column titles, see Table 13-2: Display units and column titles for usage amount output items in (3) Explanation of options in 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).

#4

The `-S` option enables you to change the display units for usage amount output items to kilobytes, megabytes, or gigabytes.

#5

This value is output on a row for which usage information for a DB area file is output. The number of rows for which DB area file usage is output is the same as the number of DB area files that comprise the DB area.

#6

If the `adbdbstatus` command with the `--shared-lock` option specified is executed, rows that contain information about deletion-pending chunks might not be output for the following tables:

- System table (base table)
- Table for which the `adbmergechunk` command is being executed

#7

If the adbdbstatus command with the --shared-lock option specified is executed at the same time as a transaction that updates the table to be processed, values being updated are output in the results of the adbdbstatus command. Therefore, a displayed value might be inconsistent with the values of other items. The following shows examples.

Example 1:

The sum of the values of Used_segments (number of segments in use) output for the same DB area file name (under DBarea_filename) does not match the value of Total_segments (total number of segments in the DB area file).

Example 2:

For the current chunk, Chunk_swap_time (timestamp when chunks were last swapped) is output.

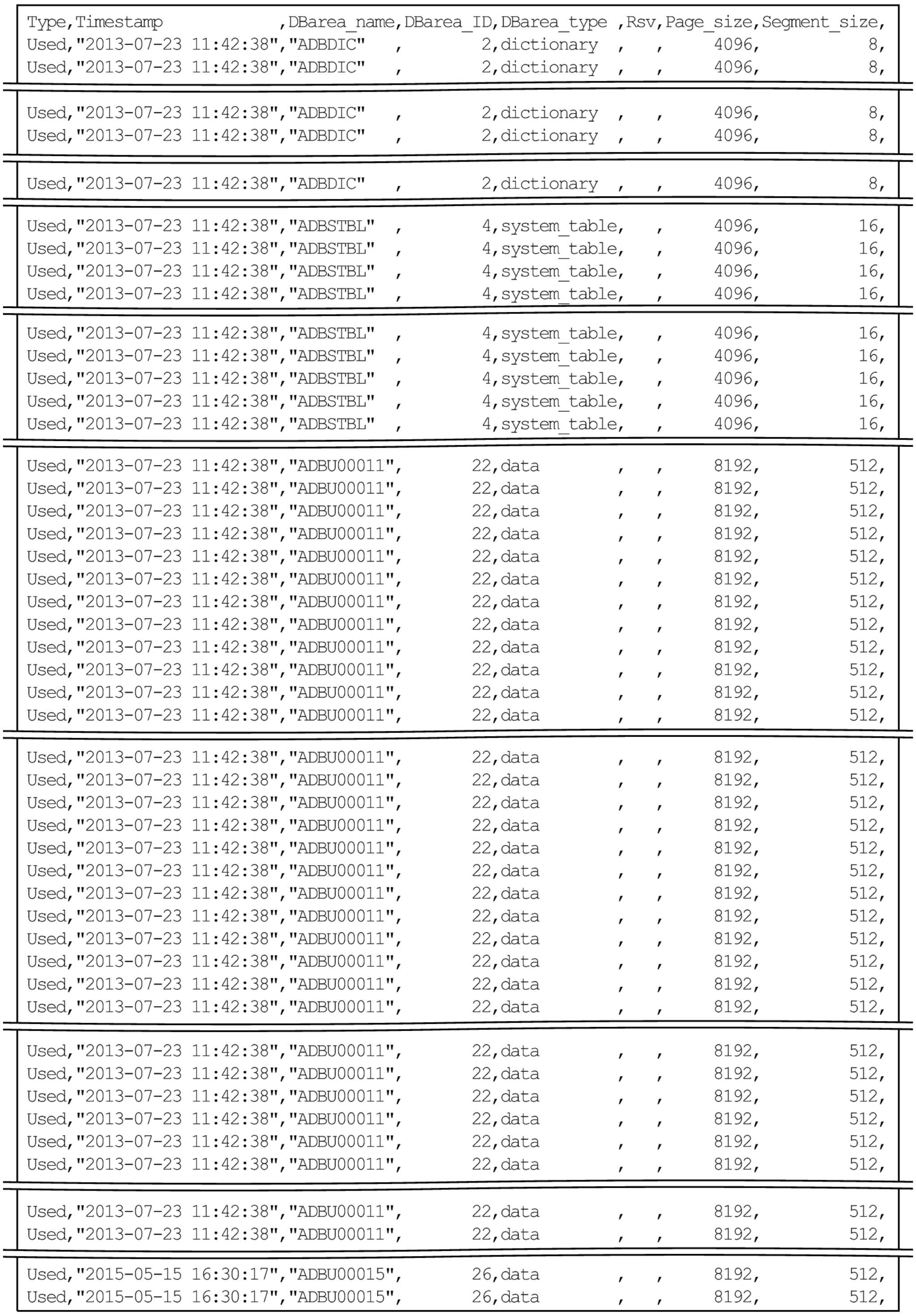
#8

This value includes the number of basic row pages that are made reusable by the updated-row columnizing facility. For details about the updated-row columnizing facility, see *Using the updated-row columnizing facility (maintaining the retrieval performance for column store tables)* in the *HADB Setup and Operation Guide*.

13.7.3 Output example of usage information for DB areas, tables, and indexes

The following shows output examples of usage information for DB areas, tables, and indexes. Because the rows in the actual output results are wide, they are shown in sections in this manual. Also, some rows are omitted here.

Figure 13-17: Output example of usage information for DB areas, tables, and indexes (columns 1 through 8)



```
Type,Timestamp            ,DBarea_name,DBarea_ID,DBarea_type ,Rsv,Page_size,Segment_size,
Used,"2013-07-23 11:42:38","ADBDIC"   ,        2,dictionary  ,   ,     4096,           8,
Used,"2013-07-23 11:42:38","ADBDIC"   ,        2,dictionary  ,   ,     4096,           8,

Used,"2013-07-23 11:42:38","ADBDIC"   ,        2,dictionary  ,   ,     4096,           8,
Used,"2013-07-23 11:42:38","ADBDIC"   ,        2,dictionary  ,   ,     4096,           8,

Used,"2013-07-23 11:42:38","ADBDIC"   ,        2,dictionary  ,   ,     4096,           8,

Used,"2013-07-23 11:42:38","ADBSTBL"  ,        4,system_table,   ,     4096,          16,
Used,"2013-07-23 11:42:38","ADBSTBL"  ,        4,system_table,   ,     4096,          16,
Used,"2013-07-23 11:42:38","ADBSTBL"  ,        4,system_table,   ,     4096,          16,
Used,"2013-07-23 11:42:38","ADBSTBL"  ,        4,system_table,   ,     4096,          16,

Used,"2013-07-23 11:42:38","ADBSTBL"  ,        4,system_table,   ,     4096,          16,
Used,"2013-07-23 11:42:38","ADBSTBL"  ,        4,system_table,   ,     4096,          16,
Used,"2013-07-23 11:42:38","ADBSTBL"  ,        4,system_table,   ,     4096,          16,
Used,"2013-07-23 11:42:38","ADBSTBL"  ,        4,system_table,   ,     4096,          16,
Used,"2013-07-23 11:42:38","ADBSTBL"  ,        4,system_table,   ,     4096,          16,

Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,

Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,

Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,

Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,
Used,"2013-07-23 11:42:38","ADBU00011",       22,data        ,   ,     8192,         512,

Used,"2015-05-15 16:30:17","ADBU00015",       26,data        ,   ,     8192,         512,
Used,"2015-05-15 16:30:17","ADBU00015",       26,data        ,   ,     8192,         512,
```

Figure 13-18: Output example of usage information for DB areas, tables, and indexes (columns 9 through 13)

```
DBarea_filename                    ,Rsv,Object_type,Schema_name,Object_identifier,
"/home/10608/DBDIR/ADBDIC"         ,   ,table      ,"HADB"     ,"SQL_COLUMNS"    ,
"/home/10608/DBDIR/ADBDIC"         ,   ,table      ,"HADB"     ,"SQL_COLUMNS"    ,

"/home/10608/DBDIR/ADBDIC"         ,   ,index      ,"HADB"     ,"SQLindexM01"    ,
"/home/10608/DBDIR/ADBDIC"         ,   ,index      ,"HADB"     ,"SQLindexM01"    ,

"/home/10608/DBDIR/ADBDIC"         ,   ,           ,           ,                 ,

"/home/10608/DBDIR/ADBSTBL"        ,   ,table      ,"HADB"     ,"STATUS_CHUNKS"  ,
"/home/10608/DBDIR/ADBSTBL"        ,   ,table      ,"HADB"     ,"STATUS_CHUNKS"  ,
"/home/10608/DBDIR/ADBSTBL"        ,   ,table      ,"HADB"     ,"STATUS_CHUNKS"  ,
"/home/10608/DBDIR/ADBSTBL"        ,   ,table      ,"HADB"     ,"STATUS_CHUNKS"  ,

"/home/10608/DBDIR/ADBSTBL"        ,   ,index      ,"HADB"     ,"STATUSINDEXM05" ,
"/home/10608/DBDIR/ADBSTBL"        ,   ,index      ,"HADB"     ,"STATUSINDEXM05" ,
"/home/10608/DBDIR/ADBSTBL"        ,   ,index      ,"HADB"     ,"STATUSINDEXM05" ,
"/home/10608/DBDIR/ADBSTBL"        ,   ,index      ,"HADB"     ,"STATUSINDEXM05" ,
"/home/10608/DBDIR/ADBSTBL"        ,   ,           ,           ,                 ,

"/home/10608/DBDIR/ADBU00011"      ,   ,table      ,"ADBUSER01","TBL014"         ,
"/home/10608/DBDIR/ADBU00011"      ,   ,table      ,"ADBUSER01","TBL014"         ,
"/home/10608/DBDIR/ADBU00011"      ,   ,table      ,"ADBUSER01","TBL014"         ,
"/home/10608/DBDIR/ADBU00011"      ,   ,table      ,"ADBUSER01","TBL014"         ,
"/home/10608/DBDIR/ADBU00011"      ,   ,table      ,"ADBUSER01","TBL014"         ,
"/home/10608/DBDIR/ADBU00011"      ,   ,table      ,"ADBUSER01","TBL014"         ,
"/home/10608/DBDIR/ADBU00011.00001",   ,table      ,"ADBUSER01","TBL014"         ,
"/home/10608/DBDIR/ADBU00011.00001",   ,table      ,"ADBUSER01","TBL014"         ,
"/home/10608/DBDIR/ADBU00011.00001",   ,table      ,"ADBUSER01","TBL014"         ,
"/home/10608/DBDIR/ADBU00011.00001",   ,table      ,"ADBUSER01","TBL014"         ,
"/home/10608/DBDIR/ADBU00011.00001",   ,table      ,"ADBUSER01","TBL014"         ,
"/home/10608/DBDIR/ADBU00011.00001",   ,table      ,"ADBUSER01","TBL014"         ,

"/home/10608/DBDIR/ADBU00011"      ,   ,table      ,"ADBUSER"  ,"TBL015"         ,
"/home/10608/DBDIR/ADBU00011"      ,   ,table      ,"ADBUSER"  ,"TBL015"         ,
"/home/10608/DBDIR/ADBU00011"      ,   ,table      ,"ADBUSER"  ,"TBL015"         ,
"/home/10608/DBDIR/ADBU00011"      ,   ,table      ,"ADBUSER"  ,"TBL015"         ,
"/home/10608/DBDIR/ADBU00011"      ,   ,table      ,"ADBUSER"  ,"TBL015"         ,
"/home/10608/DBDIR/ADBU00011"      ,   ,table      ,"ADBUSER"  ,"TBL015"         ,
"/home/10608/DBDIR/ADBU00011.00001",   ,table      ,"ADBUSER"  ,"TBL015"         ,
"/home/10608/DBDIR/ADBU00011.00001",   ,table      ,"ADBUSER"  ,"TBL015"         ,
"/home/10608/DBDIR/ADBU00011.00001",   ,table      ,"ADBUSER"  ,"TBL015"         ,
"/home/10608/DBDIR/ADBU00011.00001",   ,table      ,"ADBUSER"  ,"TBL015"         ,
"/home/10608/DBDIR/ADBU00011.00001",   ,table      ,"ADBUSER"  ,"TBL015"         ,
"/home/10608/DBDIR/ADBU00011.00001",   ,table      ,"ADBUSER"  ,"TBL015"         ,

"/home/10608/DBDIR/ADBU00011"      ,   ,index      ,"ADBUSER01","TBL014RIDX1"    ,
"/home/10608/DBDIR/ADBU00011"      ,   ,index      ,"ADBUSER01","TBL014RIDX1"    ,
"/home/10608/DBDIR/ADBU00011"      ,   ,index      ,"ADBUSER01","TBL014RIDX1"    ,
"/home/10608/DBDIR/ADBU00011"      ,   ,index      ,"ADBUSER01","TBL014RIDX1"    ,
"/home/10608/DBDIR/ADBU00011"      ,   ,index      ,"ADBUSER01","TBL014RIDX1"    ,
"/home/10608/DBDIR/ADBU00011"      ,   ,index      ,"ADBUSER01","TBL014RIDX1"    ,

"/home/10608/DBDIR/ADBU00011"      ,   ,           ,           ,                 ,
"/home/10608/DBDIR/ADBU00011.00001",   ,           ,           ,                 ,

"/home/10608/DBDIR/ADBU00015"      ,   ,           ,           ,                 ,
"/home/10608/DBDIR/ADBU00015.00001",   ,           ,           ,                 ,
```

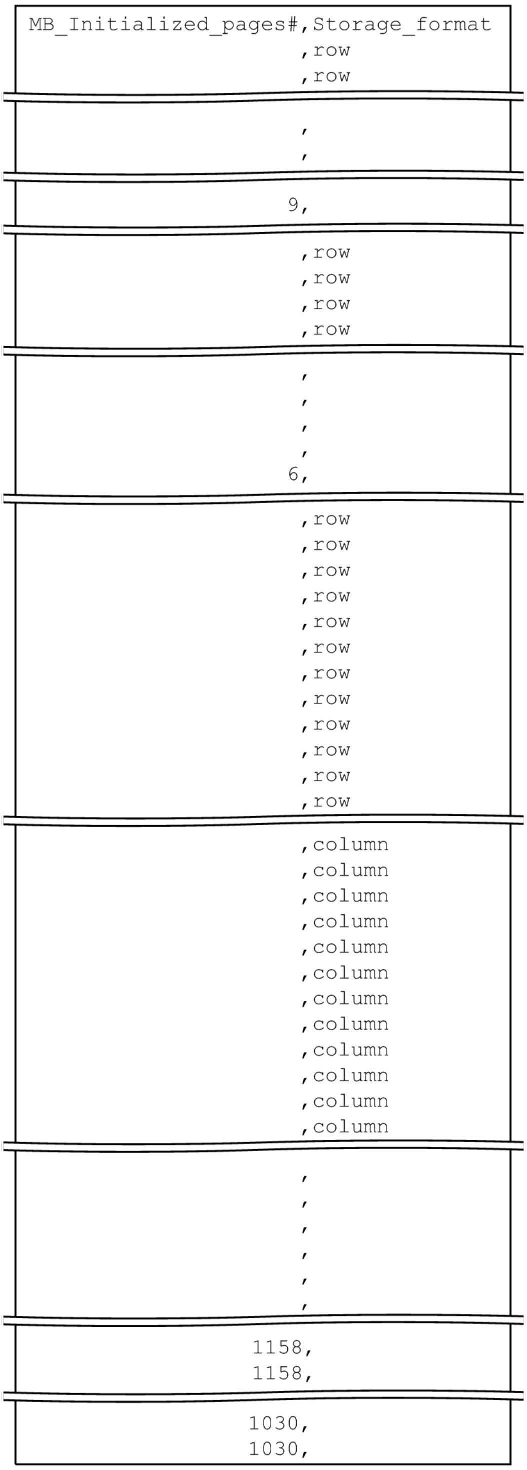
Figure 13-19: Output example of usage information for DB areas, tables, and indexes (columns 14 through 18)

```
Object_ID,Chunk_ID,Chunk_create_time    ,Chunk_swap_time      ,Chunk_status  ,
   131074,       1,"2013-07-18 16:51:34",                     ,Normal        ,
   131074,       1,"2013-07-18 16:51:34",                     ,Normal        ,

   196609,       1,"2013-07-18 16:51:34",                     ,Normal        ,
   196609,       1,"2013-07-18 16:51:34",                     ,Normal        ,

         ,        ,                     ,                     ,              ,

   131276,       5,"2013-07-19 20:41:35","2013-07-22 15:44:20",Pending_delete,
   131276,       5,"2013-07-19 20:41:35","2013-07-22 15:44:20",Pending_delete,
   131276,       5,"2013-07-22 15:44:20",                     ,Normal        ,
   131276,       5,"2013-07-22 15:44:20",                     ,Normal        ,

   197013,       4,"2013-07-19 20:41:35","2013-07-22 15:44:20",Pending_delete,
   197013,       4,"2013-07-19 20:41:35","2013-07-22 15:44:20",Pending_delete,
   197013,       5,"2013-07-22 15:44:20",                     ,Normal        ,
   197013,       5,"2013-07-22 15:44:20",                     ,Normal        ,
         ,        ,                     ,                     ,              ,

   131476,       1,"2013-07-19 20:41:35","2013-07-22 15:34:24",Pending_delete,
   131476,       1,"2013-07-19 20:41:35","2013-07-22 15:34:24",Pending_delete,
   131476,       2,"2013-07-22 15:34:24","2013-07-22 15:44:20",Wait          ,
   131476,       2,"2013-07-22 15:34:24","2013-07-22 15:44:20",Wait          ,
   131476,       3,"2013-07-22 15:44:20",                     ,Normal        ,
   131476,       3,"2013-07-22 15:44:20",                     ,Normal        ,
   131476,       1,"2013-07-19 20:41:35","2013-07-22 15:34:24",Pending_delete,
   131476,       1,"2013-07-19 20:41:35","2013-07-22 15:34:24",Pending_delete,
   131476,       2,"2013-07-22 15:34:24","2013-07-22 15:44:20",Normal        ,
   131476,       2,"2013-07-22 15:34:24","2013-07-22 15:44:20",Normal        ,
   131476,       3,"2013-07-22 15:44:20",                     ,Normal        ,
   131476,       3,"2013-07-22 15:44:20",                     ,Normal        ,

   197414,       1,"2013-07-19 20:41:35","2013-07-22 15:34:24",Pending_delete,
   197414,       1,"2013-07-19 20:41:35","2013-07-22 15:34:24",Pending_delete,
   197414,       2,"2013-07-22 15:34:24","2013-07-22 15:44:20",Normal        ,
   197414,       2,"2013-07-22 15:34:24","2013-07-22 15:44:20",Normal        ,
   197414,       3,"2013-07-22 15:44:20",                     ,Normal        ,
   197414,       3,"2013-07-22 15:44:20",                     ,Normal        ,
   197414,       1,"2013-07-19 20:41:35","2013-07-22 15:34:24",Pending_delete,
   197414,       1,"2013-07-19 20:41:35","2013-07-22 15:34:24",Pending_delete,
   197414,       2,"2013-07-22 15:34:24","2013-07-22 15:44:20",Wait          ,
   197414,       2,"2013-07-22 15:34:24","2013-07-22 15:44:20",Wait          ,
   197414,       3,"2013-07-22 15:44:20",                     ,Normal        ,
   197414,       3,"2013-07-22 15:44:20",                     ,Normal        ,

   197415,       1,"2013-07-19 20:41:35","2013-07-22 15:34:24",Pending_delete,
   197415,       1,"2013-07-19 20:41:35","2013-07-22 15:34:24",Pending_delete,
   197415,       2,"2013-07-22 15:34:24","2013-07-22 15:44:20",Normal        ,
   197415,       2,"2013-07-22 15:34:24","2013-07-22 15:44:20",Normal        ,
   197415,       3,"2013-07-22 15:44:20",                     ,Normal        ,
   197415,       3,"2013-07-22 15:44:20",                     ,Normal        ,

         ,        ,                     ,                     ,              ,
         ,        ,                     ,                     ,              ,

         ,        ,                     ,                     ,              ,
         ,        ,                     ,                     ,              ,
```

Figure 13-20: Output example of usage information for DB areas, tables, and indexes (columns 19 through 22)

```
Segment_type       ,MB_Used_segments#,MB_Used_pages#,MB_Total_segments#,
Base_row           ,                0,             0,                  ,
Branch_row         ,                0,             0,                  ,

Lower_page         ,                0,             0,                  ,
Upper_page         ,                0,             0,                  ,

                   ,                4,              ,                 4,

Base_row           ,                0,             0,                  ,
Branch_row         ,                0,             0,                  ,
Base_row           ,                0,             0,                  ,
Branch_row         ,                0,             0,                  ,

Lower_page         ,                0,             0,                  ,
Upper_page         ,                0,             0,                  ,
Lower_page         ,                0,             0,                  ,
Upper_page         ,                0,             0,                  ,
                   ,                4,              ,                 4,

Base_row           ,                0,             0,                  ,
Branch_row         ,                0,             0,                  ,
Base_row           ,               68,             4,                  ,
Branch_row         ,                0,             0,                  ,
Base_row           ,               36,             2,                  ,
Branch_row         ,                0,             0,                  ,
Base_row           ,                0,             0,                  ,
Branch_row         ,                0,             0,                  ,
Base_row           ,               68,             4,                  ,
Branch_row         ,                0,             0,                  ,
Base_row           ,               68,             4,                  ,
Branch_row         ,                0,             0,                  ,

Column_data        ,                0,             0,                  ,
Row_data           ,                0,             0,                  ,
Column_data        ,               68,            68,                  ,
Row_data           ,                0,             0,                  ,
Column_data        ,               36,            36,                  ,
Row_data           ,                0,             0,                  ,
Column_data        ,                0,             0,                  ,
Row_data           ,                0,             0,                  ,
Column_data        ,               68,            68,                  ,
Row_data           ,                0,             0,                  ,
Column_data        ,               68,            68,                  ,
Row_data           ,                0,             0,                  ,

Range_control      ,                0,             0,                  ,
Range_index_control,                0,             0,                  ,
Range_control      ,                4,             4,                  ,
Range_index_control,                4,             4,                  ,
Range_control      ,                4,             4,                  ,
Range_index_control,                4,             4,                  ,

                   ,             1152,              ,              1152,
                   ,              984,              ,               984,

                   ,                0,              ,              1024,
                   ,                0,              ,              1024,
```

Figure 13-21: Output example of usage information for DB areas, tables, and indexes (columns 23 through 24)



```
MB_Initialized_pages#,Storage_format
                     ,row
                     ,row

                     ,
                     ,

                    9,

                     ,row
                     ,row
                     ,row
                     ,row

                     ,
                     ,
                     ,
                     ,
                    6,

                     ,row
                     ,row
                     ,row
                     ,row
                     ,row
                     ,row
                     ,row
                     ,row
                     ,row
                     ,row
                     ,row
                     ,row

                     ,column
                     ,column
                     ,column
                     ,column
                     ,column
                     ,column
                     ,column
                     ,column
                     ,column
                     ,column
                     ,column
                     ,column

                     ,
                     ,
                     ,
                     ,
                     ,
                     ,

                 1158,
                 1158,

                 1030,
                 1030,
```

#

The display units for usage amount output items and the column titles are changed based on the specified -S option. For details about the -S option, see (3) Explanation of options in 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).

13.7.4 Guidelines for the number of rows that are output for usage information for DB areas, tables, and indexes

The number of rows that are output for usage information for DB areas, tables, and indexes depends on the range subject to collection of information. The table below provides references to the subsections that contain the formulas for determining the number of rows that will be output. Use these formulas as guidelines, because they provide only approximate numbers of rows.

Title lines (column header rows) in the output results are not included in the row counts.

Table 13-21: Ranges subject to information collection and the numbers of rows for usage information for DB areas, tables, and indexes

No.	Ranges subject to information collection for usage information for DB areas, tables, and indexes	-c option value	-n option value	Number of rows that are output (approximation)
1	Entire database (including tables and indexes in DB areas)#	`dbarea` or omitted	--	See (1) Collecting the usage information for the entire database.
2	Specific DB area (including tables and indexes in the DB area)#		*DB-area-name*	See (2) Collecting the usage information for a specific DB area.
3	All tables#	`table`	--	See (3) Collecting the usage information for all tables.
4	Specific table		*table-name*	See (4) Collecting the usage information for a specific table.
5	All indexes#	`index`	--	See (5) Collecting the usage information for all indexes.
6	Specific index		*index-name*	See (6) Collecting the usage information for a specific index.

Legend:

--: The `-n` option is omitted.

Note

The master directory DB area and the work table DB area are not processed by the `adbdbstatus` command.

#

If a DB area does not store tables or indexes, only DB area usage is output. (Table and index usage is not output.) Furthermore, the information related to any DB area that does not store tables or indexes is output for the number of rows that equal the number of DB area files that comprise the DB area.

(1) Collecting the usage information for the entire database

You can use the formula shown below to obtain the approximate number of rows that are output when usage information for the entire database (including tables and indexes in the DB areas) is collected.

Formula (in rows)

$$\mathit{table\_inf\_num} \times (1 + \mathit{chunk\_inf\_num}) \times 2 + \mathit{index\_inf\_num} \times (1 + \mathit{chunk\_inf\_num}) \times 2$$
$$+ \mathit{dbarea\_file\_inf\_num}$$

Explanation of variables

- *table_inf_num*: Number of table information items

Use the total of the numbers of DB area files in the DB areas that store tables for which usage information is to be collected. If a DB area stores multiple tables, include the number of DB area files for each table.

- *chunk_inf_num*: Number of chunk information items

  Obtain the value of the number of times the `adbimport` command with the `-b` option specified was executed minus one for each table for which usage information is to be output. Then set the sum of the obtained values in `chunk_inf_num`.

  For a table for which the `adbimport` command with the `-b` option specified has never been executed, there is no need to obtain this value.

- *index_inf_num*: Number of index information items

  Use the total of the numbers of DB area files in the DB areas that store indexes for which usage information is to be collected. If a DB area stores multiple indexes, include the number of DB area files for each index.

- *dbarea_file_inf_num*: Number of DB area information items

  Use the total of the numbers of DB area files in the data DB areas, dictionary DB area, and system-table DB area.

(2) Collecting the usage information for a specific DB area

You can use the formula shown below to obtain the approximate number of rows that are output when usage information for a specific DB area (including tables and indexes in the DB area) is collected.

Formula (in rows)

$$\mathit{table\_inf\_num} \times (1 + \mathit{chunk\_inf\_num}) \times 2 + \mathit{index\_inf\_num} \times (1 + \mathit{chunk\_inf\_num}) \times 2 \\ + \mathit{dbarea\_file\_inf\_num}$$

Explanation of variables

- *table_inf_num*: Number of table information items

  If the specified DB area stores tables, use *the total of the number of DB area files in that DB area* × *the number of tables that are stored*.

- *chunk_inf_num*: Number of chunk information items

  Obtain the value of the number of times the `adbimport` command with the `-b` option specified was executed minus one for each table for which usage information is to be output. Then set the sum of the obtained values in `chunk_inf_num`.

  For a table for which the `adbimport` command with the `-b` option specified has never been executed, there is no need to obtain this value.

- *index_inf_num*: Number of index information items

  If the specified DB area stores indexes, use *the total of the number of DB area files in that DB area* × *the number of indexes that are stored*.

- *dbarea_file_inf_num*: Number of DB area information items

  Use the number of DB area files in the specified DB area.

(3) Collecting the usage information for all tables

You can use the formula shown below to obtain the approximate number of rows that are output when usage information for all tables is collected.

Formula (in rows)

$$\mathit{table\_inf\_num} \times (1 + \mathit{chunk\_inf\_num}) \times 2 + \mathit{dbarea\_file\_inf\_num}$$

Explanation of variables

- *table_inf_num*: Number of table information items

  Use the total of the numbers of DB area files in the DB areas that store tables for which usage information is to be collected. If a DB area stores multiple tables, include the number of DB area files for each table.
- *chunk_inf_num*: Number of chunk information items

  Obtain the value of the number of times the `adbimport` command with the `-b` option specified was executed minus one for each table for which usage information is to be output. Then set the sum of the obtained values in `chunk_inf_num`.

  For a table for which the `adbimport` command with the `-b` option specified has never been executed, there is no need to obtain this value.
- *dbarea_file_inf_num*: Number of DB area information items

  Use the total of the numbers of DB area files in the DB areas that store tables for which usage information is to be collected.

(4) Collecting the usage information for a specific table

You can use the formula shown below to obtain the approximate number of rows that are output when usage information for a specific table is collected.

Formula (in rows)

$$\mathit{table\_inf\_num} \times (1 + \mathit{chunk\_inf\_num}) \times 2 + \mathit{dbarea\_filelinf\_num}$$

Explanation of variables

- *table_inf_num*: Number of table information items

  Use the number of DB area files in the DB area that stores the specified table.
- *chunk_inf_num*: Number of chunk information items

  Use the number of times the `adbimport` command with the `-b` option specified was executed for the specified table minus one.

  For a table for which the `adbimport` command with the `-b` option specified has never been executed, use `0`.
- *dbarea_file_inf_num*: Number of DB area information items

  Use the number of DB area files in the DB area that store the specified table.

(5) Collecting the usage information for all indexes

You can use the formula shown below to obtain the approximate number of rows that are output when usage information for all indexes is collected.

Formula (in rows)

$$\mathit{index\_inf\_num} \times (1 + \mathit{chunk\_inf\_num}) \times 2 + \mathit{dbarea\_file\_inf\_num}$$

Explanation of variables

- *index_inf_num*: Number of index information items

Use the total of the numbers of DB area files in the DB areas that store indexes for which usage information is to be collected. If a DB area stores multiple indexes, include the number of DB area files for each index.

- *chunk_inf_num*: Number of chunk information items

  Obtain the value of the number of times the `adbimport` command with the `-b` option specified was executed minus one for each table that has indexes for which usage information is to be collected. Then set the sum of the obtained values in `chunk_inf_num`.

  For a table for which the `adbimport` command with the `-b` option specified has never been executed, there is no need to obtain this value.

- *dbarea_file_inf_num*: Number of DB area information items

  Use the total of the numbers of DB area files in the DB areas that store indexes for which usage information is to be collected.

(6) Collecting the usage information for a specific index

You can use the formula shown below to obtain the approximate number of rows that are output when usage information for a specific index is collected.

Formula (in rows)

$$\mathit{index\_inf\_num} \times (1 + \mathit{chunk\_inf\_num}) \times 2 + \mathit{dbarea\_file\_inf\_num}$$

Explanation of variables

- *index_inf_num*: Number of index information items

  Use the number of DB area files in the DB area that stores the specified index.

- *chunk_inf_num*: Number of chunk information items

  Use the number of times the `adbimport` command with the `-b` option specified was executed for the table for which the specified index was defined minus one. For a table for which the `adbimport` command with the `-b` option specified has never been executed, use `0`.

- *dbarea_file_inf_num*: Number of DB area information items

  Use the total of the numbers of DB area files in the DB area that store the specified index.

13.8 Items that are output in the usage information of archived chunks

This section explains in detail about items that are output in the usage information of archived chunks.

When you want to output usage information about archived chunks, specify the following three options:

- `-d` option
  Specify `used` in the `-d` option.
- `-c` option
  Specify `archivechunk` in the `-c` option.
- `-n` option
  Specify the table name of the archivable multi-chunk table in the `-n` option.

For details about the specification format and options of the `adbdbstatus` command, see 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).

13.8.1 Number of rows that are output, and order of output, for the usage information of archived chunks

One row of usage information of archived chunks is output per archive file. In addition, information about all archive files corresponding to each chunk is output on a per-chunk basis. Accordingly, you can get the number of rows of archived chunks' usage information that are output by using the following formula.

- **Formula for the number of rows output from the usage information of archived chunks**

```
Number-of-rows-output-in-archived-chunks'-usage-information =
  Number-of-archived-chunks + number-of-archive-files-corresponding-to-each-archived-chunk
```

At this time, the column title is not included in the number of rows.

Note that there is no particular output order for these rows (information is not necessarily output in increasing or decreasing order of chunk ID).

13.8.2 List of items that are output in the usage information of archived chunks

The following table lists the titles and output content of items (columns) that are output in the usage information of archived chunks. Each column's information is output in the order of the item numbers in the following table. For the rules for outputting usage information of archived chunks, see 13.12 Output rules for the adbdbstatus command.

Table 13-22: Columns output as usage information for archived chunks

No.	Column title	Information that is output	Enclosing character
1	`Type`	Type of output information `Used_chunk`: Usage information of archived chunk	N
2	`Timestamp`	Time stamp of `adbdbstatus` command execution The date and time the `adbdbstatus` command was executed is output in the following format: `"`*YYYY*`-`*MM*`-`*DD*Δ*hh*`:`*mm*`:`*ss*`"`#	Y
3	`DBarea_name`	DB area name	Y
4	`DBarea_ID`	DB area ID	N
5	`Rsv`	--	--
6	`Object_type`	Schema object type `table`: Table	N
7	`Schema_name`	Table schema name	Y
8	`Object_identifier`	Table identifier	Y
9	`Object_ID`	Table ID	N
10	`Chunk_ID`	Chunk ID	N
11	`Chunk_status`	Chunk status • `Normal`: Normal status • `Wait`: Wait status • `Pending_delete`: Deletion-pending status	N
12	`Range_min`	Minimum value of the archive range column Information is output as the data type in the archive range column. Information is only output if there is an archive file that corresponds to a chunk (the data is stored in the chunk) whose ID is output in the `Chunk_ID` column. • If the archive file name is output in the `Archive_file_name` column The minimum value of the corresponding archive file's archive range column is output. • If nothing is output in the `Archive_file_name` column The minimum value of the archive range columns in all archive files that correspond to chunks whose ID is output in the `Chunk_ID` column is output.	Y
13	`Range_min_raw`	Minimum value (hexadecimal) of the archive range column Information is output as hexadecimal-format binary data only when the data type in the archive range column is `CHARACTER`. Information is only output if there is an archive file that corresponds to a chunk (the data is stored in the chunk) whose ID is output in the `Chunk_ID` column.	Y
14	`Range_max`	Maximum value of the archive range column Information is output as the data type in the archive range column. Information is only output if there is an archive file that corresponds to a chunk (the data is stored in the chunk) whose ID is output in the `Chunk_ID` column. • If the archive file name is output in the `Archive_file_name` column The maximum value of the corresponding archive file's archive range column is output. • If nothing is output in the `Archive_file_name` column	Y

No.	Column title	Information that is output	Enclosing character
		The maximum value of the archive range columns in all archive files that correspond to chunks whose ID is output in the `Chunk_ID` column is output.	
15	`Range_max_raw`	Maximum value (hexadecimal) of the archive range column Information is output as hexadecimal-format binary data only when the data type in the archive range column is `CHARACTER`. Information is only output if there is an archive file that corresponds to a chunk (the data is stored in the chunk) whose ID is output in the `Chunk_ID` column.	Y
16	`Archive_file_name`	Archive file name The file path of the archive file corresponding to the chunk is output. If nothing is output to this column, the total size of all archive files corresponding to chunks whose ID is output in the `Chunk_ID` column, is output to the `Archive_file_size` column.	Y
17	`Archive_file_size`	Archive file size (units: bytes) • If the archive file name is output in the `Archive_file_name` column The size of the corresponding archive file is output. • If nothing is output in the `Archive_file_name` column The total size of all archive files corresponding to chunks whose ID is output in the `Chunk_ID` column is output.	N
18	`Rsv`	--	--

Legend:

--: No value is output.

Y: The output value is enclosed in enclosing characters.

N: The output value is not enclosed in enclosing characters.

#

The year (*YYYY*) is output in 4 digits. The month (*MM*), date (*DD*), hours (*hh*), minutes (*mm*), and seconds (*ss*) are each output in 2 digits. If there are not enough digits in a value, the left side of the value is padded with zeros (`0`). Δ represents a halfwidth space.

13.8.3 Examples of outputting the usage information of archived chunks

The following are examples of outputting the usage information of archived chunks. Note that the actual output results are shown here in parts due to their horizontal length.

Figure 13-22: Examples of outputting the usage information of archived chunks (columns 1-7)

```
Type      ,Timestamp            ,DBarea_name,DBarea_ID,Rsv,Object_type,Schema_name,
Used_chunk,"2015-10-19 10:38:24","ADBU00015",       26,   ,table      ,"ADBUSER"  ,
Used_chunk,"2015-10-19 10:38:24","ADBU00015",       26,   ,table      ,"ADBUSER"  ,
Used_chunk,"2015-10-19 10:38:24","ADBU00015",       26,   ,table      ,"ADBUSER"  ,
Used_chunk,"2015-10-19 10:38:24","ADBU00015",       26,   ,table      ,"ADBUSER"  ,
Used_chunk,"2015-10-19 10:38:24","ADBU00015",       26,   ,table      ,"ADBUSER"  ,
Used_chunk,"2015-10-19 10:38:24","ADBU00015",       26,   ,table      ,"ADBUSER"  ,
Used_chunk,"2015-10-19 10:38:24","ADBU00015",       26,   ,table      ,"ADBUSER"  ,
Used_chunk,"2015-10-19 10:38:24","ADBU00015",       26,   ,table      ,"ADBUSER"  ,
Used_chunk,"2015-10-19 10:38:24","ADBU00015",       26,   ,table      ,"ADBUSER"  ,
Used_chunk,"2015-10-19 10:38:24","ADBU00015",       26,   ,table      ,"ADBUSER"  ,
Used_chunk,"2015-10-19 10:38:24","ADBU00015",       26,   ,table      ,"ADBUSER"  ,
Used_chunk,"2016-04-20 14:04:32","ADBU00016",       27,   ,table      ,"ADBUSER"  ,
Used_chunk,"2016-04-20 14:04:32","ADBU00016",       27,   ,table      ,"ADBUSER"  ,
```

Figure 13-23: Examples of outputting the usage information of archived chunks (columns 8-13)

```
Object_identifier,Object_ID,Chunk_ID,Chunk_status,Range_min   ,Range_min_raw     ,
"TBL015"         ,   131475,       3,Normal      ,"2007-03-01",                  ,
"TBL015"         ,   131475,       3,Normal      ,"2007-03-01",                  ,
"TBL015"         ,   131475,       3,Normal      ,"2007-03-01",                  ,
"TBL015"         ,   131475,       4,Normal      ,"2007-04-01",                  ,
"TBL015"         ,   131475,       4,Normal      ,"2007-04-01",                  ,
"TBL015"         ,   131475,       4,Normal      ,"2007-04-01",                  ,
"TBL015"         ,   131475,       4,Normal      ,"2007-04-01",                  ,
"TBL015"         ,   131475,       4,Normal      ,"2007-04-01",                  ,
"TBL015"         ,   131475,     108,Wait        ,"2015-12-01",                  ,
"TBL015"         ,   131475,     108,Wait        ,"2015-12-01",                  ,
"TBL015"         ,   131475,     108,Wait        ,"2015-12-01",                  ,
"TBL016"         ,   131477,       1,Normal      ,"20160401"  ,"3230313630343031",
"TBL016"         ,   131477,       1,Normal      ,"20160401"  ,"3230313630343031",
```

Figure 13-24: Examples of outputting the usage information of archived chunks (columns 14-16)

```
Range_max   ,Range_max_raw     ,Archive_file_name                                         ,
"2007-03-31",                  ,"/dev/ode/archive/ADBARCTBL_131475/ADBARCSER_8/ADBARC_1"  ,
"2007-03-31",                  ,"/dev/ode/archive/ADBARCTBL_131475/ADBARCSER_8/ADBARC_2"  ,
"2007-03-31",                  ,                                                          ,
"2007-04-30",                  ,"/dev/ode/archive/ADBARCTBL_131475/ADBARCSER_9/ADBARC_1"  ,
"2007-04-30",                  ,"/dev/ode/archive/ADBARCTBL_131475/ADBARCSER_9/ADBARC_2"  ,
"2007-04-30",                  ,"/dev/ode/archive/ADBARCTBL_131475/ADBARCSER_9/ADBARC_4"  ,
"2007-04-30",                  ,"/dev/ode/archive/ADBARCTBL_131475/ADBARCSER_9/ADBARC_5"  ,
"2007-04-30",                  ,                                                          ,
"2015-12-31",                  ,"/dev/ode/archive/ADBARCTBL_131475/ADBARCSER_123/ADBARC_2",
"2015-12-31",                  ,"/dev/ode/archive/ADBARCTBL_131475/ADBARCSER_123/ADBARC_3",
"2015-12-31",                  ,                                                          ,
"20160430"  ,"3230313630343330","/dev/ode/archive/ADBARCTBL_131477/ADBARCSER_200/ADBARC_1",
"20160430"  ,"3230313630343330",                                                          ,
```

Figure 13-25: Examples of outputting the usage information of archived chunks (columns 17-18)

```
Archive_file_size,Rsv
           411293,
           463508,
           874081,
           445598,
           445589,
           458501,
           447688,
          1797376,
           459886,
           415548,
           875434,
           411293,
           411293,
```

13.9 Items that are output as information about the need for reorganization

This section provides details of the items that are output as information about the need for reorganization.

To output information about the need for reorganization, specify the following two options:

- `-d` option
  Specify `reorginfo` in the `-d` option.
- `-n` option
  For the `-n` option, specify the table name of the table for which you want to collect information.

You can use the `-c` option to specify a particular chunk for which information is to be collected.

For details about the specification format and options of the `adbdbstatus` command, see 13.2.2 Explanation of the specification format and options (outputting information about the need for reorganization of column store tables).

13.9.1 Number of rows that are output, and order of output, for the information about the need for reorganization

Information about the need for reorganization is output for each chunk for which the information is collected. Accordingly, the number of rows that are output in the information about the need for reorganization is the same as the number of chunks for which the information is collected. Note that information for deletion-pending chunks is not collected. The column title is not included in the number of rows.

Rows are output in ascending order of chunk IDs.

13.9.2 List of items that are output as information about the need for reorganization

The following table lists the titles of the items (columns) that are output as the information about the need for reorganization, and the content that is output. The columns are output in the order listed in the table. For details about the columns that are output in the information about the need for reorganization, see 13.12 Output rules for the adbdbstatus command.

Table 13-23: Columns output in the information about the need for reorganization

No.	Column title	Information that is output	Enclosing character
1	`Type`	Type of information that is output: `Reorg_info`: Information about the need for reorganization	N
2	`Timestamp`	Timestamp when the `adbdbstatus` command was executed: The date and time the `adbdbstatus` was executed are output in the following format: `"`*YYYY*`-`*MM*`-`*DD*Δ*hh*`:`*mm*`:`*ss*`"`#1	Y

No.	Column title	Information that is output	Enclosing character
3	DBarea_name	Name of the DB area where the table is stored	Y
4	DBarea_ID	DB area ID of the DB area where the table is stored	N
5	Schema_name	Table schema name	Y
6	Table_identifier	Table identifier	Y
7	Table_ID	Table ID	N
8	Chunk_ID	Chunk ID	N
9	Column_data_num	Number of rows in column store format The total of the following values is output: • Number of rows imported into a column-data segment by the adbimport command • Number of rows converted in column store format by the updated-row columnizing facility	N
10	Base_row_pages#2	Number of basic row pages contained in the row-data segment#3, #4	N
11	Branch_row_pages#2	Number of branch row pages contained in the row-data segment#3	N
12	Invalid_row_information_pages#2	Number of invalid row information pages contained in the row-data segment#3	N
13	Reorganization_necessity_index	Troubleshooting information	N
14	Reorganization_necessity	Necessity of reorganization • Recommended: Reorganization is recommended. • Not_recommended: Reorganization is not recommended. If performance of chunk retrieval might have decreased, Recommended is output. In this case, the KFAA60200-W message that recommends reorganization is also output for each chunk. For details about the reasons why a column store table must be reorganized and how to do so, see the following in the *HADB Setup and Operation Guide*: *Checking whether a single-chunk table needs to be reorganized* or *Checking whether a multi-chunk table needs to be reorganized*.	N

Legend:

Y: The output value is enclosed in a pair of enclosing characters.

N: The output value is not enclosed in the enclosing character.

#1

The year (*YYYY*) is output in 4 digits. The month (*MM*), date (*DD*), hours (*hh*), minutes (*mm*), and seconds (*ss*) are each output in 2 digits. If there are not enough digits, leading zeros are added. Δ indicates a single-byte space.

#2

If the display units for information about the need for reorganization is changed by using the -S option, the column titles for the output items are also changed to match the selected display units. For details about the display units and column titles, see Table 13-14: Display units and column titles for the output items for the necessity of reorganization in (2) Explanation of options in 13.2.2 Explanation of the specification format and options (outputting information about the need for reorganization of column store tables).

#3

The -S option enables you to change the display units for information about the need for reorganization to kilobytes, megabytes, or gigabytes.

#4

If basic row pages are made reusable by the updated-row columnizing facility, the number of pages that have become reusable is excluded from the displayed `Base_row_pages` value (number of basic row pages contained in the row-data segment).

When the pages that have become reusable are reused and filled to capacity, the number of reused pages is added again to the displayed `Base_row_pages` value (number of basic row pages contained in the row-data segment).

For details about the updated-row columnizing facility, see *Using the updated-row columnizing facility (maintaining the retrieval performance for column store tables)* in the *HADB Setup and Operation Guide*.

13.9.3 Output example of information about the need for reorganization

The following shows an output example of information about the need for reorganization. Because the rows in the actual output results are wide, they are shown in sections in this manual.

Figure 13-26: Output example of information about the need for reorganization (columns 1 to 9)

```
Type      ,Timestamp            ,DBarea_name  ,DBarea_ID,Schema_name,Table_identifier,Table_ID,Chunk_ID,Column_data_num,
Reorg_info,"2018-07-25 15:40:25","DBAREATBL01",       11,"ADBUSER01","TABLE01"       ,  131473,       1,          34522,
Reorg_info,"2018-07-25 15:40:25","DBAREATBL01",       11,"ADBUSER01","TABLE01"       ,  131473,       2,          84907,
Reorg_info,"2018-07-25 15:40:25","DBAREATBL01",       11,"ADBUSER01","TABLE01"       ,  131473,       3,          55589,
Reorg_info,"2018-07-25 15:40:25","DBAREATBL01",       11,"ADBUSER01","TABLE01"       ,  131473,       4,          74696,
Reorg_info,"2018-07-25 15:40:25","DBAREATBL01",       11,"ADBUSER01","TABLE01"       ,  131473,       5,          15230,
```

Figure 13-27: Output example of information about the need for reorganization (columns 10 to 14)

```
Base_row_pages#,Branch_row_pages#,Invalid_row_information_pages#,Reorganization_necessity_index,Reorganization_necessity
              0,                2,                             0,                           100,Not_recommended
              0,                0,                             0,                           100,Not_recommended
            100,                5,                            10,                           113,Recommended
              0,                7,                             0,                           100,Not_recommended
              0,                0,                             6,                           100,Not_recommended
```

#

The display units for output items and the column titles are changed based on the specified -S option. For details about the -S option, see (2) Explanation of options in 13.2.2 Explanation of the specification format and options (outputting information about the need for reorganization of column store tables).

13.10 Examples

This section explains by way of examples how to use the adbdbstatus command to check the database status.

In the output examples, the delimiters and some of the columns are omitted for readability.

13.10.1 Checking the usage of the entire database (output of summary information for DB areas)

This example outputs summary information for the DB areas and checks the following:

- Usage rate of each DB area
- Usage (in megabytes) of each DB area

■ adbdbstatus command that is executed

```
adbdbstatus -S M
```

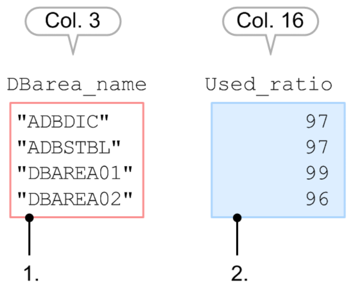
Explanation:

- If the adbdbstatus command is executed with the -d, -c, and -n options all omitted, summary information for all DB areas is output.
- Specifies megabytes as the units for displaying usage amounts.

The following shows how to check the output results.

(1) Checking the usage rate of each DB area

■ Output example



```
DBarea_name     Used_ratio
"ADBDIC"                97
"ADBSTBL"               97
"DBAREA01"              99
"DBAREA02"              96
```

To check the usage rate of each DB area:

1. Check the DB area names.
   Check the DB area names output in the column headed DBarea_name (column 3 in the output results).
2. Check the segment usage rate of each DB area.
   Check the DB area segment usage rates (%) that are output in the column headed Used_ratio (column 16 in the output results). If a row's segment usage rate is close to 100%, consider taking the following actions according to the type of DB area file:
   - If the DB area files are block special files

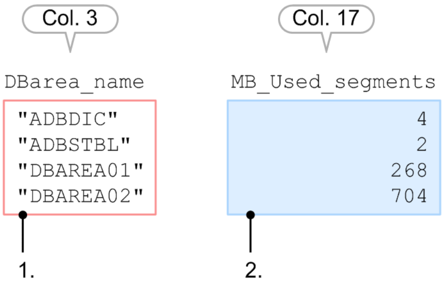
If you continue to store data, the DB area files might become full. Consider providing more free space on the disk where the block special file is stored based on the explanation in *When a free space shortage is caused by an increase in the size of the DB area files* in the *HADB Setup and Operation Guide*.

- If the DB area files are regular files

  If you continue to store data, the DB area will be extended automatically. When a DB area is extended automatically, its capacity increases, but the efficiency of update processing will decrease when the DB area's files are expanded. If a DB area automatic extension has failed, see *When a free space shortage is caused by failed DB area automatic extension* in the *HADB Setup and Operation Guide*.

(2) Checking the usage amount (in megabytes) of each DB area

■ Output example



```
DBarea_name      MB_Used_segments
"ADBDIC"                        4
"ADBSTBL"                       2
"DBAREA01"                    268
"DBAREA02"                    704
```

To check the usage amount (in megabytes) of each DB area:

1. Check the DB area names.
   Check the DB area names output in the column headed `DBarea_name` (column 3 in the output results).
2. Check the usage amount (in megabytes) of each DB area.
   Obtain the value of the `MB_Used_segments` column (col. 17 in the output results) for each row (each DB area).

13.10.2 Checking the usage of a specific DB area (output of summary information for a DB area)

This example outputs summary information for DB area `DBAREA01` and checks the following:

- Usage rate of DB area `DBAREA01`
- Usage amount (in megabytes) of DB area `DBAREA01`

■ `adbdbstatus` command that is executed

```
adbdbstatus -c dbarea           ...1
            -n DBAREA01         ...2
            -S M                ...3
```

Explanation:

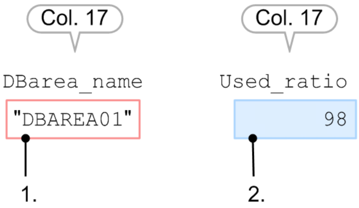
1. Specifies DB area as the range of summary information collection.
2. Specifies the name of the DB area (`DBAREA01`) for which summary information is to be collected.
3. Specifies megabytes as the units for displaying usage amounts.

When the `adbdbstatus` command is executed, summary information for DB area `DBAREA01` is output.

The following shows how to check the output results.

(1) Checking the usage rate of a DB area

■ Output example



```
DBarea_name    Used_ratio
"DBAREA01"             98
```

To check the usage rate of DB area `DBAREA01`:

1. Check the DB area name.

   Verify that DB area name `DBAREA01` is displayed in the column headed `DBarea_name` (column 3 in the output results).

2. Check the usage rate of the DB area.

   The segment usage rate (%) of the DB area is output in the column headed `Used_ratio` (column 16 in the output results). If the segment usage rate is close to 100%, consider taking the following actions according to the type of DB area file:

   - If the DB area files are block special files

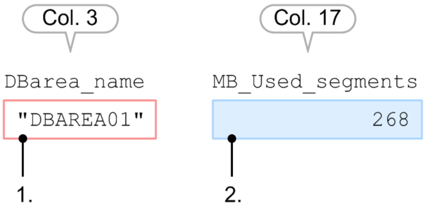
     If you continue to store data, the DB area files might become full. Consider providing more free space on the disk where the block special file is stored based on the explanation in *When a free space shortage is caused by an increase in the size of the DB area files* in the *HADB Setup and Operation Guide*.

   - If the DB area files are regular files

     If you continue to store data, the DB area will be extended automatically. When a DB area is extended automatically, its capacity increases, but the efficiency of update processing will decrease when the DB area's files are expanded. If a DB area automatic extension has failed, see *When a free space shortage is caused by failed DB area automatic extension* in the *HADB Setup and Operation Guide*.

(2) Checking the usage amount (in megabytes) of a DB area

■ Output example



```
DBarea_name   MB_Used_segments
"DBAREA01"                 268
```

To check the usage amount (in megabytes) of a DB area:

1. Check the DB area name.

   Verify that DB area name `DBAREA01` that you specified in the `adbdbstatus` command is displayed in the column headed `DBarea_name` (column 3 in the output results).

2. Check the usage amount (in megabytes) of the DB area.

Obtain the value of the MB_Used_segments column (col. 17 in the output results).

13.10.3 Checking the usage of a specific table (output of summary information for a table)

This example outputs summary information for a specific table (ADBUSER01.TABLE01) and checks the following:

- Usage amount (in kilobytes) of table ADBUSER01.TABLE01

■ adbdbstatus **command that is executed**

```
adbdbstatus -c table              ...1
            -n ADBUSER01.TABLE01  ...2
            -S K                  ...3
            --shared-lock         ...4
```

Explanation:

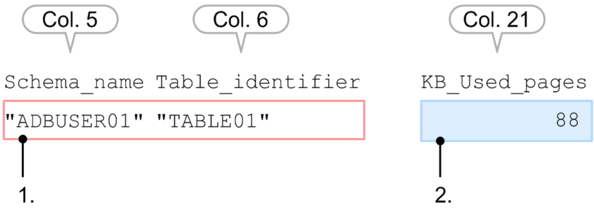
1. Specifies table as the range of summary information collection.
2. Specifies the name of the table (ADBUSER01.TABLE01) for which summary information is to be collected.
3. Specifies kilobytes as the units for displaying usage amounts.
4. Specifies the --shared-lock option so that information can be output even while another SQL statement or command is updating the database.
   Note that if the --shared-lock option is specified, the usage of deletion-pending chunks in a system table (base table) cannot be output.

When the adbdbstatus command is executed, summary information for table ADBUSER01.TABLE01 is output.

The following shows how to check the output results.

(1) Checking the usage amount (in kilobytes) of a table

■ Output example



```
Schema_name Table_identifier     KB_Used_pages
"ADBUSER01" "TABLE01"                       88
```

To check the usage amount (in kilobytes) of table ADBUSER01.TABLE01:

1. Check the schema name and table identifier of the table.
   Verify that the schema name (ADBUSER01) of the table is output in the column headed Schema_name (column 5 in the output results) and the table identifier (TABLE01) is output in the column headed Table_identifier (column 6 in the output results).
2. Check the usage amount (in kilobytes) of the table.
   Obtain the value of the KB_Used_pages column (col. 21 in the output results).

13.10.4 Checking the status of a specific table (output of summary information for a table)

This example outputs summary information for a specific table (ADBUSER01.TABLE01) and checks the following:

- Whether table ADBUSER01.TABLE01 is non-updatable
- Whether the command executed on table ADBUSER01.TABLE01 must be re-executed.

■ adbdbstatus command that is executed

```
adbdbstatus -c table              ...1
            -n ADBUSER01.TABLE01  ...2
```

Explanation:

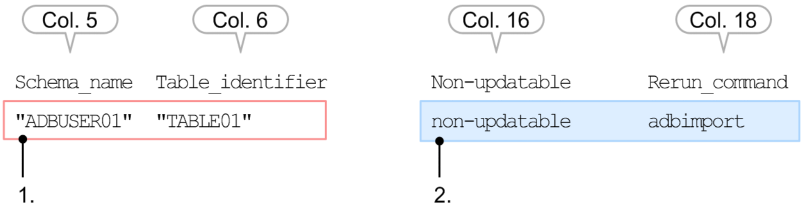
1. Specifies table as the range of summary information collection.
2. Specifies the name of the table (ADBUSER01.TABLE01) for which summary information is to be collected.

When the adbdbstatus command is executed, summary information for table ADBUSER01.TABLE01 is output.

The following shows how to check the output results.

(1) Determining whether the table is non-updatable

■ Output example



```
Schema_name  Table_identifier     Non-updatable     Rerun_command
"ADBUSER01"  "TABLE01"            non-updatable     adbimport
```

To determine whether table ADBUSER01.TABLE01 is non-updatable:

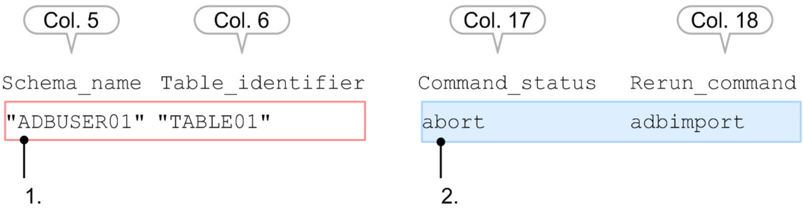
1. Check the schema name and table identifier of the table.
   Verify that the schema name (ADBUSER01) of the table is output in the column headed Schema_name (column 5 in the output results) and the table identifier (TABLE01) is output in the column headed Table_identifier (column 6 in the output results).
2. Determine whether the table is non-updatable.
   If the table is non-updatable, non-updatable is displayed in the column headed Non-updatable (column 16 in the output results). To release the table from non-updatable status, re-execute the command displayed in the Rerun_command column (column 18 in the output results).
   In this example, to release table ADBUSER01.TABLE01 from non-updatable status, the adbimport command must be re-executed.

Note

If the --shared-lock option is specified, no_check is displayed in the column headed Non-updatable (column 16 in the output results). Information about whether the table is non-updatable is not output. If you want to determine whether the table is non-updatable, do not specify the `--shared-lock` option.

(2) Checking the termination status of the command executed on the table

■ Output example



```
Schema_name  Table_identifier   Command_status   Rerun_command
"ADBUSER01" "TABLE01"           abort            adbimport
```

To check the termination status of the command executed on table `ADBUSER01.TABLE01`:

1. Check the schema name and table identifier of the table.
   Verify that the schema name (`ADBUSER01`) of the table is output in the column headed `Schema_name` (column 5 in the output results) and the table identifier (`TABLE01`) is output in the column headed `Table_identifier` (column 6 in the output results).
2. Check the termination status of the command executed on the table.
   If the command must be re-executed on the table, `abort` is output in the column headed `Command_status` (column 17 in the output results). If `abort` is output, re-execute the command displayed in the column headed `Rerun_command` (column 18 in the output results).
   In this example, the `adbimport` command must be re-executed for table `ADBUSER01.TABLE01`.

Note

If the `--shared-lock` option is specified, `no_check` is displayed in the column headed `Command_status` (column 17 in the output results). The command termination status is not output. If you want to check the termination status of the command executed on the table, do not specify the `--shared-lock` option.

13.10.5 Checking the usage of a specific index (output of summary information for an index)

This example outputs summary information for a specific index (`ADBUSER01.INDEX01`) and checks the following:

- Usage amount (in kilobytes) of index `ADBUSER01.INDEX01`

■ `adbdbstatus` **command that is executed**

```
adbdbstatus -c index                ...1
            -n ADBUSER01.INDEX01    ...2
            -S K                    ...3
            --shared-lock           ...4
```

Explanation:

1. Specifies index as the range of summary information collection.
2. Specifies the name of the index (ADBUSER01.INDEX01) for which summary information is to be collected.
3. Specifies kilobytes as the units for displaying usage amounts.
4. Specifies the --shared-lock option so that information can be output even while another SQL statement or command is updating the database.

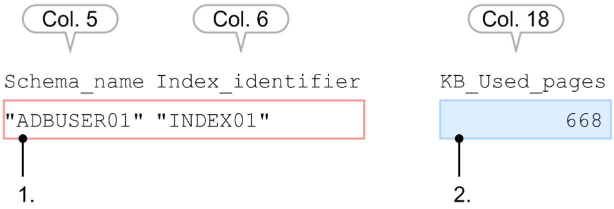
   Note that if the --shared-lock option is specified, the usage of deletion-pending chunks for the indexes defined for a system table (base table) cannot be output.

When the adbdbstatus command is executed, summary information for index ADBUSER01.INDEX01 is output.

The following shows how to check the output results.

(1) Checking the usage amount (in kilobytes) of an index

■ Output example



```
Schema_name Index_identifier         KB_Used_pages
"ADBUSER01" "INDEX01"                          668
```

To check the amount of index ADBUSER01.INDEX01 that is being used:

1. Check the schema name and index identifier of the index.

   Verify that the schema name (ADBUSER01) of the index is output in the column headed Schema_name (column 5 in the output results) and the index identifier (INDEX01) is output in the column headed Index_identifier (column 6 in the output results).
2. Check the usage amount (in kilobytes) of the index.

   Obtain the value of the KB_Used_pages column (col. 18 in the output results).

13.10.6 Checking the status of a specific index (output of summary information for an index)

This example outputs summary information for a specific index (ADBUSER01.INDEX01) and checks the following:

- Whether index ADBUSER01.INDEX01 is in unfinished status
- Whether a uniqueness constraint violation has occurred on index ADBUSER01.INDEX01

■ adbdbstatus command that is executed

```
adbdbstatus -c index              ...1
            -n ADBUSER01.INDEX01  ...2
```

Explanation:

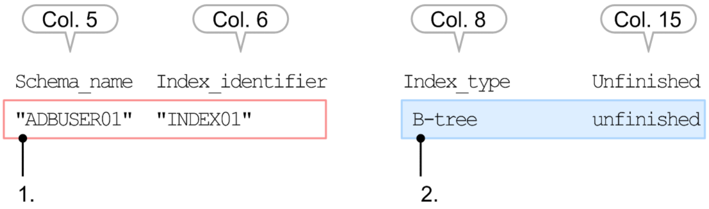
1. Specifies index as the range of summary information collection.
2. Specifies the name of the index (ADBUSER01.INDEX01) for which summary information is to be collected.

When the adbdbstatus command is executed, summary information for index ADBUSER01.INDEX01 is output.

The following shows how to check the output results.

(1) Determining whether the index is in unfinished status

■ Output example



```
Schema_name  Index_identifier     Index_type      Unfinished
"ADBUSER01"  "INDEX01"            B-tree          unfinished
```

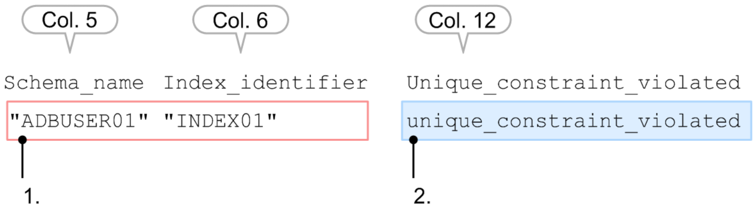
To determine whether index ADBUSER01.INDEX01 is in unfinished status:

1. Check the schema name and index identifier of the index.
   Verify that the schema name (ADBUSER01) of the index is output in the column headed Schema_name (column 5 in the output results) and the index identifier (INDEX01) is output in the column headed Index_identifier (column 6 in the output results).
2. Determine whether the index is in unfinished status.
   If the index is in unfinished status, unfinished is displayed in the column headed Unfinished (column 15 in the output results). To determine the type of the index that is in unfinished status, check the column headed Index_type (column 8 in the output results).
   For details about how to release B-tree indexes from unfinished status, see *Steps to take when unfinished status is applied to a B-tree index* in the *HADB Setup and Operation Guide*.
   For details about how to release text indexes from unfinished status, see *Steps to take when unfinished status is applied to a text index* in the *HADB Setup and Operation Guide*.
   For details about how to release range indexes from unfinished status, see *Steps to take when unfinished status is applied to a range index* in the *HADB Setup and Operation Guide*.

> **Note**
>
> If the --shared-lock option is specified, no_check is displayed in the column headed Unfinished (column 15 in the output results). Information about whether the index is in unfinished status is not output. If you want to determine whether the index is in unfinished status, do not specify the --shared-lock option.

(2) Determining whether a uniqueness constraint violation occurred on the index

■ Output example



```
Schema_name  Index_identifier    Unique_constraint_violated
"ADBUSER01" "INDEX01"            unique_constraint_violated
```

To determine whether a uniqueness constraint violation occurred on index `ADBUSER01.INDEX01`:

1. Check the schema name and index identifier of the index.
   Verify that the schema name (`ADBUSER01`) of the index is output in the column headed `Schema_name` (column 5 in the output results) and the index identifier (`INDEX01`) is output in the column headed `Index_identifier` (column 6 in the output results).
2. Check whether a uniqueness constraint violation occurred on the index.
   If a uniqueness constraint violation occurred on the index, `unique_constraint_violated` is displayed in the column headed `Unique_constraint_violated` (column 12 in the output results).
   For details about how to release the index from uniqueness constraint violation, see *Steps to take when the uniqueness constraint is violated (when the KFAA61205-W message is output)* in the *HADB Setup and Operation Guide*.

> **Note**
>
> If the `--shared-lock` option is specified, `no_check` is displayed in the column headed `Unique_constraint_violated` (column 12 in the output results). Information about uniqueness constraint violation of the index is not output. If you want to determine whether a uniqueness constraint violation occurred on the index, do not specify the `--shared-lock` option.

13.10.7 Finding the total size of all archive files corresponding to an archived chunk created in a specific table (outputting summary information of an archived chunk)

The total size (units: bytes) of all archive files corresponding to the archived chunk created in a specific table (`ADBUSER01.TABLE01`) is found by outputting summary information from the archived chunk.

- **Example of executing the `adbdbstatus` command**

```
adbdbstatus -c archivechunk          ...1
            -n ADBUSER01.TABLE01     ...2
            -r 2015/01/01-2015/12/31 ...3
```

Explanation:

1. Archived chunks are specified as a range from which to obtain summary information.
2. The table name (`ADBUSER01.TABLE01`) from which to obtain summary information about archived chunks is specified. The archivable multi-chunk table must be specified at this time.

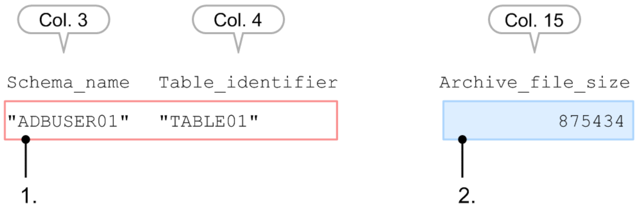
3. A range of dates (`2015/01/01-2015/12/31`) for which to output chunk information is specified based on the values of the archive range columns corresponding to the chunks.

   The data type in the archive range column is `DATE`.

When the `adbdbstatus` command is executed, the summary information of archived chunks created in the table `ADBUSER01.TABLE01` is output.

In the following procedure, the total size (units: bytes) of all archive files corresponding to the archived chunks is found based on the output results.

■ Output example



```
Schema_name  Table_identifier        Archive_file_size
"ADBUSER01"  "TABLE01"                          875434
```

Procedure

1. Check the table schema name and table identifier.

   Verify that the schema name (`ADBUSER01`) of the table is output in the column headed `Schema_name` (column 3 in the output results) and the table identifier (`TABLE01`) is output in the column headed `Table_identifier` (column 4 in the output results).

2. Find the total size (units: bytes) of all archive files corresponding to the archived chunks.

   Find the value of the `Archive_file_size` column (15th column in the output results).

13.10.8 Checking the usage rate and usage of a DB area file in a specific DB area (output of usage information for DB areas, tables, and indexes)

This example outputs usage information for a specific DB area (`DBAREA01`) and determines the following:

- Usage rate of the DB area files in `DBAREA01`
- Usage amount (in megabytes) of `DBAREA01` for each DB area file
- Usage amount (in megabytes) of a table stored in `DBAREA01` for each DB area file
- Usage amount (in megabytes) of an index stored in `DBAREA01` for each DB area file

■ Executing the `adbdbstatus` command

```
adbdbstatus -d used                   ...1
            -c dbarea                 ...2
            -n DBAREA01               ...3
            -S M                      ...4
            --shared-lock             ...5
```

Explanation:

1. Specifies usage information as the information to be output.

2. Specifies DB area as the range of usage information collection.
3. Specifies the name of the DB area (`DBAREA01`) for which usage information is to be collected.
4. Specifies megabytes as the units for displaying usage amounts.
5. Specifies the `--shared-lock` option so that information can be output even while another SQL statement or command is updating the database.

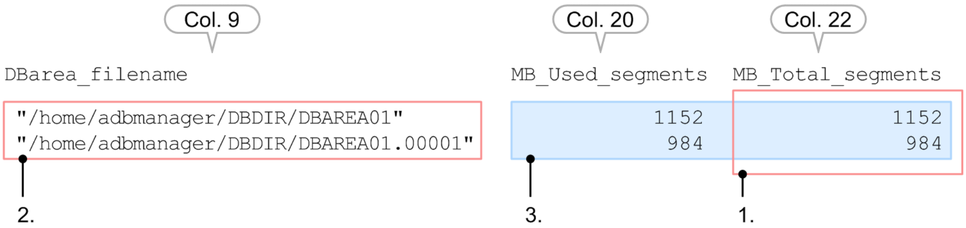
   Note that if the `--shared-lock` option is specified, the usage of deletion-pending chunks in a system table (base table) and indexes defined for the system table (base table) cannot be output.

When the `adbdbstatus` command is executed, the usage information for DB area `DBAREA01` is output.

The following shows how to check the output results.

(1) Checking the usage rate of a DB area in the DB area file

■ Output example



```
DBarea_filename                                 MB_Used_segments  MB_Total_segments

 "/home/adbmanager/DBDIR/DBAREA01"                          1152               1152
 "/home/adbmanager/DBDIR/DBAREA01.00001"                     984                984
```

To check the usage rate of the DB area files in `DBAREA01`:

1. Find the rows required for the calculation of the usage rate.

   To check the usage rates of the DB area files of the DB area, find the rows that contain a value in the column headed `MB_Total_segments` (column 22 in the output results).
2. Verify the DB area file names.

   Verify the DB area file names displayed in the column headed `DBarea_filename` (column 9 in the output results).
3. Obtain the usage rate for each DB area file in the DB area.

   On each row (each DB area file), check the values of the columns headed `MB_Used_segments` (column 20 in the output results) and `MB_Total_segments` (column 22 in the output results). Set these values in the following formula to obtain the segment usage rate for each DB area file.

   Segment usage rate for each DB area file (%)

   MB_Used_segments ÷ *MB_Total_segments* × 100

   Explanation of variables

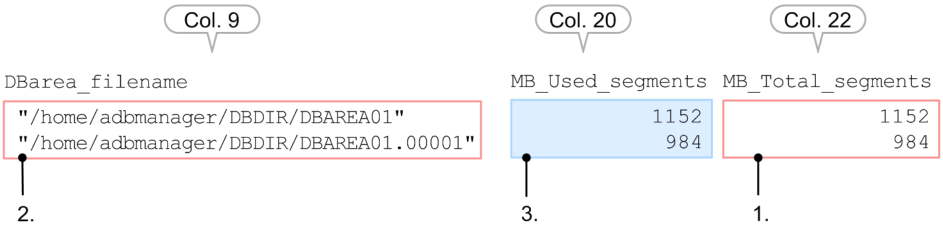
   MB_Used_segments: Value of the `MB_Used_segments` column (in megabytes)

   MB_Total_segments: Value of the `MB_Total_segments` column (in megabytes)

   In this example, the usage rate of DB area file `/home/adbmanager/DBDIR/DBAREA01` is 1,152 megabytes ÷ 1,152 megabytes × 100 = 100%, and the usage rage of DB area file `/home/adbmanager/DBDIR/DBAREA01.00001` is 984 megabytes ÷ 984 megabytes × 100 = 100%.

(2) Checking the usage amount (in megabytes) of a DB area for each DB area file

■ Output example



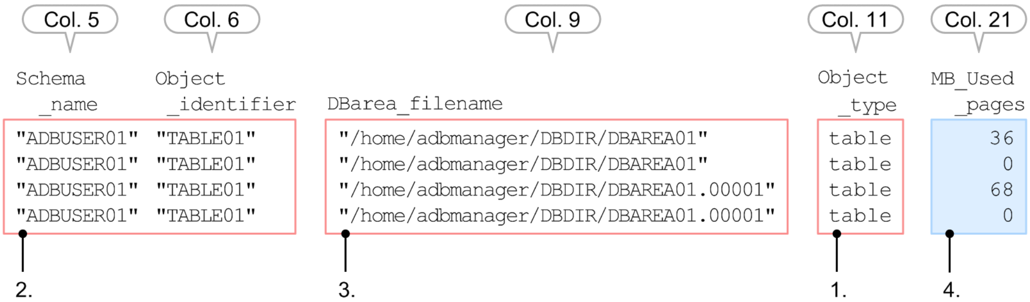
```
DBarea_filename                                     MB_Used_segments  MB_Total_segments
"/home/adbmanager/DBDIR/DBAREA01"                               1152               1152
"/home/adbmanager/DBDIR/DBAREA01.00001"                          984                984
```

To check the usage amount (in megabytes) of DB area `DBAREA01` for each DB area file:

1. Find the rows required for the calculation of usage values.
   To check the usage values for the DB area files in the DB area, find the rows that contain a value in the column headed `MB_Total_segments` (column 22 in the output results).
2. Verify the DB area file names.
   Verify the name of the DB area file names displayed in the column headed `DBarea_filename` (column 9 in the output results).
3. Obtain the usage amount (in megabytes) of the DB area for each DB area file.
   Obtain the value of the `MB_Used_segments` column (col. 20 in the output results) for each row (each DB area file).

(3) Checking the usage amount (in megabytes) of a table for each DB area file

■ Output example



```
Schema       Object                                                   Object   MB_Used
 _name        _identifier  DBarea_filename                             _type     _pages
"ADBUSER01"  "TABLE01"     "/home/adbmanager/DBDIR/DBAREA01"          table        36
"ADBUSER01"  "TABLE01"     "/home/adbmanager/DBDIR/DBAREA01"          table         0
"ADBUSER01"  "TABLE01"     "/home/adbmanager/DBDIR/DBAREA01.00001"    table        68
"ADBUSER01"  "TABLE01"     "/home/adbmanager/DBDIR/DBAREA01.00001"    table         0
```

To check the usage amount (in megabytes) of a table for each DB area file:

1. Find the rows required for the calculation of usage values.
   To check the usage values for the table in the DB area files, find the rows that contain `table` in the column headed `Object_type` (column 11 in the output results).
2. Verify the schema name and table identifier of the table whose usage values are to be checked.
   Verify the table' schema name displayed in the column headed `Schema_name` (column 5 in the output results) and the table identifier displayed in the column headed `Object_identifier` (column 6 in the output results).
3. Verify the DB area file names.

Verify the names of the DB area files displayed in the column headed `DBarea_filename` (column 9 in the output results).

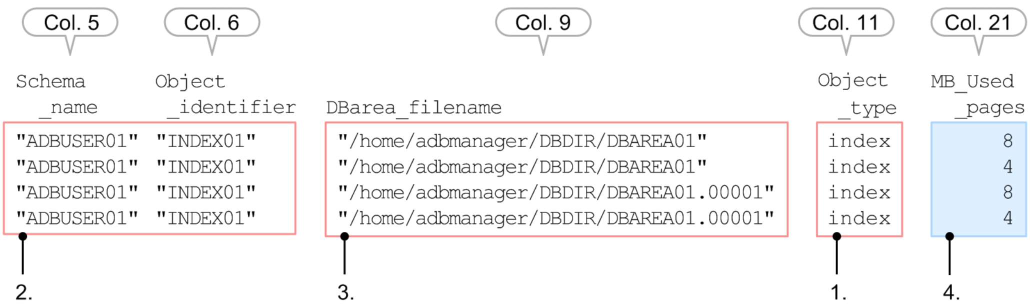
4. Obtain the usage amount (in megabytes) of the table for each DB area file.

   Obtain the value of the `MB_Used_pages` column (col. 21 in the output results) for each row. Total the values obtained for each DB area file to obtain the amount of table usage in each DB area file.

   In this example, the usage of DB area file `/home/adbmanager/DBDIR/DBAREA01` for table `ADBUSER01.TABLE0` is 36 megabytes + 0 megabytes = 36 megabytes, and the usage of DB area file `/home/adbmanager/DBDIR/DBAREA01.00001` for table `ADBUSER01.TABLE01` is 68 megabytes + 0 megabytes = 68 megabytes.

(4) Checking the usage amount (in megabytes) of an index for each DB area file

■ Output example



```
Schema       Object                                                 Object    MB_Used
 _name        _identifier  DBarea_filename                           _type     _pages
"ADBUSER01"  "INDEX01"     "/home/adbmanager/DBDIR/DBAREA01"        index          8
"ADBUSER01"  "INDEX01"     "/home/adbmanager/DBDIR/DBAREA01"        index          4
"ADBUSER01"  "INDEX01"     "/home/adbmanager/DBDIR/DBAREA01.00001"  index          8
"ADBUSER01"  "INDEX01"     "/home/adbmanager/DBDIR/DBAREA01.00001"  index          4
```

To check the usage amount (in megabytes) of an index for each DB area file:

1. Find the rows required for the calculation of usage values.

   To check the usage values for the index in the DB area files, find the rows that contain `index` in the column headed `Object_type` (column 11 in the output results).

2. Verify the schema name and index identifier of the index whose usage values are to be checked.

   Verify the index's schema name displayed in the column headed `Schema_name` (column 5 in the output results) and the index identifier displayed in the column headed `Object_identifier` (column 6 in the output results).

3. Verify the DB area file names.

   Verify the names of the DB area files displayed in the column headed `DBarea_filename` (column 9 in the output results).

4. Obtain the usage amount (in megabytes) of the index for each DB area file.

   Obtain the value of the `MB_Used_pages` column (col. 21 in the output results) for each row. Total the values obtained for each DB area file to obtain the amount of index usage in each DB area file.

   In this example, the usage of DB area file `/home/adbmanager/DBDIR/DBAREA01` for index `ADBUSER01.INDEX01` is 8 megabytes + 4 megabytes = 12 megabytes, and the usage of DB area file `/home/adbmanager/DBDIR/DBAREA01.00001` for index `ADBUSER01.INDEX01` is also 8 megabytes + 4 megabytes = 12 megabytes.

13.10.9 Finding the file size of each archive file corresponding to an archived chunk created in a specific table (outputting usage information of an archived chunk)

Find the file size (units: bytes) of each archive file corresponding to an archived chunk created in a specific table (ADBUSER01.TABLE01) by outputting usage information from the archived chunk.

- **Example of executing the adbdbstatus command**

```
adbdbstatus -d used                  ...1
            -c archivechunk          ...2
            -n ADBUSER01.TABLE01     ...3
            -r 2015/01/01-2015/12/31 ...4
```

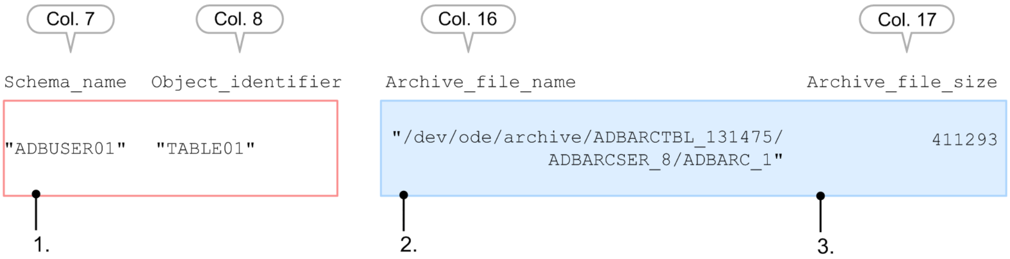
Explanation:

1. Usage information is specified as the information to be output.
2. Archived chunks are specified as a range from which to obtain usage information.
3. The table name (ADBUSER01.TABLE01) from which to obtain usage information about archived chunks is specified. The archivable multi-chunk table must be specified at this time.
4. A range of dates (2015/01/01-2015/12/31) for which to output chunk information is specified based on the values of the archive range columns corresponding to the chunks.
   The data type in the archive range column is DATE.

When the adbdbstatus command is executed, the usage information of archived chunks created in the table ADBUSER01.TABLE01 is output.

In the following procedure, the file size (units: bytes) of each archive file corresponding to the archived chunks is found based on the output results.

■ Output example



```
Schema_name  Object_identifier  Archive_file_name                                    Archive_file_size

"ADBUSER01"  "TABLE01"          "/dev/ode/archive/ADBARCTBL_131475/                             411293
                                          ADBARCSER_8/ADBARC_1"
```

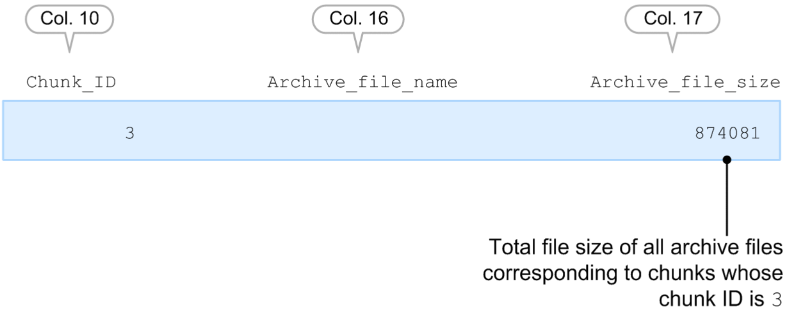
Procedure

1. Check the table schema name and table identifier.
   Confirm that the table schema name (ADBUSER01) is output in the Schema_name column (7th column in the output results), and that the table identifier (TABLE01) is output in the Object_identifier column (8th column in the output results).
2. Check the file name of each archive file corresponding to the archived chunk.
   Check the name of the file output in the Archive_file_name column (16th column in the output results).
3. Find the file size (units: bytes) of each archive file corresponding to the archived chunk.
   Check the value of the Archive_file_size column (17th column in the output results).

Note

If nothing is output to the `Archive_file_name` column, the value output to the `Archive_file_size` column is the total size of all archive files corresponding to the chunk. The chunk ID of the corresponding chunk can be checked in the `Chunk_ID` column (column 10 in the output results).

■ Output example



```
Chunk_ID           Archive_file_name            Archive_file_size

       3                                                   874081
```

Col. 10
Col. 16
Col. 17
Total file size of all archive files corresponding to chunks whose chunk ID is 3

13.10.10 Checking whether the column store table must be reorganized (output of information about the need for reorganization)

This example outputs the information about the need for reorganization of the specific column store table (`ADBUSER01.TABLE01`), and checks whether reorganization is required.

■Example of executing the adbdbstatus command

```
adbdbstatus -d reorginfo              ...1
            -n ADBUSER01.TABLE01      ...2
            -c 1-3                    ...3
```

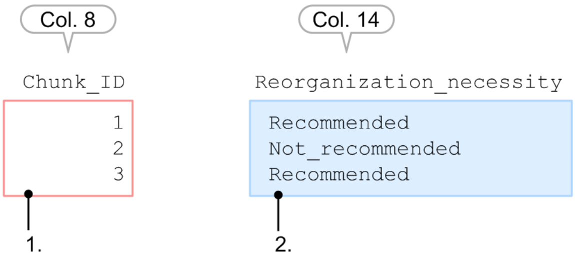
Explanation:

1. Specifies information about the need for reorganization as the information to be output.
2. Specifies the name of the column store table (`ADBUSER01.TABLE01`) from which information about the need for reorganization is to be collected.
3. Specifies the chunk IDs to be analyzed (information for chunks with chunk IDs 1 to 3 will be collected).

When the `adbdbstatus` command is executed, information about the need for reorganization for table `ADBUSER01.TABLE01` is output.

Use the following procedure to determine whether each chunk must be reorganized based on the output results.

■ Output example



```
Chunk_ID        Reorganization_necessity
       1          Recommended
       2          Not_recommended
       3          Recommended
```

Procedure

1. Check the chunk IDs.

   Confirm that the chunk IDs are output in the `Chunk_ID` column (column 8 in the output results).

2. Check whether each chunk must be reorganized.

   If a chunk must be reorganized, `Recommended` is output in the `Reorganization_necessity` column (column 14 in the output results). In this case, the `KFAA60200-W` message that recommends reorganization of the relevant chunk is also output.

   For details about how to reorganize a column store table, see *Reorganizing a single-chunk table* or *Reorganizing a multi-chunk table: Chunk-based reorganization* in the *HADB Setup and Operation Guide*.

13.11 Return codes

The following table lists and describes the return codes of the adbdbstatus command.

Table 13-24: Return codes of the adbdbstatus command

Return code	Description
0	The adbdbstatus command terminated normally.
4	The adbdbstatus command terminated normally, but a warning message was output. Check the displayed message or the message log file and take appropriate action. If the KFAA60200-W message that recommends reorganization is output but no other warning message or error message is output, return code 0 is output.
8	One of the following errors occurred: • The specified adbdbstatus command did not execute because it was invalid. • An error occurred during execution of the adbdbstatus command. For the cause of the error and corrective action to take, see the displayed error message or the message that has been output to a message log file.

13.12 Output rules for the adbdbstatus command

The following are the output rules for summary information, usage information, and information about the need for reorganization:

1. Column titles are output on the top rows of the output results. If the `-t` option is specified, no titles are output. The titles do not use enclosing characters.
2. The date and time information that is output is based on the time zone set in the `TZ` environment variable at the machine where the HADB server is installed.
3. The last item output is suffixed with a linefeed (`LF`) code.
4. The characters used in the output results are based on the character encoding set in the `ADBLANG` environment variable at the machine where the HADB server installed is used.
5. The following characters cannot be displayed. Therefore, a single-byte space is displayed for one character.
    - If the character encoding is `SJIS`

      `0x00` to `0x1F`, `0x7F` to `0xA0`, `0xE0` to `0xFF`, and byte strings not supported by `SJIS`
    - If the character encoding is `UTF-8`

      `0x00` to `0x1F`, `0x7F`, and byte strings not supported by `UTF-8`

13.13 Notes

1. If the adbdbstatus command terminates normally, it does not output a start message or a termination message with a return code. If the adbdbstatus command has terminated abnormally, check the displayed message or message log file and take appropriate action.

2. The lock of the adbdbstatus command differs depending on the specified option. When you execute the adbdbstatus command concurrently with another SQL statement or command, make sure that a lock conflict does not occur. For details about locks for SQL statements and commands, see the following in the *HADB Setup and Operation Guide*: *Locked resources that are reserved and their lock modes*, and *Locks performed when the adbdbstatus command is executed (to analyze the database status)* in *Examples of locks*.

   When summary information for DB areas is output

   A definition SQL statement or adbmodarea command cannot be executed concurrently with the adbdbstatus command.

   When information other than summary information for DB areas is output

   A definition SQL statement or adbmodarea command cannot be executed concurrently with the adbdbstatus command.

   Also note that the adbdbstatus command that processes DB areas cannot be executed concurrently with an SQL statement or command that updates those DB areas.

   In addition, the adbdbstatus command that processes tables or indexes cannot be executed concurrently with an SQL statement or command that updates the DB area where those tables and indexes are stored.

   Note that the adbdbstatus command with the --shared-lock option specified can be executed concurrently with other SQL statements and commands. For details about the --shared-lock option, see --shared-lock option in (3) Explanation of options in 13.2.1 Explanation of the specification format and options (outputting summary information or usage information).

3. Use the adbcancel command to cancel execution of the adbdbstatus command. For details about how to use the adbcancel command to suspend an executing command, see 1.6 When suspending (forcibly terminating) an executing command.

4. DB status analysis processing is performed by server processes. Therefore, using the **Ctrl**+**C** keys to forcibly terminate the adbdbstatus command process will not interrupt database status analysis processing that is being performed by server processes (this processing continues). When the DB status analysis processing is complete, the KFAA81001-I message is output to the message log file.

5. Because the adbdbstatus command does not change any contents of DB areas, tables, or indexes, there is no need to restore tables or indexes in the event of abnormal termination of the adbdbstatus command. In such a case, re-execute the adbdbstatus command as needed.

6. If the adbdbstatus command is executed while the table is non-updatable due to cancellation of one of the following commands, latest values might not be displayed for items relating to the number of pages, segments, or chunks. (The values that existed when the relevant command started might be displayed.)

   - adbimport command (when background import is performed)
   - adbidxrebuild command (executed with the --create-temp-file option specified after background import is canceled)
   - adbunarchivechunk command

7. If the adbdbstatus command is executed following any of the events listed below, the values output for the number of segments in use and the number of pages in use might not be correct:

   - Abnormal termination of the HADB server while the database was being updated.

- Output of the KFAA30919-E error message indicating that a global buffer shortage occurred during database update processing in multiple transactions.
- Error termination of a command or of an SQL statement (such as a definition SQL statement) that releases segments.

8. If merge-source chunks have not been completely deleted by the adbmergechunk command, the number of chunks created for the table in the summary information for the table will include the system chunk that is used temporarily during merge chunk processing. Therefore, the number of chunks created for the table in the table summary information might be greater by one than the maximum number of chunks specified in the table definition.
9. If table segments are merged by the adbmergechunk command, the number of pages that cannot be reused might increase, resulting in an increase in the number of pages that are used. Therefore, when you use the adbdbstatus command to check the number of pages in use, the values before and after the merge chunk processing might differ (might increase afterwards).
10. When the following items are output in megabytes or gigabytes (M or G, respectively, is specified in the -S option), the value of each item is rounded up and output as positive integers or as 0.

    - Output items for usage amounts (summary information or usage information)

      Table 13-2: Display units and column titles for usage amount output items in (3) Explanation of options in 13.2.1 Explanation of the specification format and options (outputting summary information or usage information)

    - Output items for the necessity of reorganization (information about the need for reorganization)

      Table 13-14: Display units and column titles for the output items for the necessity of reorganization in (2) Explanation of options in 13.2.2 Explanation of the specification format and options (outputting information about the need for reorganization of column store tables)

    If the value of each item is small (such as less than 102.4 kilobytes when usage amounts are output in megabytes, or less than 102.4 megabytes when usage amounts are output in gigabytes), 0 is output for each item. If you want to obtain accurate values, specify K in the -S option so that the items will be output in kilobytes.

11. The display units for the following items are shown in the title line.

    - Output items for usage amounts (summary information or usage information)

      Table 13-2: Display units and column titles for usage amount output items in (3) Explanation of options in 13.2.1 Explanation of the specification format and options (outputting summary information or usage information)

    - Output items for the necessity of reorganization (information about the need for reorganization)

      Table 13-14: Display units and column titles for the output items for the necessity of reorganization in (2) Explanation of options in 13.2.2 Explanation of the specification format and options (outputting information about the need for reorganization of column store tables)

    To check the display units for each item, you must omit the -t option so that the title line will be output. When you save the output results of the adbdbstatus command, save both the title line and the output results so that you can check the display units.

12. When any of the types of information listed below are output, information related to the tables and indexes used by HADB is output for the rows that are related to dictionary DB areas and system-table DB areas. (Information other than the tables and indexes defined by HADB users is output.)

    - Table summary information
    - Index summary information
    - Usage information for DB areas, tables, and indexes

13. If an error occurs during execution of the adbreorgsystemdata command, the remaining pre-reorganization data, if any, is output as deletion-pending chunk.

14 adbexport (Export Data)

This chapter explains the functionality provided by the `adbexport` command and how to use the command.

Note that in this chapter the term *dictionary table* means *dictionary table (viewed table)* and the term *system table* means *system table (viewed table)*.

14.1 Description

This section explains the functionality provided by the adbexport command and the threads to be used.

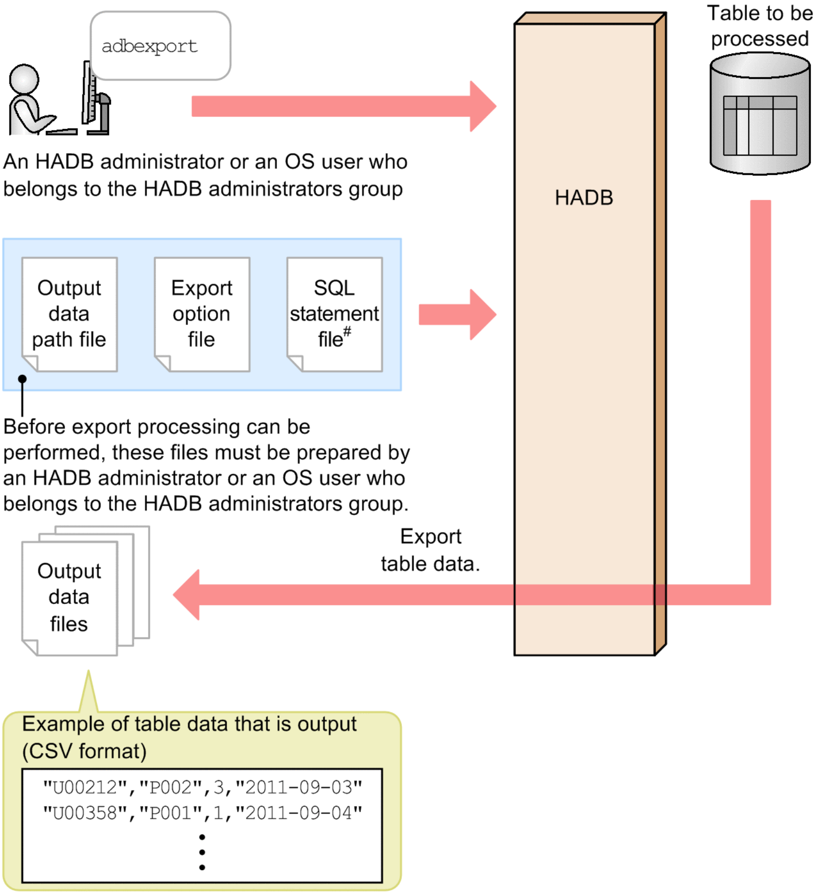
When executed, the adbexport command establishes a connection to the HADB server.

14.1.1 Functional overview

The adbexport command is used to export data from base tables and viewed tables into files. This processing is called *data export processing*. If you specified a table name for the adbexport command, data of the specified table (base table or viewed table) is output. If you specified an SQL statement file for the adbexport command, the search results of the SQL statement (the SELECT statement) written in the file is output.

The following figure provides an overview of data export processing.

Figure 14-1: Overview of data export processing


adbexport
An HADB administrator or an OS user who belongs to the HADB administrators group
Table to be processed
HADB
Output data path file
Export option file
SQL statement file#
Before export processing can be performed, these files must be prepared by an HADB administrator or an OS user who belongs to the HADB administrators group.
Export table data.
Output data files
Example of table data that is output (CSV format)

```
"U00212","P002",3,"2011-09-03"
"U00358","P001",1,"2011-09-04"
              :
```

#: You specify an SQL statement file only if you want to export a smaller subset of the retrieved data.

Explanation:

The adbexport command enables you to export data from a table into a file. It also enables you to export a smaller subset of the retrieved data when you specify an SQL statement file.

The adbexport command can export data from the following types of tables:

- Base tables for which the EXPORT TABLE privilege has been granted

- Viewed tables for which the EXPORT TABLE privilege has been granted
- Dictionary tables
- System tables

Note

Exporting data in a multi-chunk table

If you want to output data contained in a *chunk in wait status*, you need to specify the chunk ID of that chunk as the output target to export only a chunk of data. If the chunk ID is not specified, the chunk in wait status will not be output.

For details about how to output a chunk of data, see the explanation of the -c option in (3) Explanation of options in 14.2.1 Explanation of the specification format and options.

The following explains the files:

- **Output data files**

  The table data is output to these files. The output format is CSV. For details about the CSV format, see 14.5 Format of output data files (CSV format).

  An output data file can be compressed and then exported as a GZIP-format file. For details about the compression format of output data files, see 14.6 Compression format of output data files.

- **Output data path file**

  This file is used to specify a list of path names of the output destinations for the output data files.

- **Export option file**

  This file contains options that are to be used to set up an execution environment for the adbexport command.

- **SQL statement file**

  This file is used to specify a data retrieval SELECT statement for exporting a smaller subset of the retrieved data.

14.1.2 Relationships among threads used by the adbexport command

The adbexport command uses processing real threads in the following processing:

1. Retrieving table data

   One or more processing real threads are used.

2. Outputting retrieval results

   One or more processing real threads are used.

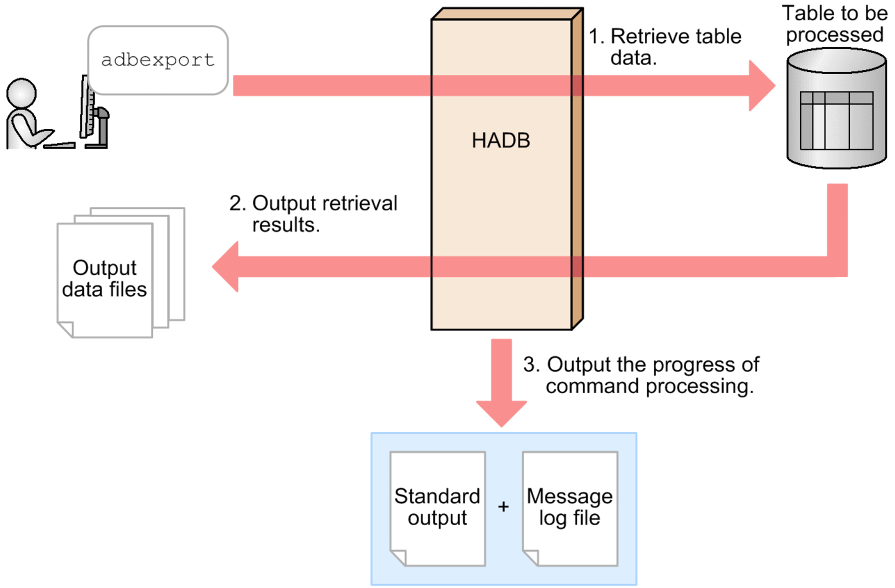
3. Outputting the progress of command processing

   One processing real thread is used.

Steps 1 to 3 are performed at the same time.

The following figure shows the processing of the adbexport command using processing real threads. The preceding numbers correspond to the numbers in the figure.

Figure 14-2: Processing of the adbexport command using processing real threads



The number of processing real threads used by the `adbexport` command for each step is automatically calculated from the *number of processing real threads to be used by the `adbexport` command*. For details about the *number of processing real threads to be used by the `adbexport` command*, see the description of the adb_export_rthd_num export option in (a) Options related to performance in (2) Explanation of specification format in 14.2.2 Format of export options.

14.2 Specification format for the adbexport command

This section explains the specification format for the `adbexport` command and how to specify the export options.

The following OS users can execute the `adbexport` command:

- HADB administrator
- OS users who belong to the HADB administrators group

14.2.1 Explanation of the specification format and options

(1) Specification format

```
adbexport
  -u authorization-identifier
  [-p password]
  [-k {enclosing-character|none}]
  [-s {delimiter|tab}]
  [-g output-interval-for-data-export-progress-message]
  [-z export-option-file-name]
  {-q SQL-statement-file-name|-n target-table[ -c chunk-ID]}
  [--compress GZIP]
  [--with-column-name]
  output-data-path-file-name
```

(2) Relationships among options

The options to be specified depend on what type of information you want to export with the `adbexport` command. The table below shows the relationships among the option specifications for each data export pattern. For details about the options, see the descriptions of the -q, -n, and -c options in (3) Explanation of options.

Table 14-1: Option specifications for the data export patterns

No.	Data export pattern	-q option	-n option	-c option
1	Exporting all data from a table#1	N	Y	N
2	Exporting only the data that was imported during a specific period of time (data in a specified chunk)#2	N	Y	Y
3	Exporting the retrieval results from a `SELECT` statement#1	Y	N	N

Legend:

Y: The option is specified.

N: The option is omitted.

Note:

The `adbexport` command results in an error when any of the following combinations of options is specified:

- The `-q` and `-n` options are specified together
- The `-q` and `-c` options are specified together
- Only the `-c` option is specified

#1

Data included in a wait status chunk cannot be exported.

#2

This data export pattern is applicable only to multi-chunk tables.

(3) Explanation of options

- `-u` *authorization-identifier*

  ~<character string>((1 to 100 bytes))

  Specifies the authorization identifier of the HADB user who executes the `adbexport` command.

  In this option, specify the authorization identifier of an HADB user who has the following privileges:

  - When exporting a dictionary table or system table

    Specify the authorization identifier of an HADB user who has the `CONNECT` privilege.

  - When exporting a base table or viewed table that was defined by the HADB user

    Specify the authorization identifier of an HADB user who has the `CONNECT` privilege and the `EXPORT TABLE` privilege for the table to be exported.

  For this option, specify an authorization identifier in the range from 1 to 100 bytes. Note that the byte count (1 to 100 bytes) does not include the double quotation marks used to enclose the authorization identifier.

  > **Important**
  >
  > If the character string used as the authorization identifier includes any lowercase letter or a backslash (\), make sure that you check the rules for specifying authorization identifiers. For the rules for specifying authorization identifiers, see 1.4.2 Rules for specifying authorization identifiers and passwords.

- `-p` *password*

  ~<character string>((1 to 255 bytes))

  Specifies the password for the authorization identifier that was specified in the `-u` option.

  > **Important**
  >
  > If the password includes a character that has a special meaning in the OS or shell, such as a double quotation mark (") or vertical bar (|), make sure that you check the rules for specifying passwords. For the rules for specifying passwords, see 1.4.2 Rules for specifying authorization identifiers and passwords.

  If the `adbexport` command is executed with this option omitted, a message prompting the user to enter a password is displayed. In an environment in which a password cannot be entered from the standard input, such as when the command is executed in the background, make sure that you do not omit this option.

- `-k {`*enclosing-character*`|none}`

  ~<character string>((1 byte))<<">>

  Specifies an *enclosing character* to be used to enclose field data in the output data file. If this option is omitted, the double quotation mark (") is assumed to be the enclosing character. If `none` is specified for this option, data without enclosing characters is output.

  A single-byte character must be specified as the enclosing character. A character string consisting of two or more bytes cannot be specified. None of the following characters can be specified as the enclosing character:

- Space, tab, asterisk (`*`)
- The character that is used as the delimiter (the value of or default value for the `-s` option)

If numeric data or datetime data is defined for a table that is to be exported, do not specify any character that is used in that numeric or datetime data as the enclosing character. If such a character is specified as the enclosing character, a conversion error might occur when data that was exported is imported.

Important

- If you specify a single quotation mark (`'`) as the enclosing character, enclose it in double quotation marks (`"`).

  Example: If you specify a single quotation mark (`'`) as the enclosing character:

  ```
  adbexport ... -k "'" ...
  ```

- If you specify as the enclosing character a special character used by the shell, such as the vertical bar (`|`), enclose it in either double quotation marks (`"`) or single quotation marks (`'`).

  Example: If you specify the vertical bar (`|`) as the enclosing character:

  ```
  adbexport ... -k "|" ...
  ```

- If character string data contains the enclosing character, two consecutive instances of the enclosing character are output.

Important

If `none` is specified for this option, data of all data types is output without enclosing characters to the output data file. The output data file containing data without enclosing characters might not be able to be used as an input data file of the `adbimport` command. The following shows examples.

Example 1: If data `A,B,C` is stored in a `CHARACTER` type column:

The output data is `A,B,C`. If you import this data with a comma (`,`) specified as the delimiter, the imported data is handled as separate data `A`, `B`, and `C`. In other words, the data is not handled as a single data unit `A,B,C`. Therefore, the data might not be stored in the table as expected or an import error might occur.

Example 2: If zero-byte length data is stored in a `VARCHAR` type column:

The output data is a zero-byte length character string. If this data is imported without using enclosing characters, the data is handled as a null value. Therefore, the data is handled as different data from the original one.

Example 3: If data `AB\nC` including the line break code (`\n`) is stored in a `CHARACTER` type column:

The output data is `AB\nC`. If this data is imported, it is handled as separate data `AB` and `C`. In other words, the data is not handled as a single data unit `AB\nC`. Therefore, the data might not be stored in the table as expected or an import error might occur.

- `-s {`*delimiter*`|tab}`

  ~<character string>((1 byte))<<`,`>>

  Specifies the *delimiter* that is to be used to delimit field data in the output data files. To specify the tab character as the delimiter, specify `tab`. If this option is omitted, the comma (`,`) is assumed to be the delimiter.

  Only the `tab` character or a single-byte character can be specified as a delimiter. Except for the `tab` character, a character string consisting of two or more bytes cannot be specified. None of the following characters can be specified as the delimiter:

- Alphanumeric characters (uppercase letters `A` to `Z`, lowercase letters `a` to `z`, numeric characters `0` to `9`), underscore (_), double quotation mark (`"`), asterisk (`*`)
- The character that is used as the enclosing character (the value of or default value for the `-k` option)

None of the following characters is suitable for the delimiter because they might also be used as character codes in table data:

- Signs (+, −), forward slash (/), colon (`:`), period (`.`), vertical bar (`|`), backslash (\), `[`, `]`, `(`, `)`, `{`, `}`, swung dash (~)

Important

- If you specify a single quotation mark (`'`) as the delimiter, enclose it in double quotation marks (`"`).

  Example: If you specify a single quotation mark (`'`) as the delimiter:

  ```
  adbexport ... -s "'" ...
  ```

- If you specify as the delimiter a special character used by the shell, such as the vertical bar (`|`), enclose it in either double quotation marks (`"`) or single quotation marks (`'`).

  Example: If you specify the vertical bar (`|`) as the delimiter:

  ```
  adbexport ... -s "|" ...
  ```

- `-g` *output-interval-for-data-export-progress-message*

  ~<integer>((0 to 1,000,000,000))<<0>>(in 1,000 rows)

  Specifies the interval (in units of 1,000 rows) at which the data export processing progress message is to be output. For example, if `2` is specified in this option, the `KFAA80205-I` progress message is output each time 2,000 rows of data are exported.

  If this option is omitted or `0` is specified in this option, the progress message is not output.

- `-z` *export-option-file-name*

  ~<OS path name>((2 to 510 bytes))

  Specifies the absolute path name of the export option file that contains the export options. If this option is omitted, all export options are set to their default values because no export option file is specified.

  For details about the export options, see 14.2.2 Format of export options.

  Make sure that the export option file name is unique from all other file names. If the export option file has the same name as another file, data in the file might become corrupted. For details about the files whose names must differ from the export option file name, see 14.8 Notes.

- `-q` *SQL-statement-file-name*

  ~<OS path name>((2 to 510 bytes))

  Specifies the absolute path name of the SQL statement file. When this option is specified, the `-c` option cannot be specified.

  An SQL statement file is used to specify a `SELECT` statement for data retrieval when you intend to export that statement's retrieval results.

  The following shows an example of an SQL statement file.

  Example of SQL statement file

  ```
  /*COMMENT*/ SELECT * FROM "TABLE01"
  WHERE "C1" < 3000
  ```

  The specification rules for the SQL statement file are as follows:

  - The maximum size of the SQL statement that can be specified in an SQL statement file is 16,000,000 bytes.

- Only one SQL statement can be specified in an SQL statement file.
- No dynamic parameters can be specified in an SQL statement file.
- To create an SQL statement file, you must use the character encoding set in the `ADBLANG` environment variable for the machine on which the HADB server is installed.
- All code characters in the range from `X'00'` to `X'1F'` are converted to single-byte spaces.
- The SQL statement can contain line break codes. Use `X'0A'` (`LF`) or `X'0D0A'` (`CRLF`) as the line break code. Each line break code in the SQL statement is converted to a single-byte space, and multiple consecutive line break codes are concatenated for interpretation.
- All characters specified between `/*` and the first `*/` that follows `/*` in an SQL statement file are treated as a comment. For details about specifying comments, see *Comments* in *Rules for separators* in *SQL writing conventions* in *SQL Basics* in the manual *HADB SQL Reference*.
- Only one semicolon (`;`) can be specified at the end of the SQL statement. The specified semicolon is converted to a single-byte space for interpretation. A character string that contains single-byte spaces, tabs, or line breaks can only be specified after the specified semicolon.

> **Important**
>
> If a work table might be created during retrieval processing (such as when the `ORDER BY` clause, `GROUP BY` clause, or `DISTINCT` is specified), it might take a while after the `adbexport` command is executed before a progress message or an output count message is output.

The SQL statement specified in an SQL statement file can retrieve the following tables:

- Base tables or viewed tables for which the `EXPORT TABLE` privilege has been granted

  If you are retrieving a base table or a viewed table that you own, you can omit the schema name. If you are retrieving a base table or a viewed table owned by another HADB user, you must specify the schema name.
- Dictionary tables and system tables

  If you will be retrieving a dictionary table or a system table, specify `"MASTER"` as the schema name.

Note that the SQL statement that is actually executed is output to the message log file.

- `-n` *table-to-be-processed*

Specifies the table from which data is to be exported in the format *schema-name*`.`*table-identifier*. For details about the specification rules that apply when a schema name or table identifier includes any lowercase letter or backslash (`\`), see 1.4.3 Table name specification rules.

If you want to export only a chunk of data from a multi-chunk table, specify the `-c` option together with this option.

You can specify the following tables in this option:

- Base tables or viewed tables for which the `EXPORT TABLE` privilege has been granted

  If you are retrieving a base table or a viewed table that you own, you can omit the schema name. If you are retrieving a base table or a viewed table owned by another HADB user, you must specify the schema name.
- A dictionary table or system table

  If you will be retrieving a dictionary table or a system table, specify `"MASTER"` as the schema name.

- `-c` *chunk-ID*

~<integer>((1 to 9,223,372,036,854,775,807))

Specifies a chunk ID to export only a chunk of data from a multi-chunk table. If you specify this option, you can output data contained in the chunk of the specified ID even if that chunk is in wait status.

When you specify this option, you must specify a multi-chunk table in the `-n` option. When this option is specified, the `-q` option cannot be specified.

If the specified chunk ID does not exist in the table to be processed, the output data count will be zero.

The following explains how to determine the desired chunk ID:

- Search system tables for STATUS_CHUNKS

  Search system tables for STATUS_CHUNKS, and check the chunk IDs. For details, see *Checking the information about all chunks in a table based on a table name* in *Searching system tables* in *System Tables* in the *HADB Setup and Operation Guide*.

- Execute the adbdbstatus command

  Execute the following command to output the usage information for DB areas, tables, and indexes, and then check the chunk IDs (Chunk_ID) (the results are output in CSV format). For details about the items that are output as usage information for DB areas, tables, and indexes, see 13.7.2 List of items that are output as usage information for DB areas, tables, and indexes.

  ```
  adbdbstatus -d used -c table -n base-table-name
  ```

  Specify *base-table-name* in the format *schema-name*.*table-identifier*.

- --compress GZIP

  Specify this option if you want to compress the output data file and then export it as a GZIP-format file.

- --with-column-name

  Specify this option if you want to output the column name in the first line in each output data file.

  The column name to be output is determined by the option specified with this option. The following table describes the rules for output column names.

  Table 14-2: Column name that is output

No.	Option specified with this option	Column name that is output
1	-n option (to output data of the table to be processed)	Column name of the table to be processed

2	-q option (to output the result of searching for the SQL statement in the *SQL-statement-file*)	Table column names derived as a result of a query expression in an SQL statement specified in the *SQL-statement-file* (retrieval item column names)
		Example: If the following SQL statement is specified in the *SQL-statement-file*, column names C1, MAX_C2, and MIN_C2 are output.
		`SELECT "C1",MAX("C2") AS "MAX_C2",MIN("C2") AS "MIN_C2"`
		`    FROM "T1" GROUP BY "C1"`
		For details about the naming rules for columns in derived tables, see *Rules* in *Specification format and rules for the SELECT statement* in the manual *HADB SQL Reference*.

  Column names are output in the same format as columns of VARCHAR type data. For details about the output format of VARCHAR type data, see 14.5.3 Output format for field data.

  If multiple output data files are specified, the column name is output in the first line in every output data file. However, the column name is not output in files for which no data is output.

  > **Important**
  >
  > Do not specify this option if you want to use the output data file as an input data file of the adbimport command. If a file that was output with this option specified is used as an input data file of the adbimport command, the column name is also stored as table data.

- *output-data-path-file-name*

  ~<OS path name>((2 to 510 bytes))

Specifies the absolute path name of the output data path file. If the path name contains a space, enclose the entire path name in single quotation marks (`'`).

In the output data path file, specify the storage location for output data files. The storage location for output data files must be expressed as an absolute path consisting of 2 to 510 bytes. The characters permitted in the path names are the same as for the output data path file name.

The HADB server does not add an extension to output data files. The output data file is created as specified for the absolute path of the output data file. Therefore, when you specify the `--compress GZIP` option, we recommend that you specify the file extension `.gz` at the end of the absolute path of the output data file. Specifying the file extension `.gz` at the end of the path makes it easier to identify the exported file as a GZIP-format file.

To improve the performance of data export processing, we recommend that you specify multiple output data file storage locations. However, if it is important to store the exported data exactly in the order of the retrieval results, specify only one output data file storage location.

If frequent output operations are performed on the output data files stored under a single directory, performance might be adversely affected. We recommend that you specify directories on multiple disks as output data file storage locations to distribute the workload of output operations on the disk.

The following shows an example of an output data path file.

- **Example of output data path file (if the `--compress GZIP` option is not specified)**

```
/mnt/diska/wwwww/exp1.csv
/mnt/diskb/xxxxx/exp2.csv
/mnt/diskc/yyyyy/exp3.csv
/mnt/diskd/zzzzz/exp4.csv
```

This example specifies four output data file storage locations to export data to four output data files.

- **Example of output data path file (if the `--compress GZIP` option is specified)**

```
/mnt/diska/wwwww/exp1.csv.gz
/mnt/diskb/xxxxx/exp2.csv.gz
/mnt/diskc/yyyyy/exp3.csv.gz
/mnt/diskd/zzzzz/exp4.csv.gz
```

This example specifies four storage locations for output data files to export data to four output data files (files compressed in GZIP format).

If the `--compress GZIP` option is specified, we recommend that you specify the file extension `.gz` at the end of the absolute path of the output data file as shown in the preceding example. Specifying the file extension `.gz` at the end of the path makes it easier to identify the exported file as a GZIP-format file.

Important

- When the `adbexport` command is executed, empty files are created at all output data file storage locations specified by the output data path file. Data is then exported into each file. Note that even if there is no data to be exported, a file with a size of zero bytes will be created.

  If the `--compress GZIP` is specified and there is no data to be exported, a file (GZIP format) with a size of zero bytes will be created.
- When multiple output data files are specified, the number of data items output for each file varies. Consequently, a file containing zero data items might be created.
- If the output data file storage locations specified by the output data path file contain a file with the same name as an output data file, the `adbexport` command will terminate in an error. If this happens, move or delete the existing file, and then re-execute the `adbexport` command.

Make sure that no file names used by any programs other than the `adbexport` command are the same as the output data path file name or any of the output data file storage locations specified by the output data path file. If there is any duplication of names, data in the output data path file and output data files might be corrupted by other programs. For details about the files whose names must differ from the names of the output data path file and output data file storage locations, see 14.8 Notes.

14.2.2 Format of export options

(1) Specification format

```
Options related to performance
  [set adb_export_rthd_num = number-of-processing-real-threads-to-be-used-by-adbexpor
t-command]
  [set adb_export_scan_buff_size = table-data-scan-buffer-size]
  [set adb_export_wrktbl_blk_num = number-of-local-work-table-buffer-pages]
  [set adb_export_hashtbl_area_size = hash-table-area-size]
  [set adb_export_hashflt_area_size = hash-filter-area-size]

Options related to output format
  [set adb_export_order_mode = {BYTE|ISO}]
```

The specification rules for these export options are the same as for the server definition. For details about the specification rules for the server definition, see *Syntax rules for the server definition* in the *HADB Setup and Operation Guide*.

(2) Explanation of specification format

(a) Options related to performance

- adb_export_rthd_num = *number-of-processing-real-threads-to-be-used-by-adbexport-command*

  ~<integer>((3 to 4,096))

  Specifies the number of processing real threads to be used for executing the adbexport command.

  Specification of this option is recommended. The adbexport command uses the number of processing real threads specified by this option. Therefore, specify an appropriate number of processing real threads for use by the adbexport command. For details about the processing of the adbexport command that uses processing real threads and the number of processing real threads to be used for each step, see 14.1.2 Relationships among threads used by the adbexport command.

  If this option is omitted, the number of processing real threads to be used for executing the adbexport command is determined by the adb_sql_exe_max_rthd_num operand in the server definition. The following table shows the relationship between the value of this option or the adb_sql_exe_max_rthd_num operand in the server definition and the number of processing real threads used for executing the command.

Table 14-3: Number of processing real threads used for executing the command

No.	Value in this option or in the server definition		Number of processing real threads used for executing the command
1	When this option is omitted	If the adb_sql_exe_max_rthd_num operand in the server definition is not specified	The default value of the adb_sql_exe_max_rthd_num operand in the server definition is used.
2		If a value in the range from 0 to 2 is specified in the adb_sql_exe_max_rthd_num operand in the server definition	--

No.	Value in this option or in the server definition		Number of processing real threads used for executing the command
3		If 3 or a greater value is specified for the `adb_sql_exe_max_rthd_num` operand in the server definition	The value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition is used.
4	If 3 or a greater value is specified in this option		The value specified in this option is used.

Legend:

--: The `adbexport` command results in an error.

Note:

For details about the `adb_sql_exe_max_rthd_num` operand in the server definition, see *Operands related to performance (set format)* in the *HADB Setup and Operation Guide*.

- `adb_export_scan_buff_size` = *table-data-scan-buffer-size*

  ~<integer>((1 to 2,048))<<16>> (units: megabytes)

  Specify the size (in megabytes) of the buffer to be used for retaining table data search results. A larger value can reduce overhead in searching table data.

  As a guideline, specify a value equivalent to the data length in the search results times about 10,000.

  For details about how to obtain data lengths in search results, see *Data length of each data type* in *Determining the memory requirement during normal operation* in *Estimating the HADB server's memory requirement* in *Preparing Resources* in the *HADB Setup and Operation Guide*.

  Note that all the retrieval results stored in one scan buffer are exported to the same output data file.

  If the specified buffer size is too small to store any data item obtained in the retrieval results, the `adbexport` command will result in an error.

- `adb_export_wrktbl_blk_num` = *number-of-local-work-table-buffer-pages*

  ~<integer>((5 to 100,000,000))

  Normally, you will omit this option. When this option is omitted, the number of local work table buffer pages specified in the `adb_dbbuff_wrktbl_clt_blk_num` server definition operand is applied.

  To minimize the execution time of an SQL statement that creates a local work table when the `adbexport` command is executed, specify in this option the number of local work table buffer pages. The local work table buffer allocated based on this option's value is used only when an SQL statement that creates a local work table is executed by the `adbexport` command. If this option is specified together with the `adb_dbbuff_wrktbl_clt_blk_num` server definition operand, the number of local work table buffer pages specified in this option is used when the `adbexport` command executes.

  For details about how to estimate a value for this option, see *Estimating the number of pages in the buffer for local work tables* in *Estimates related to work tables* in *Preparing Resources* in the *HADB Setup and Operation Guide*.

  The number of local work table buffer pages allocated by specifying this option cannot be changed by the `adbmodbuff` command. For details about the `adbmodbuff` command, see 35. adbmodbuff (Change Buffer).

Note

A local work table is created for each real thread that processes the SQL statement. Therefore, a local work table buffer with the number of pages specified in this option is allocated for each real thread that processes the SQL statement. For details about the SQL statements that create local work tables, see *Work tables created when SQL statements are executed* in *Considerations when executing an SQL statement that creates work tables* in *Designs Related to Improvement of Application Program Performance* in the *HADB Application Development Guide*.

- `adb_export_hashtbl_area_size =` *hash-table-area-size*

  ~<integer>((0 to 4,194,304)) (megabytes)

  Normally, you will omit this option. If this option is omitted, the size of the hash table area specified in the `adb_sql_exe_hashtbl_area_size` operand in the server definition operand is applied.

  To minimize the execution time of an SQL statement to which the hash retrieval is applied when the `adbexport` command is executed, specify in this option the size of the hash table area in megabytes. For details about estimating the value to be specified for this option, see *Tuning to shorten SQL statement execution time by re-examining the hash table area size* in the *HADB Setup and Operation Guide*.

  The hash table area allocated based on this option's value is used only when an SQL statement to which hash retrieval is applied is executed by the `adbexport` command. If this option is specified together with the `adb_sql_exe_hashtbl_area_size` operand in the server definition, the size of the hash table area specified in this option is applied when the `adbexport` command is executed. For details about the `adb_sql_exe_hashtbl_area_size` operand in the server definition, see *Operands related to performance* in the *HADB Setup and Operation Guide*.

- `adb_export_hashflt_area_size =` *hash-filter-area-size*

  ~<integer>((0 to 419,430)) (megabytes)

  Normally, you will omit this option. The value that is applied when this option is omitted varies depending on whether the `adb_export_hashtbl_area_size` export option is specified.

  - If the `adb_export_hashtbl_area_size` export option is specified:

    The following value is applied.

    ↑*value-of-the-adb_export_hashtbl_area_size-export-option*÷10↑

  - If the `adb_export_hashtbl_area_size` export option is not specified:

    The value of the `adb_sql_exe_hashflt_area_size` operand in the server definition is applied.

  To minimize the execution time of an SQL statement to which the hash filter is applied when the `adbexport` command is executed, specify in this option the size of the hash filter area in megabytes. For details about estimating the value to be specified for this option, see *Tuning to shorten SQL statement execution time by re-examining the hash filter area size* in the *HADB Setup and Operation Guide*.

  The hash filter area allocated based on this option's value is used only when an SQL statement to which hash filters are applied is executed by the `adbexport` command. If this option is specified together with the `adb_sql_exe_hashflt_area_size` operand in the server definition, the size of the hash filter area specified in this option is applied when the `adbexport` command is executed. For details about the `adb_sql_exe_hashflt_area_size` operand in the server definition, see *Operands related to performance* in the *HADB Setup and Operation Guide*.

(b) Options related to output results

- `adb_export_order_mode = {BYTE|ISO}`

  Specifies the sort order for character string data when the `ORDER BY` clause is specified in the SQL statement file.

  Specify this option when you want to export the retrieval results in a different sort order than the order specified in the `adb_sql_order_mode` server definition operand.

  If this option is omitted and an `ORDER BY` clause is specified in the SQL statement file, the retrieval results are exported in the order specified in the `adb_sql_order_mode` server definition operand. For details about the `adb_sql_order_mode` operand, see the topic *Operands related to SQL statements (set format)* in *Detailed descriptions of the server definition operands* in *Designing the Server Definition* in the *HADB Setup and Operation Guide*.

BYTE

Output character string data by bytecode.

ISO

Output character string data by sort code (ISO/IEC 14651:2011 compliance). Note that ISO cannot be specified when SJIS is specified in the HADB server's ADBLANG environment variable.

14.3 Examples

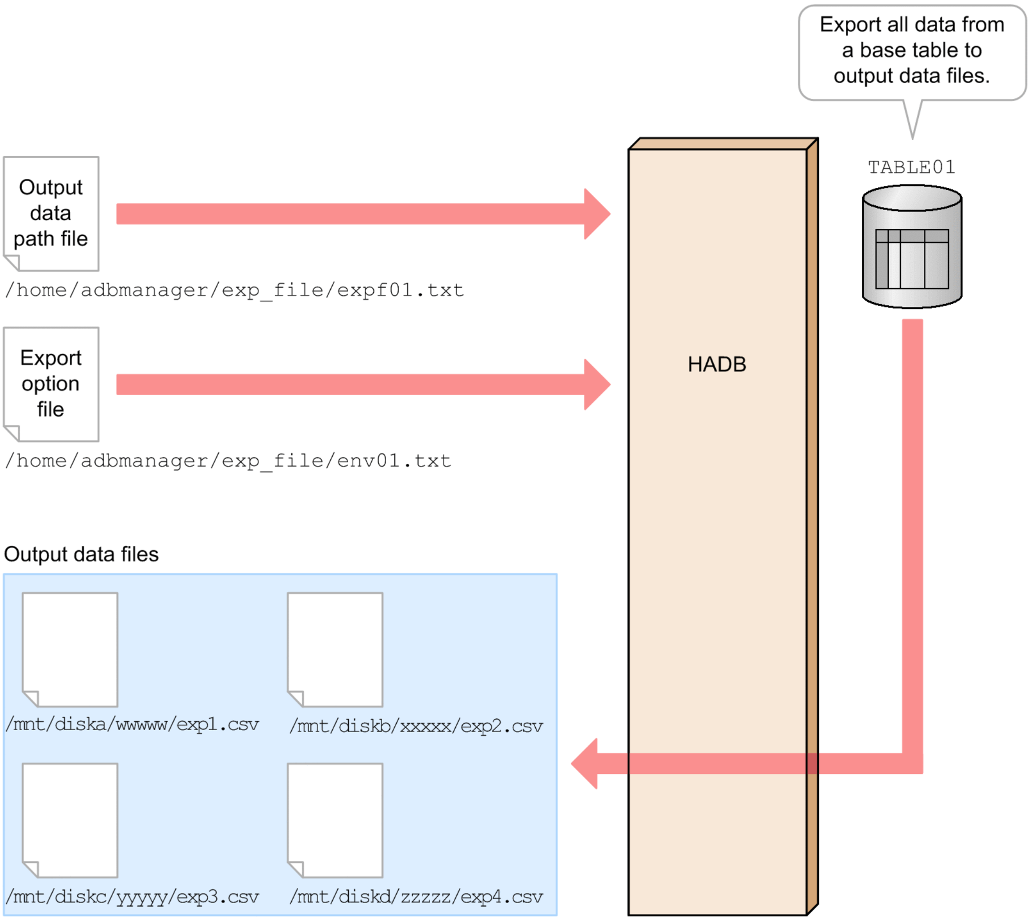
This section explains by way of examples how to use the `adbexport` command to export table data.

If the `adbexport` command terminates abnormally, take appropriate action based on the explanation in 14.7 Handling abnormal termination of the adbexport command.

14.3.1 Exporting all data from a table

This example exports table data under the following conditions:

- All data stored in base table `TABLE01` is exported.
- The export results are output to four output data files.


Export all data from a base table to output data files.
TABLE01
Output data path file
/home/adbmanager/exp_file/expf01.txt
Export option file
/home/adbmanager/exp_file/env01.txt
HADB
Output data files
/mnt/diska/wwwww/exp1.csv
/mnt/diskb/xxxxx/exp2.csv
/mnt/diskc/yyyyy/exp3.csv
/mnt/diskd/zzzzz/exp4.csv

This example uses the following procedure to execute the `adbexport` command.

Procedure

1. Create an output data path file.
2. Create an export option file.
3. Execute the `adbexport` command.

Using examples, the following subsections explain each of these steps.

(1) Creating an output data path file

This example creates an output data path file and specifies in it paths for all the output data files.

Example specification of an output data path file

```
/mnt/diska/wwwww/exp1.csv
/mnt/diskb/xxxxx/exp2.csv
/mnt/diskc/yyyyy/exp3.csv
/mnt/diskd/zzzzz/exp4.csv
```

(2) Creating an export option file

This example creates an export option file and specifies export options in it.

Example specification of an export option file

```
set adb_export_rthd_num = 3            ...1
set adb_export_scan_buff_size = 64     ...2
```

Explanation:

1. Specifies the number of processing real threads to be used for executing the `adbexport` command.
2. Specifies the size of the buffer to be used for retaining table data retrieval results.

(3) Executing the adbexport command

```
adbexport -u ADBUSER01                                ...1
          -p '#HelloHADB_01'                          ...2
          -z /home/adbmanager/exp_file/env01.txt      ...3
          -n TABLE01                                  ...4
          /home/adbmanager/exp_file/expf01.txt        ...5
```

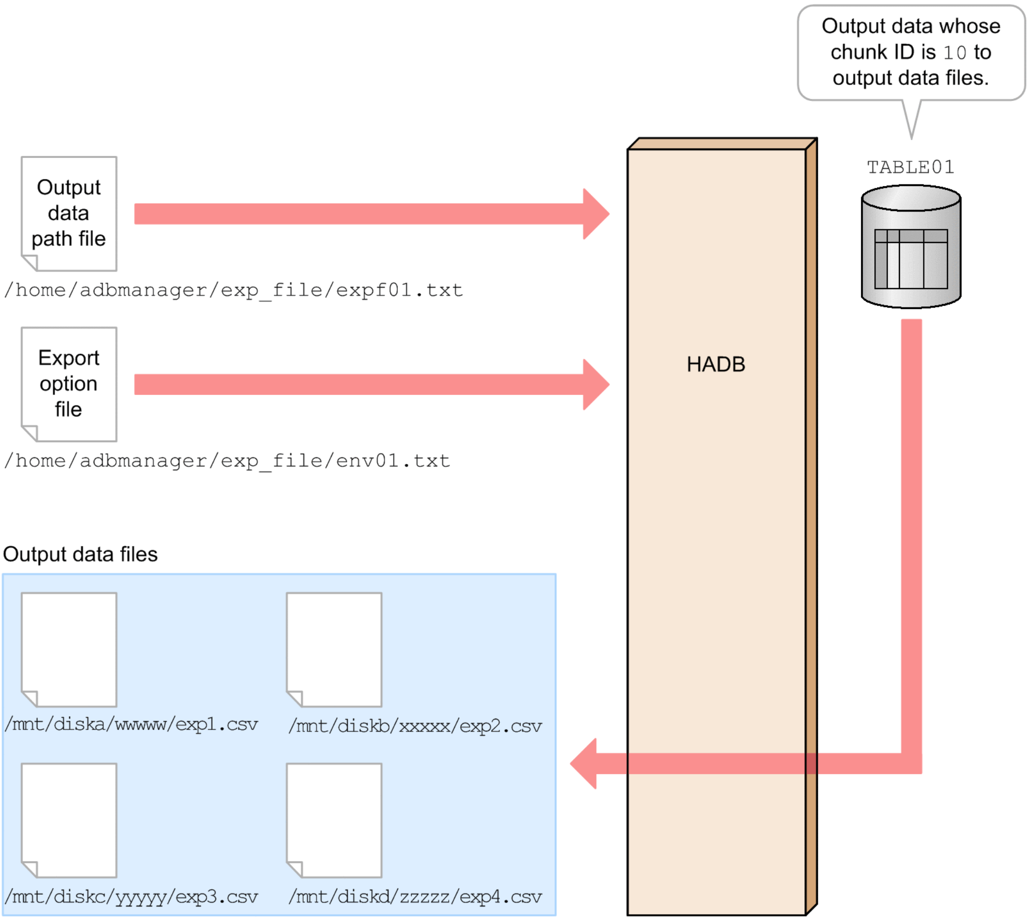
Explanation:

1. Specifies the authorization identifier of the HADB user who executes the `adbexport` command.
2. Specifies the password for the authorization identifier that is specified in the `-u` option.
3. Specifies the absolute path name of the export option file created in (2) Creating an export option file.
4. Specifies the table that is to be the target of export processing.
5. Specifies the absolute path name of the output data path file created in (1) Creating an output data path file.

14.3.2 Exporting data with a specific chunk ID

This example exports table data under the following conditions:

- Data for the chunk ID `10` is exported from multi-chunk table `TABLE01`.
- The export results are output to four output data files.


Output data whose chunk ID is 10 to output data files.
TABLE01
Output data path file
/home/adbmanager/exp_file/expf01.txt
Export option file
/home/adbmanager/exp_file/env01.txt
HADB
Output data files
/mnt/diska/wwwww/exp1.csv
/mnt/diskb/xxxxx/exp2.csv
/mnt/diskc/yyyyy/exp3.csv
/mnt/diskd/zzzzz/exp4.csv

This example uses the following procedure to execute the `adbexport` command.

Procedure

1. Create an output data path file.
2. Create an export option file.
3. Execute the `adbexport` command.

Using examples, the following subsections explain each of these steps.

(1) Creating an output data path file

This example creates an output data path file and specifies in it paths for all the output data files.

Example specification of an output data path file

```
/mnt/diska/wwwww/exp1.csv
/mnt/diskb/xxxxx/exp2.csv
/mnt/diskc/yyyyy/exp3.csv
/mnt/diskd/zzzzz/exp4.csv
```

(2) Creating an export option file

This example creates an export option file and specifies export options in it.

Example specification of an export option file

```
set adb_export_rthd_num = 3              ...1
set adb_export_scan_buff_size = 64       ...2
```

Explanation:

1. Specifies the number of processing real threads to be used for executing the `adbexport` command.
2. Specifies the size of the buffer to be used for retaining table data retrieval results.

(3) Executing the adbexport command

```
adbexport -u ADBUSER01                                ...1
          -p '#HelloHADB_01'                          ...2
          -z /home/adbmanager/exp_file/env01.txt      ...3
          -n TABLE01                                  ...4
          -c 10                                       ...5
          /home/adbmanager/exp_file/expf01.txt        ...6
```

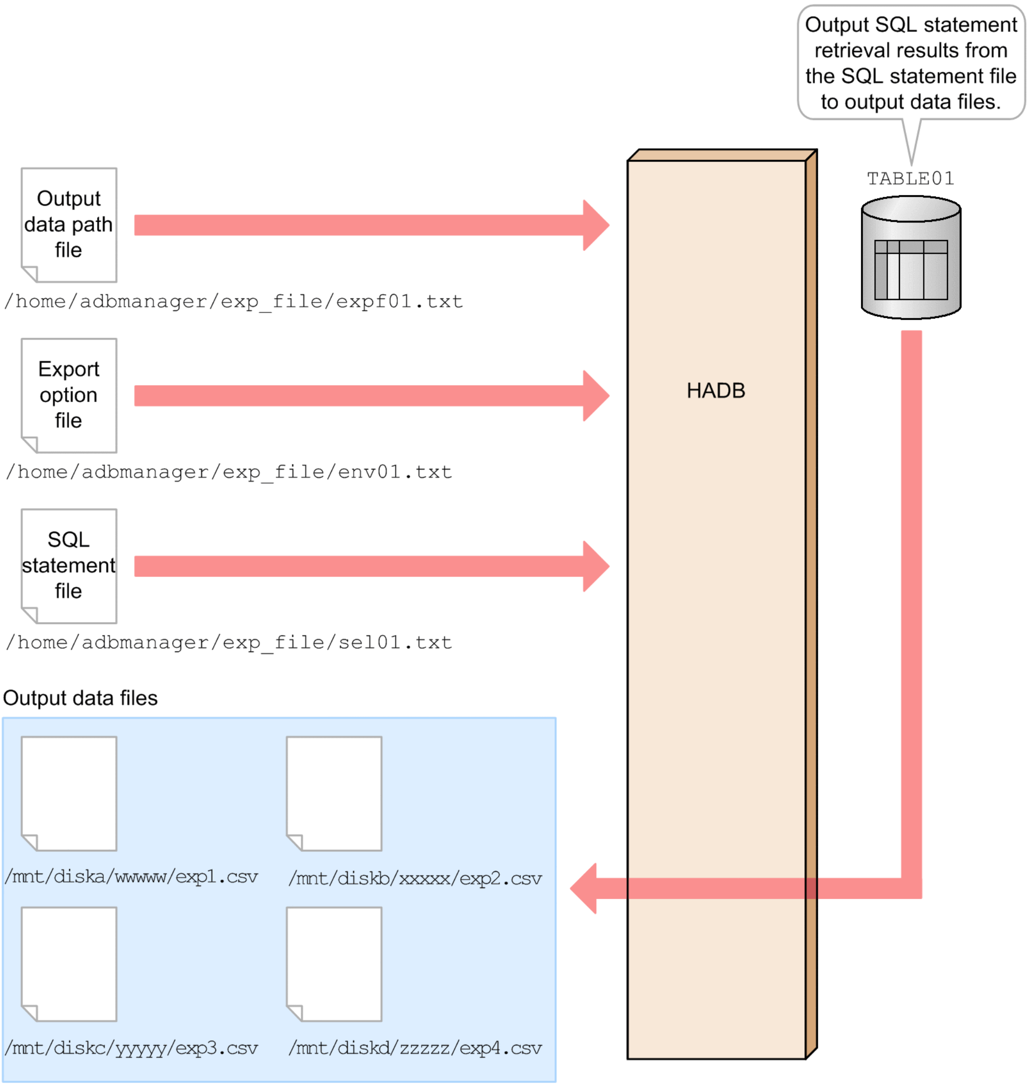
Explanation:

1. Specifies the authorization identifier of the HADB user who executes the `adbexport` command.
2. Specifies the password for the authorization identifier that is specified in the `-u` option.
3. Specifies the absolute path name of the export option file created in (2) Creating an export option file.
4. Specifies the table that is to be the target of export processing.
5. Specifies the chunk ID.
6. Specifies the absolute path name of the output data path file created in (1) Creating an output data path file.

14.3.3 Exporting the retrieval results of an SQL statement

This example exports table data under the following conditions:

- Data stored in base table `TABLE01` is exported.
- The retrieval results of the SQL statement specified in an SQL statement file are exported.
- The export results are output to four output data files.


Output SQL statement retrieval results from the SQL statement file to output data files.
TABLE01
Output data path file
/home/adbmanager/exp_file/expf01.txt
Export option file
/home/adbmanager/exp_file/env01.txt
SQL statement file
/home/adbmanager/exp_file/sel01.txt
HADB
Output data files
/mnt/diska/wwwww/exp1.csv
/mnt/diskb/xxxxx/exp2.csv
/mnt/diskc/yyyyy/exp3.csv
/mnt/diskd/zzzzz/exp4.csv

This example uses the following procedure to execute the `adbexport` command.

Procedure

1. Create an output data path file.
2. Create an export option file.
3. Create an SQL statement file.
4. Execute the `adbexport` command.

Using examples, the following subsections explain each of these steps.

(1) Creating an output data path file

This example creates an output data path file and specifies in it paths for all the output data files.

Example specification of an output data path file

```
/mnt/diska/wwwww/exp1.csv
/mnt/diskb/xxxxx/exp2.csv
/mnt/diskc/yyyyy/exp3.csv
/mnt/diskd/zzzzz/exp4.csv
```

(2) Creating an export option file

This example creates an export option file and specifies export options in it.

Example specification of an export option file

```
set adb_export_rthd_num = 3              ...1
set adb_export_scan_buff_size = 64       ...2
```

Explanation:

1. Specifies the number of processing real threads to be used for executing the `adbexport` command.
2. Specifies the size of the buffer to be used for retaining table data retrieval results.

(3) Creating an SQL statement file

This example creates an SQL statement file and specifies in it the SQL statement to be used to retrieve the desired data.

Example specification of an SQL statement file

```
SELECT * FROM "TABLE01"
     WHERE "C1" < 10000
```

(4) Executing the adbexport command

```
adbexport -u ADBUSER01                                  ...1
          -p '#HelloHADB_01'                            ...2
          -z /home/adbmanager/exp_file/env01.txt        ...3
          -q /home/adbmanager/exp_file/sel01.txt        ...4
          /home/adbmanager/exp_file/expf01.txt          ...5
```

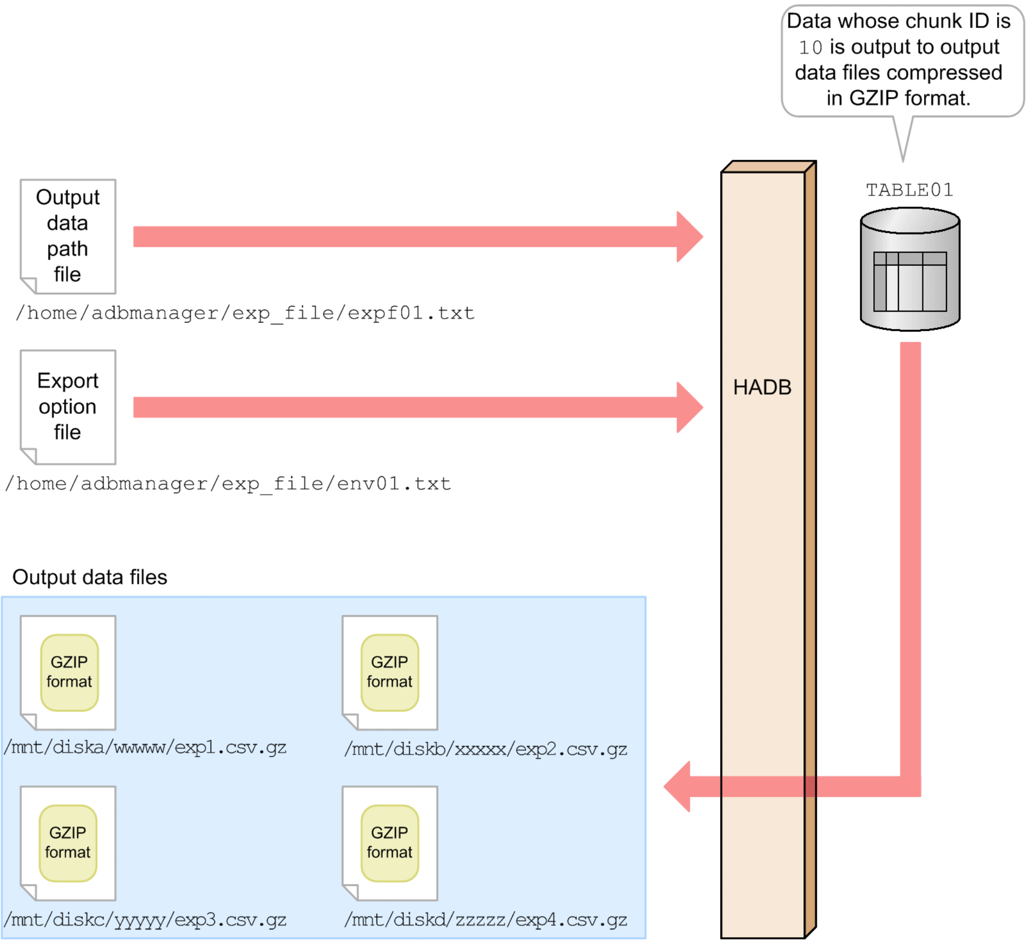
Explanation:

1. Specifies the authorization identifier of the HADB user who executes the `adbexport` command.
2. Specifies the password for the authorization identifier that is specified in the `-u` option.
3. Specifies the absolute path name of the export option file created in (2) Creating an export option file.
4. Specifies the absolute path name of the SQL statement file created in (3) Creating an SQL statement file.
5. Specifies the absolute path name of the output data path file created in (1) Creating an output data path file.

14.3.4 Exporting data for specific chunk IDs to files compressed in GZIP format

In this example, table data is exported under the following conditions:

- Data for the chunk ID `10` is exported from multi-chunk table `TABLE01`.
- The export results are output to four output data files.
- Compress output data files in GZIP format.


Data whose chunk ID is 10 is output to output data files compressed in GZIP format.
TABLE01
Output data path file
/home/adbmanager/exp_file/expf01.txt
Export option file
/home/adbmanager/exp_file/env01.txt
HADB
Output data files
GZIP format
/mnt/diska/wwwww/exp1.csv.gz
GZIP format
/mnt/diskb/xxxxx/exp2.csv.gz
GZIP format
/mnt/diskc/yyyyy/exp3.csv.gz
GZIP format
/mnt/diskd/zzzzz/exp4.csv.gz

This example uses the following procedure to execute the `adbexport` command.

Procedure

1. Create an output data path file.
2. Create an export option file.
3. Execute the `adbexport` command.

Using examples, the following subsections explain each of these steps.

(1) Creating an output data path file

In this example, an output data path file is created, in which paths for all the output data files are specified.

Example specification of an output data path file

```
/mnt/diska/wwwww/exp1.csv.gz
/mnt/diskb/xxxxx/exp2.csv.gz
/mnt/diskc/yyyyy/exp3.csv.gz
/mnt/diskd/zzzzz/exp4.csv.gz
```

Explanation:

When exporting to a file containing table data compressed in GZIP format, we recommend that you specify the file extension `.gz` at the end of the absolute path of the output data file. Specifying the file extension `.gz` at the end of the path makes it easier to identify the exported file as a GZIP-format file.

(2) Creating an export option file

In this example, an export option file is created, in which export options are specified.

Example specification of an export option file

```
set adb_export_rthd_num = 3            ...1
set adb_export_scan_buff_size = 64     ...2
```

Explanation:

1. Specifies the number of processing real threads to be used for executing the `adbexport` command.
2. Specifies the size of the buffer to be used for retaining table data search results.

(3) Executing the adbexport command

```
adbexport -u ADBUSER01                                ...1
          -p '#HelloHADB_01'                          ...2
          -z /home/adbmanager/exp_file/env01.txt      ...3
          -n TABLE01                                  ...4
          -c 10                                       ...5
          --compress GZIP                             ...6
          /home/adbmanager/exp_file/expf01.txt        ...7
```

Explanation:

1. Specifies the authorization identifier of the HADB user who executes the `adbexport` command.
2. Specifies the password for the authorization identifier that is specified in the `-u` option.
3. Specifies the absolute path name of the export option file created in (2) Creating an export option file.
4. Specifies the table to be processed.
5. Specifies the chunk ID.
6. Specifies the option to compress the output data file in GZIP format.
7. Specifies the absolute path name of the output data path file created in (1) Creating an output data path file.

14.4 Return codes

The following table lists and describes the return codes of the `adbexport` command.

Table 14-4: Return codes of the adbexport command

Return code	Description
`0`	The `adbexport` command terminated normally.
`4`	The `adbexport` command terminated normally, but a warning message was issued. Check the displayed message or the message output to a message log file and take appropriate action.
`8`	One of the following errors occurred: • The specified `adbexport` command was not executed because it was invalid. • An error occurred during execution of the `adbexport` command. For the cause of the error and corrective action to take, see the displayed error message or the message that has been output to a message log file.

14.5 Format of output data files (CSV format)

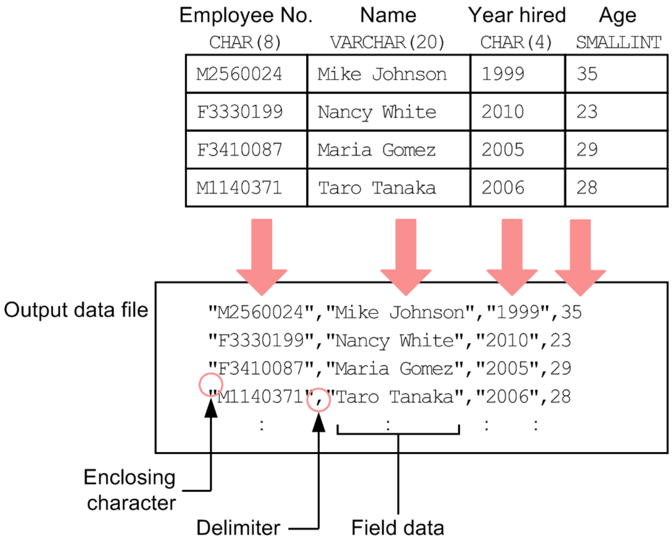
This section explains the format of output data files.

For details about the compression format of output data files, see 14.6 Compression format of output data files.

14.5.1 CSV format specifications

In the CSV format, field data is separated by delimiters. The following figure shows an example of the CSV format.

Figure 14-3: Example of the CSV format



Employee No. CHAR(8)	Name VARCHAR(20)	Year hired CHAR(4)	Age SMALLINT
M2560024	Mike Johnson	1999	35
F3330199	Nancy White	2010	23
F3410087	Maria Gomez	2005	29
M1140371	Taro Tanaka	2006	28

Output data file

```
"M2560024","Mike Johnson","1999",35
"F3330199","Nancy White","2010",23
"F3410087","Maria Gomez","2005",29
"M1140371","Taro Tanaka","2006",28
     :           :            :     :
```

Explanation:

- Each row of data forms a row of data in the output data file.
- The order of the columns in the table is the same as the order of the field data that is output to the output data file.
- Character string data, datetime data, and binary data are enclosed between the enclosing character and then output. The enclosing character can be specified in the `-k` option in the `adbexport` command.

  If `none` is specified for the `-k` option, no enclosing character is output.
- The delimiter is output between field data items. The delimiter can be specified in the `-s` option in the `adbexport` command.

14.5.2 Output rules for output data

- The character encoding of the data stored in the table is used for output to the output data files.
- Each row of data is exported to one row in the file.
- The line break code is output at the end of each line. The line break code is `X'0A'` (`LF`).
- The maximum length of a row of output data is 512 megabytes (536,870,912 bytes). If a row exceeds 512 megabytes, a file output error will occur.
- Character string data, datetime data, and binary data are enclosed between the enclosing character and then output.

- The delimiter is output between field data items.
- If the `--with-column-name` option is specified, the column name is output in the first line in the file. The output format of each column name is the same as that of `VARCHAR` type data.

14.5.3 Output format for field data

The table below describes the output format for field data by data type.

In the case of the null value, no data is output regardless of the data type. The plus sign (+) is not output for numeric data.

Table 14-5: Output format for field data

No.	Data type		Output format for field data
1	Numeric data	`INTEGER`	**Output format** • An integer is output. • The plus sign (+) is not output for a positive number. The minus sign (-) is output for a negative number. • No data is output for the null value. **Output examples** • `+1234 → 1234` • `0 → 0` • `-1234 → -1234`
2		`SMALLINT`	
3		`DECIMAL`	**Output format** • Data is output in the format *integer*`[.`*decimal*`]`. Integer part: As many digits as there are in the data are output. However, if the scaling and precision are the same, a single `0` is output. Decimal places: If the scaling is `0`, the decimal point and the digits following it are not output. If the scaling is `1`, as many digits as there are in the data are output to the fraction part. However, if the fraction part consists of only zeros, a single `0` is output. • The plus sign (+) is not output for a positive number. The minus sign (-) is output for a negative number. • No data is output for the null value. **Output examples** • `+123.45 → 123.45` • `-.123 → -0.123` • `+123. → 123` • `+123.45000 → 123.45` • `+123.00000 → 123.0`
4		`DOUBLE PRECISION`	**Output format** • Data is output in the format *mantissa*`E`*exponent*. Mantissa part: As many digits as there are in the data are output for both the mantissa and the exponent. However, if the fraction part consists of only zeros, a single `0` is output. Exponent part: As many digits as there are in the data are output. • The plus sign (+) is not output for a positive number. The minus sign (-) is output for a negative number. The plus sign (+) is not output for the exponent. • No data is output for the null value.

No.	Data type		Output format for field data
			Output examples • `+1.1234567890000000E-09` →`1.123456789E-9` • `+0.0000000000000000E+00` →`0.0E0` • `-0.1230000000000000E-123` →`-0.123E-123`
5	Character string data	`CHAR`	**Output format** • A character string enclosed in the enclosing character is output. • Any spaces to the right of the data are deleted. If the data consists of only spaces, a single space is output. • No data is output for the null value. **Output example** • `ABCD`→`"ABCD"` This example uses the double quotation mark (`"`) as the enclosing character.
6		`VARCHAR`	**Output format** • A character string enclosed in the enclosing character is output. • Data with the length of real data is output. • For data with a length of zero bytes, only two instances of the enclosing character are output. • No data is output for the null value. **Output examples** • `ABCD`→`"ABCD"` • `(`*data-with-a-length-of-zero-bytes*`)` →`""` This example uses the double quotation mark (`"`) as the enclosing character.
7	Datetime data	`DATE`	**Output format** • A date enclosed in the enclosing character is output. • The date part is output in the format *YYYY*-*MM*-*DD*. The hyphen delimits the year (*YYYY*), month (*MM*), and date (*DD*). • No data is output for the null value. **Output example** • `2013-06-10`→`"2013-06-10"` This example uses the double quotation mark (`"`) as the enclosing character.
8		`TIME`	**Output format** • A date and time enclosed between the enclosing character is output. • The time is output in the format *hh*`:`*mm*`:`*ss*`.`*nn....n*. The hour (*hh*), minute (*mm*), and second (*ss*) are delimited with the colon (`:`). If the fractional seconds precision is 1 or greater, the fraction of a second is displayed as `.`*nn....n*. • Any trailing zeros are deleted from the fraction of a second. If the fraction of a second consists of all zeros, only one zero is output. • No data is output for the null value. **Output examples** • `11:03:58`→`"11:03:58"` • `15:36:08.580`→`"15:36:08.58"` • `08:03:44.000000`→`"08:03:44.0"` This example uses the double quotation mark (`"`) as the enclosing character.
9		`TIMESTAMP`	**Output format** • A date and time enclosed in the enclosing character is output. • A date and time are output in the format *YYYY*-*MM*-*DD*Δ*hh*`:`*mm*`:`*ss*`.`*nn....n*.

No.	Data type		Output format for field data
			The hyphen delimits the year (*YYYY*), month (*MM*), and date (*DD*), followed by a single-byte space (Δ), and then the hour (*hh*), minute (*mm*), and second (*ss*) follow, delimited with the colon (`:`). If the fractional seconds precision is 1 or greater, the fraction of a second is displayed as `.`*nn....n*. • Any trailing zeros are deleted from the fraction of a second. If the fraction of a second consists of all zeros, only one zero is output. • No data is output for the null value. **Output examples** • `2013-06-10 11:03:58` → `"2013-06-10 11:03:58"` • `2014-07-15 10:11:28.330` → `"2014-07-15 10:11:28.33"` • `2014-07-08 07:12:44.000` → `"2014-07-08 07:12:44.0"` This example uses the double quotation mark (`"`) as the enclosing character.
10	Binary data	`BINARY`	**Output format** • Binary data in hexadecimal format that is enclosed in the enclosing character is output. • No data is output for the null value. **Output example** • `12345678` → `"12345678"` • `90ABCDEF` → `"90ABCDEF"` This example uses the double quotation mark (`"`) as the enclosing character.
11		`VARBINARY`	**Output format** • Binary data in hexadecimal format that is enclosed in the enclosing character is output. • For data with a length of zero bytes, only two instances of the enclosing character are output. • No data is output for the null value. **Output example** • `12345678` → `"12345678"` • `90ABCDEF` → `"90ABCDEF"` • `(`*data-with-a-length-of-zero-bytes*`)` → `""` This example uses the double quotation mark (`"`) as the enclosing character.

Note

- If you want to output datetime data in the format *YYYY*`/`*MM*`/`*DD*, specify a `CONVERT` scalar function in the selection expression in the `SELECT` statement that is specified in the SQL statement file. For details about the `CONVERT` scalar function, see *Scalar Functions* in the manual *HADB SQL Reference*.

  Example 1: Exporting date data of the `DATE` type in column `C1` in the format *YYYY*`/`*MM*`/`*DD*

  `SELECT CONVERT("C1",CHAR(10),'YYYY/MM/DD'), "C2" FROM "T1"`

  Example 2: Exporting time stamp data of the `TIMESTAMP` type in column `C1` in the format *YYYY*`/`*MM*`/`*DDhh*`:`*mm*`:`*ss*

  `SELECT CONVERT("C1",CHAR(19),'YYYY/MM/DD HH:MI:SS'), "C2" FROM "T1"`

- If you want to output binary data in binary format enclosed between enclosing characters, specify a `BIN` scalar function in the selection expression in the `SELECT` statement that is specified in the SQL statement file. For details about the `BIN` scalar function, see *Scalar Functions* in the manual *HADB SQL Reference*.

Example: If you want to output the binary data in column C1 in binary format

```
SELECT BIN("C1") FROM "T1"
```

14.6 Compression format of output data files

This section explains the compression format of output data files.

The `adbexport` command can be used to export table data by compressing the output data file (CSV format) in the following compression format.

▪ **Compression format of output data files**

- GZIP format

You can obtain output data files (CSV format) compressed in GZIP format by executing the `adbexport` command with the `--compress GZIP` option specified.

> **Note**
>
> For details about the format of output data files (CSV format), see 14.5 Format of output data files (CSV format).

The data of output data files compressed in GZIP format (CSV format) can be searched by using the `ADB_CSVREAD` function. For details about the `ADB_CSVREAD` function, see *ADB_CSVREAD function* in *System-defined functions* in *Constituent Elements* in the manual *HADB SQL Reference*.

The data of output data files compressed in GZIP format (CSV format) can be imported by using the `adbimport` command. Output data files compressed in GZIP format (CSV format) can be used as input data files compressed in GZIP format (CSV format). For details about the compression format of input data files, see 17.7 Compression format of input data files.

> **Note**
>
> Output data files compressed in GZIP format can be decompressed by using the following OS commands:
>
> - `gzip -d` command
> - `gunzip` command
> - `zcat` command

14.7 Handling abnormal termination of the adbexport command

If the `adbexport` command terminates abnormally, check if the following messages have been output:

- `KFAA50225-E`
- `KFAA96606-E`

If these messages have been output, errors were detected in the specified command options or export options. Correct all errors in options and re-execute the `adbexport` command.

If neither of these messages was output, eliminate the cause of the abnormal termination and re-execute the `adbexport` command.

Note that the `adbexport` command does not change the contents of tables or indexes. Therefore, there is no need to restore tables or indexes if the `adbexport` command terminates abnormally.

14.8 Notes

1. Do not perform read or write operations on the output data files (by using other applications) while the adbexport command is executing.
2. The adbexport command can export the following types of tables:
    - Base tables for which the EXPORT TABLE privilege has been granted
    - Viewed tables for which the EXPORT TABLE privilege has been granted
    - Dictionary tables
    - System tables
3. For details about how to improve the performance of retrieval processing by the SQL statement that is executed, see *Reducing the execution time of the adbexport command* in *Tuning to shorten command execution time* in *Tuning* in the *HADB Setup and Operation Guide*.

    When the -n option is specified, the SQL statement used for retrieving table data will be SELECT * FROM "*schema-name*"."*table-identifier*".
4. Use X'0A' (LF) or X'0D0A' (CRLF) for the line break code in files specified in the options of the adbexport command.
5. Use the adbcancel command to cancel execution of the adbexport command. Alternatively, use a method such as the adbstop --force command or the OS's kill command to forcibly terminate the server process. For details about how to use the adbcancel command to suspend an executing command, see 1.6 When suspending (forcibly terminating) an executing command.
6. Data export processing is performed by server processes. Therefore, using the **Ctrl**+**C** keys to forcibly terminate the adbexport command process will not interrupt data export processing that is being performed by server processes (this processing continues). When the data export processing is completed, the command outputs the KFAA81001-I message to the message log file.
7. When multiple output data files are specified, the number of data items output for each file varies. Consequently, a file containing zero data items might be created.
8. The output results of a retrieval with the ORDER BY clause specified are as follows:
    - Exporting data to a single output data file

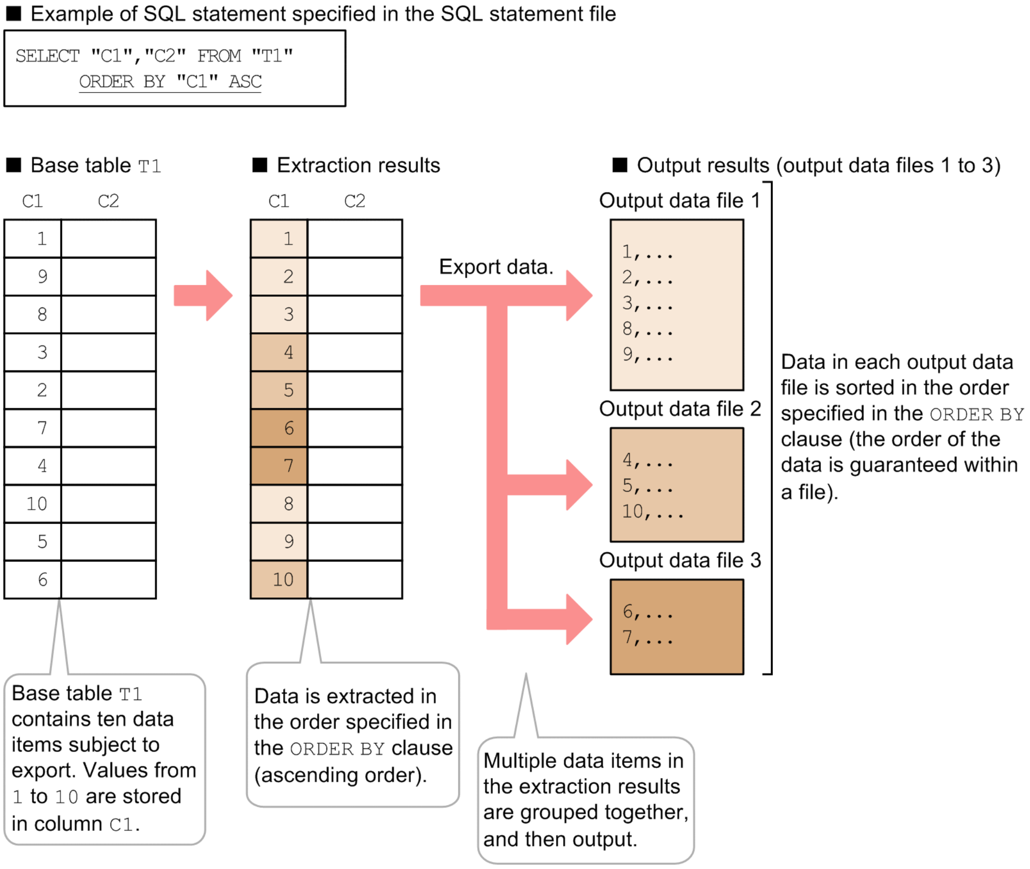
        Data is exported to the output data file in the order specified in the ORDER BY clause.
    - Exporting data to two or more output data files

        Data is extracted in the order specified in the ORDER BY clause, but data is grouped together and then output to the output data files. Within each file, data is sorted in the order specified in the ORDER BY clause.

        If the number of retrieval results is small, all data might be output in a single data file, or an output file might be created in which no records are output.

        Example:

        Base table T1 contains 10 data items subject to export and values from 1 to 10 are stored in column C1. This example uses the adbexport command to export the retrieval results to three output data files in ascending order of the data in column C1.



■ Example of SQL statement specified in the SQL statement file

```
SELECT "C1","C2" FROM "T1"
      ORDER BY "C1" ASC
```

■ Base table T1

C1	C2
1	
9	
8	
3	
2	
7	
4	
10	
5	
6	

■ Extraction results

C1	C2
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

■ Output results (output data files 1 to 3)

Data in each output data file is sorted in the order specified in the ORDER BY clause (the order of the data is guaranteed within a file).

Base table T1 contains ten data items subject to export. Values from 1 to 10 are stored in column C1.

Data is extracted in the order specified in the ORDER BY clause (ascending order).

Multiple data items in the extraction results are grouped together, and then output.

9. Make sure that there is no duplication of absolute paths for the files specified in an `adbexport` command's options. This applies to the following files:

   - Export option file (specified in the `-z` option)
   - Output data path file (specified as a command option)
   - Output data files (specified in the output data path file)

   Make sure that the names of these files differ from any file names used by any programs other than the `adbexport` command. If a duplicate file name is used, data in the file might be corrupted. Also make sure that symbolic links and relative paths are specified correctly.

10. If the `adbexport` command is executed while another command or application program is running, HADB might not be able to allocate the required number of processing real threads. If that happens, HADB waits until the required number of processing real threads can be allocated before executing the `adbexport` command. If allocation of processing real threads is placed in wait status, see the topic *Points to consider about the number of processing real threads to be used during command execution* in *Points to consider when executing commands concurrently* in *Preparing Resources* in the *HADB Setup and Operation Guide* and take appropriate action.

11. If `none` is specified for the `-k` option in the `adbexport` command, data without enclosing characters is output. For data with a length of zero bytes, nothing is output. Therefore, the data cannot be distinguished from the null value.

15 adbgetcst (Collect Cost Information)

This chapter explains the functionality provided by the `adbgetcst` command and how to use the command.

Note that the term *tables* in this chapter means base tables, and that the term *indexes* means B-tree indexes and text indexes.

15.1 Description

This section explains the functionality provided by the adbgetcst command and the threads it uses.

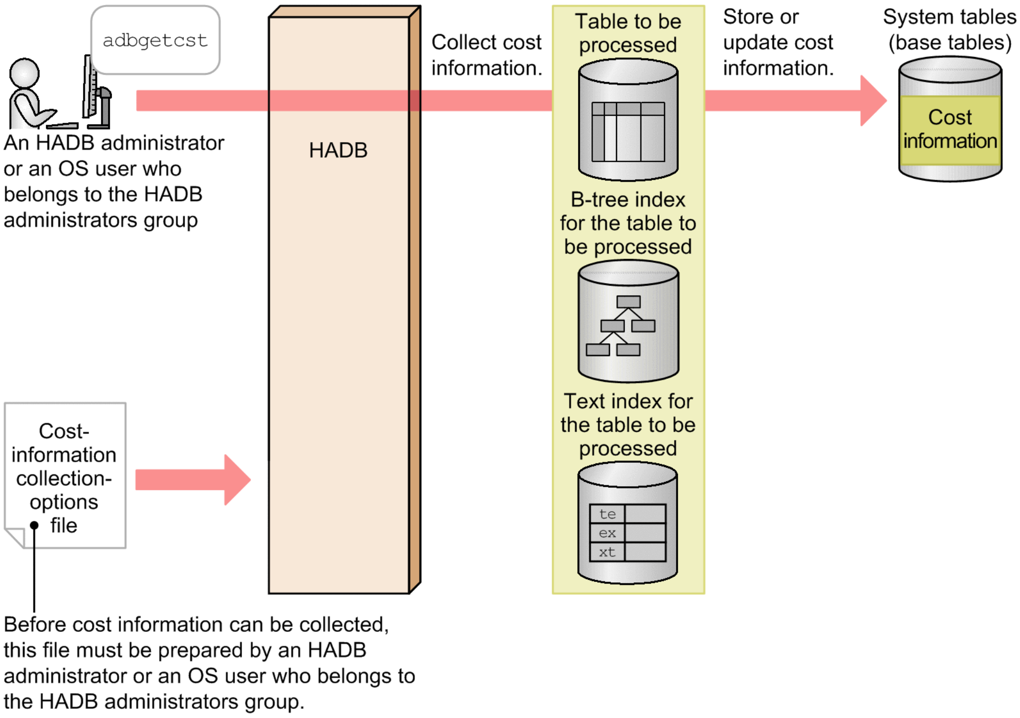
When executed, the adbgetcst command establishes a connection to the HADB server.

15.1.1 Collecting cost information

The information used by HADB servers to optimize table retrieval processing is called *cost information*. The adbgetcst command enables you to store cost information for tables and table indexes in system tables (base tables). If applicable cost information is already stored in the system tables (base tables), the adbgetcst command updates the stored cost information. These operations are referred to as *cost-information collection processing*.

The following figure provides an overview of cost-information collection processing.

Figure 15-1: Overview of cost-information collection processing


adbgetcst
An HADB administrator or an OS user who belongs to the HADB administrators group
HADB
Collect cost information.
Table to be processed
B-tree index for the table to be processed
Text index for the table to be processed
te
ex
xt
Store or update cost information.
System tables (base tables)
Cost information
Cost-information collection-options file
Before cost information can be collected, this file must be prepared by an HADB administrator or an OS user who belongs to the HADB administrators group.

Explanation:

The adbgetcst command collects cost information from tables and their indexes, and then stores the cost information in system tables (base tables). If the system tables (base tables) already contain cost information for the targeted tables or indexes, the command updates the stored cost information.

Cost information about archived chunks is not collected by cost-information collection processing.

Types of cost information

The following shows the types of cost information that can be collected by the adbgetcst command:

- Total number of rows stored in the table to be processed

  In the case of row store tables, B-tree indexes are used during collection of this information. For this reason, we recommend that you define B-tree indexes for one or more column in a row store table for which cost information is to be collected.

- Maximum and minimum values for data stored in each column in the table to be processed
  A range index or B-tree index is used during collection of this information.
- Characteristics of data (such as the degree of duplication and appearance frequency) stored in the table to be processed

The following explains the file that is used for cost-information collection processing:

- Cost-information collection-options file
  This file contains the options that specify the execution environment for the `adbgetcst` command.

We recommend that you use the `adbgetcst` command to store cost information in the system tables (base tables) after you have performed the following types of processing:

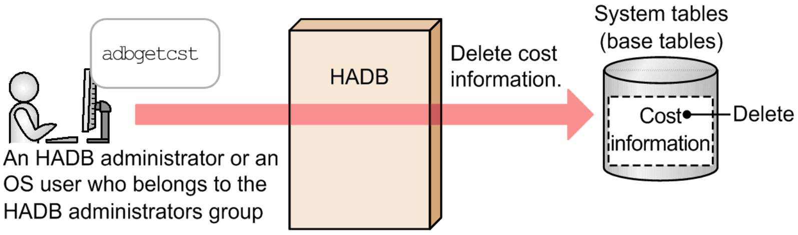
- Storing data in tables (executing the `adbimport` command)
- Rebuilding indexes (executing the `adbidxrebuild` command)
- Adding a large amount of data (with the `INSERT` statement)
- Updating a large amount of data (with the `UPDATE` statement)
- Deleting a large amount of data (with the `DELETE` statement)
- Adding, updating, or deleting a small amount of data multiple times

15.1.2 Deleting cost information

You can delete cost information stored in the system tables (base tables) by executing the `adbgetcst` command with the `-d` option specified.

The following figure shows the general procedure for deleting cost information.

Figure 15-2: Deleting cost information


adbgetcst
HADB
Delete cost information.
System tables (base tables)
Cost information
Delete
An HADB administrator or an OS user who belongs to the HADB administrators group

15.1.3 Relationships among threads used by the adbgetcst command

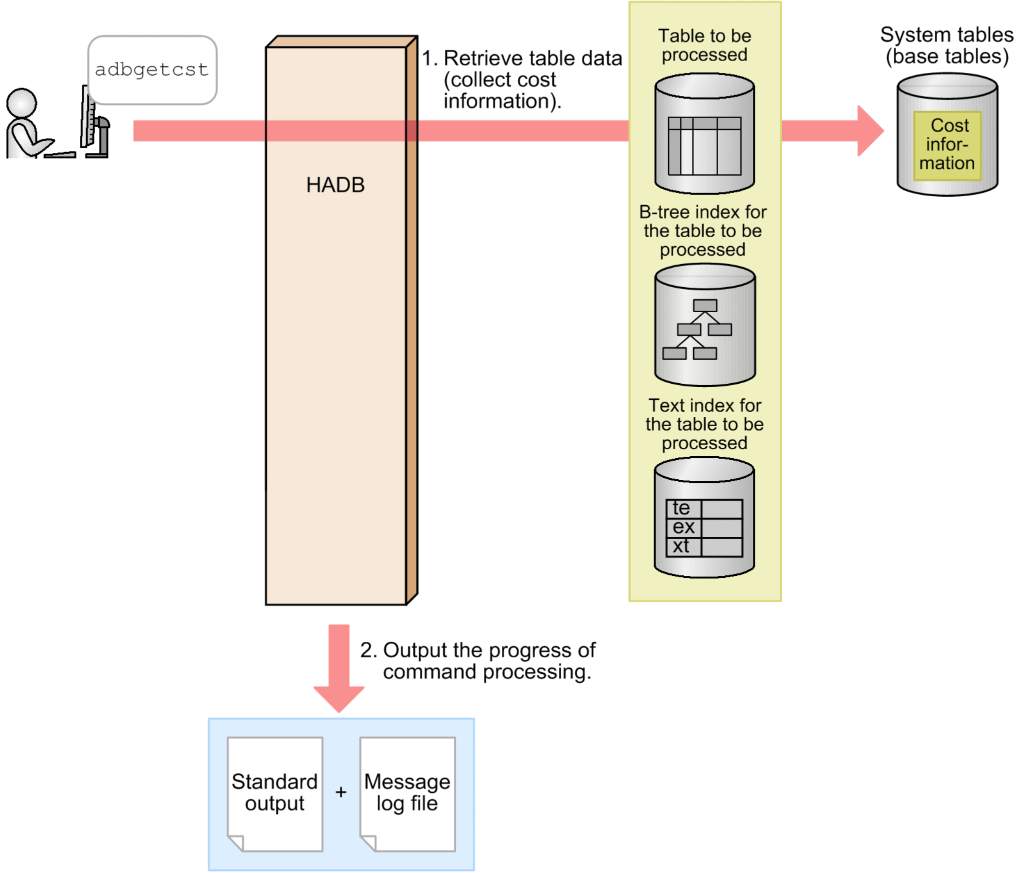
The `adbgetcst` command uses processing real threads in the following processing:

1. Retrieving table data (collecting cost information)
   One or more processing real threads are used.
2. Outputting the progress of command processing
   One processing real thread is used.

Steps 1 and 2 are performed at the same time.

The following figure shows the processing of the `adbgetcst` command using processing real threads. The preceding numbers correspond to the numbers in the figure.

Figure 15-3: Processing of the adbgetcst command using processing real threads


adbgetcst
HADB
1. Retrieve table data (collect cost information).
Table to be processed
B-tree index for the table to be processed
Text index for the table to be processed
te
ex
xt
System tables (base tables)
Cost infor-mation
2. Output the progress of command processing.
Standard output
+
Message log file

The number of processing real threads used by the `adbgetcst` command for each step is automatically calculated from the *number of processing real threads to be used by the* `adbgetcst` *command*. For details about the *number of processing real threads used by the* `adbgetcst` *command*, see the description of the adb_getcst_rthd_num cost information collection option in (2) Explanation of specification format in 15.2.2 Format of the cost-information collection option.

15.2 Specification format for the adbgetcst command

This section explains the specification format for the `adbgetcst` command and the cost information collection option.

The following OS users can execute the `adbgetcst` command:

- HADB administrator
- OS users who belong to the HADB administrators group

15.2.1 Explanation of the specification format and options

(1) Specification format

```
adbgetcst
     -u authorization-identifier
    [-p password]
    [-t table-name]
    [-d]
    [-z cost-information-collection-options-file-name]
    [-g output-interval-for-cost-information-collection-progress-message]
```

(2) Explanation of options

- `-u` *authorization-identifier*

  ~<character string>((1 to 100 bytes))

  Specifies the authorization identifier of the HADB user who executes the `adbgetcst` command. You must have the following two privileges to execute the `adbgetcst` command:

  - `CONNECT` privilege
  - `GET COSTINFO` privilege for the table to be processed

  For this option, specify an authorization identifier in the range from 1 to 100 bytes. Note that the byte count (1 to 100 bytes) does not include the double quotation marks used to enclose the authorization identifier.

  > **Important**
  >
  > If the character string used as the authorization identifier includes any lowercase letter or a backslash (`\`), make sure that you check the rules for specifying authorization identifiers. For the rules for specifying authorization identifiers, see 1.4.2 Rules for specifying authorization identifiers and passwords.

- `-p` *password*

  ~<character string>((1 to 255 bytes))

  Specifies the password for the authorization identifier that was specified in the `-u` option.

  > **Important**
  >
  > If the password includes a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), make sure that you check the rules for specifying passwords. For

the rules for specifying passwords, see 1.4.2 Rules for specifying authorization identifiers and passwords.

If the `adbgetcst` command is executed with this option omitted, a message prompting the user to enter a password is displayed. In an environment in which a password cannot be entered from the standard input, such as when the command is executed in the background, make sure that you do not omit this option.

- `-t` *table-name*

  ~<character string>

  Specifies the table name of the table for which cost information is to be collected. When the `-d` option is specified, specify the table name of a table whose cost information is to be deleted from the system tables (base tables).

  If this option is omitted, the `adbgetcst` command processes all tables defined in the schema having the authorization identifier specified by the `-u` option.

  The following rules apply to specifying this option:

  - Specify in the format *schema-name*`.`*table-identifier*.

    If the user owns the table, the schema name can be omitted. If another HADB user owns the table, the schema name must be specified.

  - For details about the specification rules that apply when a schema name or table identifier includes any lowercase letter or backslash (`\`), see 1.4.3 Table name specification rules.

  - The user can specify only a table for which the user has the `GET COSTINFO` privilege.

  - A viewed table cannot be specified.

- `-d`

  Specifies that cost information is to be deleted from the system tables (base tables).

  If there is no cost information in the system tables (base tables), the command skips deletion processing.

- `-z` *cost-information-collection-options-file-name*

  ~<OS path name>((2 to 510 bytes))

  Specifies the absolute path name of the cost-information collection-options file that contains an option for cost-information collection processing. If this option is omitted, the cost information collection option is set to its default value because no cost-information collection-options file is specified.

  For details about the cost information collection option, see 15.2.2 Format of the cost-information collection option.

- `-g` *output-interval-for-cost-information-collection-progress-message*

  ~<integer>((0 to 1,000,000,000))<<0>>(in 1,000 rows)

  Specifies the output interval for the cost information collection progress message. For example, if `2` is specified in this option, the progress message (`KFAA80205-I`) is output each time 2,000 rows of cost information are collected.

  If this option is specified and the `-t` option is omitted, the progress message is output for each table that is processed.

  If this option is omitted or `0` is specified in this option, the progress message is not output.

  This option is ignored when the `-d` option is specified.

15.2.2 Format of the cost-information collection option

(1) Specification format

```
[set adb_getcst_rthd_num = number-of-processing-real-threads-to-be-used-by-adbgetcst-
command]
[set adb_getcst_collect_time = time-for-collecting-cost-information]
```

The specification rules for this cost information collection option are the same as for the server definition. For details about the specification rules for the server definition, see *Syntax rules for the server definition* in the *HADB Setup and Operation Guide*.

(2) Explanation of specification format

- `adb_getcst_rthd_num` = *number-of-processing-real-threads-to-be-used-by-adbgetcst-command*

  ~<integer> ((2 to 4,096))

  Specifies the number of processing real threads to be used for executing the `adbgetcst` command.

  Specification of this option is recommended. The `adbgetcst` command uses the number of processing real threads specified by this option. Therefore, specify an appropriate number of processing real threads for use by the `adbgetcst` command. For details about the processing of the `adbgetcst` command that uses processing real threads and the number of processing real threads to be used for each step, see 15.1.3 Relationships among threads used by the adbgetcst command.

  If this option is omitted, the number of processing real threads to be used for executing the `adbgetcst` command is determined by the `adb_sql_exe_max_rthd_num` operand in the server definition. The following table shows the relationship between the value of this option or the `adb_sql_exe_max_rthd_num` operand in the server definition and the number of processing real threads used for executing the command.

  Table 15-1: Number of processing real threads used for executing the command

No.	Value in this option or in the server definition		Number of processing real threads used for executing the command
1	When this option is omitted	If the `adb_sql_exe_max_rthd_num` operand in the server definition is not specified	The default value of the `adb_sql_exe_max_rthd_num` operand in the server definition is used.
2		If `0` or `1` is specified in the `adb_sql_exe_max_rthd_num` operand in the server definition	--
3		If `2` or a greater value is specified for the `adb_sql_exe_max_rthd_num` operand in the server definition	The value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition is used.
4	If `2` or a greater value is specified in this option		The value specified in this option is used.

  Legend:

  --: The `adbgetcst` command results in an error.

  Note:

  For details about the `adb_sql_exe_max_rthd_num` operand in the server definition, see *Operands related to performance (set format)* in the *HADB Setup and Operation Guide*.

- `adb_getcst_collect_time` = *time-for-collecting-cost-information*

  ~<integer>((0, 5 to 1,440))<<10>> (minutes)

Specifies in minutes the time during which cost information for the table to be processed is collected.

If this option is specified, only the cost information that was collected within the specified period of time is stored in the system table. If you want to collect all cost information for the table to be processed regardless of the cost information collection time, specify `0` for this option.

If the `-t` option of the `adbgetcst` command is omitted, cost information for all tables in the schema is collected. At this time, the value of this option is applied to each table in the schema that has the authorization identifier specified in the `-u` option. If cost information collection for the table to be processed does not finish in the time specified for this option, the cost information that was collected until the specified time has passed is stored in the system table. Then, cost information collection begins for the next table to be processed.

The following table shows the relationship between the value specified for this option and cost information stored in the system table.

Table 15-2: Relationship between the value of this option and cost information stored in the system table

No.	Value of this option	Time required for collecting cost information for the table to be processed	Cost information stored in the system table
1	The option is omitted or `5` or a greater value is specified.	Longer than the time specified for this option (or the default value)	Cost information# collected until the time specified for this option (or the default value) has passed If the time specified for this option has passed during cost information collection processing (after the `KFAA80235-I` is output and until the `KFAA80236-I` message is output), the `KFAA80237-I` message is output. If cost information is to be collected for all tables in the schema, when the time specified for this option has passed, cost information collection for the next table to be processed begins. At this time, the `KFAA80237-I` message is output.
2		Equal to or shorter than the time specified for this option (or the default value)	All cost information for tables to be processed If cost information is to be collected for all tables in the schema, when cost information collection for the table to be processed is completed, cost information collection for the next table begins.
3	If `0` is specified	--	All cost information for tables to be processed If cost information is to be collected for all tables in the schema, when cost information collection for the table to be processed is completed, cost information collection for the next table begins.

Legend:

--: Not applicable. All cost information for the table to be processed is collected, regardless of the cost information collection time.

#

Compared to a case in which all cost information for the table to be processed is collected, optimization of SQL statements (results of access path selection) might be different.

15.3 Examples

This section explains by way of examples how to use the adbgetcst command to collect and delete cost information.

If the adbgetcst command terminates abnormally during execution, take appropriate action based on the explanation in 15.5 Handling abnormal termination of the adbgetcst command.

15.3.1 Collecting cost information for a specific table

This example collects cost information for table TABLE01. The absolute path name of the cost-information collection-options file is /home/adbmanager/getcst_file/env01.txt.

To collect cost information:

1. Create a cost-information collection-options file.
2. Execute the adbgetcst command.

Using an example, the following subsections explain each of these steps.

(1) Creating a cost-information collection-options file

This example creates a cost-information collection-options file and specifies the number of processing real threads to be used for executing the adbgetcst command.

Specification example of cost-information collection-options file

```
set adb_getcst_rthd_num = 10
```

(2) Executing the adbgetcst command

```
adbgetcst -u ADBUSER01                                ...1
          -p '#HelloHADB_01'                          ...2
          -t TABLE01                                  ...3
          -z /home/adbmanager/getcst_file/env01.txt   ...4
```

Explanation:

1. Specifies the authorization identifier of the HADB user who executes the adbgetcst command.
2. Specifies the password for the authorization identifier that was specified in the -u option.
3. Specifies the table name of the table for which cost information is to be collected.
4. Specifies the absolute path name of the cost-information collection-options file created in (1) Creating a cost-information collection-options file.

15.3.2 Collecting cost information for all tables

This example collects cost information for all tables in schema ADBUSER01. The absolute path name of the cost-information collection-options file is /home/adbmanager/getcst_file/env01.txt.

To collect cost information:

1. Create a cost-information collection-options file.
2. Execute the `adbgetcst` command.

Using an example, the following subsections explain each of these steps.

(1) Creating a cost-information collection-options file

This example creates a cost-information collection-options file and specifies the number of processing real threads to be used for executing the `adbgetcst` command.

Specification example of cost-information collection-options file

```
set adb_getcst_rthd_num = 10
```

(2) Executing the adbgetcst command

```
adbgetcst -u ADBUSER01                                 ...1
          -p '#HelloHADB_01'                           ...2
          -z /home/adbmanager/getcst_file/env01.txt    ...3
```

Explanation:

1. Specifies the authorization identifier of the HADB user who executes the `adbgetcst` command.
2. Specifies the password for the authorization identifier that was specified in the `-u` option.
3. Specifies the absolute path name of the cost-information collection-options file created in (1) Creating a cost-information collection-options file.

15.3.3 Deleting cost information for a specific table

This example deletes the cost information for table `TABLE01`.

(1) Executing the adbgetcst command

```
adbgetcst -u ADBUSER01                 ...1
          -p '#HelloHADB_01'           ...2
          -t TABLE01                   ...3
          -d                           ...4
```

Explanation:

1. Specifies the authorization identifier of the HADB user who executes the `adbgetcst` command.
2. Specifies the password for the authorization identifier that was specified in the `-u` option.
3. Specifies the table name of the table for which cost information is to be deleted from the system table (base table).
4. Specifies that the cost information is to be deleted.

15.3.4 Deleting cost information for all tables

This example deletes the cost information for all tables in schema ADBUSER01.

(1) Executing the adbgetcst command

```
adbgetcst -u ADBUSER01                ...1
          -p '#HelloHADB_01'          ...2
          -d                          ...3
```

Explanation:

1. Specifies the authorization identifier of the HADB user who executes the adbgetcst command.
2. Specifies the password for the authorization identifier that was specified in the -u option.
3. Specifies that the cost information is to be deleted.

15.4 Return codes

The following table lists and describes the return codes of the adbgetcst command.

Table 15-3: Return codes of the adbgetcst command

Return code	Description
0	The adbgetcst command terminated normally.
4	The adbgetcst command terminated normally, but a warning message was issued. Check the displayed message or the message output to a message log file and take appropriate action. If the KFAA50291-W message was output, cost information was not collected for one or more tables for one or more of the following reasons: • The tables to be processed were non-updatable. • The B-tree indexes, text indexes, or range indexes defined in the tables to be processed were in unfinished status. • The adbimport or adbidxrebuild command, executed on the tables to be processed, was canceled. In such a case, release the tables from non-updatable status or the applicable indexes from unfinished status, or re-execute the canceled command, and then collect cost information. For details about how to release a table from non-updatable status, see *Steps to take when a base table becomes non-updatable* in the *HADB Setup and Operation Guide*. For details about how to release B-tree indexes from unfinished status, see *Steps to take when unfinished status is applied to a B-tree index* in the *HADB Setup and Operation Guide*. For details about how to release text indexes from unfinished status, see *Steps to take when unfinished status is applied to a text index* in the *HADB Setup and Operation Guide*. For details about how to release range indexes from unfinished status, see *Steps to take when unfinished status is applied to a range index* in the *HADB Setup and Operation Guide*. To identify the canceled command, output the table summary information by specifying, in the adbdbstatus command, table for the -c option and the table to be processed for the -n option. The name of the command that was canceled is output to the Rerun_command column in the output result. For details about the adbdbstatus command, see 13. adbdbstatus (Analyze the Database Status).
8	One of the following errors occurred: • The specified adbgetcst command did not execute because it was invalid. • An error occurred while the adbgetcst command was executing. For the cause of the error and corrective action to take, see the displayed error message or the message that has been output to a message log file.

15.5 Handling abnormal termination of the adbgetcst command

This section explains how to handle abnormal termination of the `adbgetcst` command.

Because the `adbgetcst` command does not change the contents of tables or indexes, there is no need to restore tables or indexes in the event of abnormal termination of the `adbgetcst` command.

15.5.1 What to check if the adbgetcst command terminates abnormally

To check and determine the necessary action to take:

1. Check the specified command for any error.

   Check if the following messages were issued:

   - `KFAA50225-E`
   - `KFAA96606-E`

   If either of these messages was issued, an error was detected in the specified command options or in the cost-information collection option. Check and correct the command options and cost information collection option, and then re-execute the `adbgetcst` command.

   If neither of these messages was issued, go to the next step.

2. Check if the `KFAA80204-I` message was issued.

   If the `KFAA80204-I` message was issued with return code `0`, collection or deletion of cost information has been completed. No action is needed.

   If the `KFAA80204-I` message was not issued with return code `0`, the appropriate action must be taken according to the processing that terminated abnormally. The following table shows the relationship between the processing resulting in abnormal termination and the action to be taken.

   Table 15-4: Relationship between the executed processing and the action to be taken

No.	Processing resulting in abnormal termination	Scope of processing	Action
1	Collection of cost information	Specific table	See 15.5.2 If the command terminated abnormally while collecting cost information for a specific table.
2		All tables	See 15.5.3 If the command terminated abnormally while collecting cost information for all tables.
3	Deletion of cost information	Specific table	See 15.5.4 If the command terminated abnormally while deleting cost information for a specific table.
4		All tables	See 15.5.5 If the command terminated abnormally while deleting cost information for all tables.

15.5.2 If the command terminated abnormally while collecting cost information for a specific table

To troubleshoot if the command with the `-t` option specified terminated abnormally while collecting cost information for a specific table:

1. Check if the `KFAA80285-I` message was issued.

If the KFAA80285-I message (indicating that cost information for a table was collected) was issued, the cost information has been collected. No action is needed.

If the KFAA80285-I message was not issued, go to the next step.

2. Check the date and time cost information was collected.

   Retrieve the system tables to check the date and time cost information was collected for the table specified in the -t option and for any indexes defined for that table.

   The following SELECT statement checks the names of all tables for which cost information was collected and the dates and times the cost information was collected:

   ```
   SELECT "TABLE_SCHEMA","TABLE_NAME","COLLECT_TIME"
        FROM "MASTER"."STATUS_TABLES"
   ```

   The following SELECT statement checks the names of all indexes for which cost information was collected and the dates and times the cost information was collected:

   ```
   SELECT "TABLE_SCHEMA","INDEX_NAME","COLLECT_TIME"
        FROM "MASTER"."STATUS_INDEXES"
   ```

   > **Important**
   >
   > Issue the COMMIT statement immediately after you retrieve the system tables. However, if you terminate the adbsql command after you retrieve the system tables, there is no need to execute the COMMIT statement.

   If the dates and times the cost information was collected for the tables and indexes are after the date and time the adbgetcst command was executed, the cost information has been collected. In this case, no action is needed.

   If the dates and times the cost information was collected for the tables and indexes are before the date and time the adbgetcst command executed or no cost information is stored in the system tables (base tables), go to the next step.

3. Check the message that was issued.

   Eliminate the cause of the error according to the information provided in the message.

4. Re-execute the adbgetcst command.

   Re-execute the adbgetcst command with the same options.

15.5.3 If the command terminated abnormally while collecting cost information for all tables

To troubleshoot if the command with the -t option omitted terminated abnormally while collecting cost information for all the tables in a schema:

1. Check if the KFAA80285-I message was issued.

   If the KFAA80285-I message (indicating that cost information for a table was collected) was issued, the cost information has been collected. No action is needed.

   The KFAA80285-I message is issued for each table processed. Therefore, you must check whether the KFAA80285-I message was issued for every table in the schema. If the KFAA80285-I message was not issued for a table, go to the next step.

2. Check the date and time cost information was collected.

Retrieve the system tables to check the dates and times cost information was collected for all the tables in the schema and for the indexes defined for those tables.

The following `SELECT` statement checks the names of all tables for which cost information was collected and the dates and times the cost information was collected:

```
SELECT "TABLE_SCHEMA","TABLE_NAME","COLLECT_TIME"
     FROM "MASTER"."STATUS_TABLES"
```

The following `SELECT` statement checks the names of all indexes for which cost information was collected and the dates and times the cost information was collected:

```
SELECT "TABLE_SCHEMA","INDEX_NAME","COLLECT_TIME"
     FROM "MASTER"."STATUS_INDEXES"
```

> **Important**
>
> Issue the `COMMIT` statement immediately after you retrieve the system tables. However, if you terminate the `adbsql` command after you retrieve the system tables, there is no need to execute the `COMMIT` statement.

If the dates and times the cost information was collected for the tables and indexes are after the date and time the `adbgetcst` command was executed, the cost information has been collected. In this case, no action is needed.

If the dates and times the cost information was collected for the tables and indexes are before the date and time the `adbgetcst` command was executed or no cost information is stored in the system table (base table), go to the next step.

3. Check the message that was issued.

   Eliminate the cause of the error according to the information provided in the message.

4. Re-execute the `adbgetcst` command.

   Re-execute the `adbgetcst` command with the `-t` option specifying the table name of a table for which cost information was not collected. This also applies to a table with an index defined for which cost information was not collected. Only one table name can be specified by the `-t` option. Therefore, if cost information was not collected for more than one table, you must execute the `adbgetcst` command as many times as there are tables for which cost information needs to be collected.

   If cost information was not collected for any of the tables in the schema, re-execute the `adbgetcst` command with the `-t` option omitted.

15.5.4 If the command terminated abnormally while deleting cost information for a specific table

To troubleshoot if the command with the `-t` option specified terminated abnormally while deleting cost information for a specific table:

1. Check if the `KFAA80287-I` message was issued.

   If the `KFAA80287-I` message (indicating that cost information for a table was deleted) was issued, the cost information has been deleted. No action is needed.

   If the `KFAA80287-I` message was not issued, go to the next step.

2. Check the message that was issued.

   Eliminate the cause of the error according to the information provided in the message.

3. Re-execute the `adbgetcst` command.

   Re-execute the `adbgetcst` command with the same options.

15.5.5 If the command terminated abnormally while deleting cost information for all tables

To troubleshoot if the command with the `-t` option omitted terminated abnormally while deleting cost information for all the tables in a schema:

1. Check if the `KFAA80287-I` message was issued.

   The `KFAA80287-I` message (indicating that cost information for a table was deleted) is issued for each table for which cost information was deleted. Therefore, you must check whether the `KFAA80287-I` message was issued for every table in the schema. If the `KFAA80287-I` message was issued for every table, the cost information has been deleted. In this case, no action is needed.

   If the `KFAA80287-I` message was not issued for any of the tables, go to the next step.

2. Check the message that was issued.

   Eliminate the cause of the error according to the information provided in the message.

3. Re-execute the `adbgetcst` command.

   Re-execute the `adbgetcst` command with the same options.

15.6 Notes

1. Cost information collected on an HADB server whose version is earlier than **04-03** cannot be used on HADB servers whose version is **04-03** or later. Therefore, if the HADB server has been upgraded from a version earlier than **04-03** to **04-03** or later, you must collect cost information again by using the `adbgetcst` command.
2. If the `adbgetcst` command is executed while another command or application program is running, HADB might not be able to allocate the required number of processing real threads. If that happens, HADB waits until the required number of processing real threads can be allocated before executing the `adbgetcst` command. If allocation of processing real threads is placed in wait status, see the topic *Points to consider about the number of processing real threads to be used during command execution* in *Points to consider when executing commands concurrently* in *Preparing Resources* in the *HADB Setup and Operation Guide* and take appropriate action.
3. Cost-information collection processing for a table is skipped and the `KFAA50291-W` message is issued in the following cases:
   - The table to be processed is non-updatable.
   - The B-tree indexes, text indexes, or range indexes defined in the tables to be processed are in unfinished status.
   - The `adbimport` or `adbidxrebuild` command, executed on the tables to be processed, was canceled.

   When deleting cost information, the command deletes the cost information from the system table (base table) regardless of the status of the tables or the status of any B-tree, text, or range indexes defined for those tables.
4. When cost information is collected, the HADB server uses the `SELECT` statement to retrieve the target tables. Therefore, we recommend that you execute the command in an environment in which out-of-order execution is applied. For details about environments in which out-of-order execution is applied, see *Database retrieval method (out-of-order execution)* in the *HADB Setup and Operation Guide*.
5. Make sure that the cost-information collection-options file and the path of its storage directory differ from any of the files and directories used by a program other than the `adbgetcst` command. If there is any duplication, the data in the file might be lost. This also applies to symbolic link and relative path specifications.
6. You can use the `adbcancel` command to cancel execution of the `adbgetcst` command. For details about how to use the `adbcancel` command to suspend an executing command, see 1.6 When suspending (forcibly terminating) an executing command.
7. Cost-information collection and deletion processing is handled by server processes. Therefore, using the **Ctrl**+**C** keys to forcibly terminate the `adbgetcst` command process will not interrupt cost-information collection or deletion processing that is being performed by server processes (such processing continues). When the collection and deletion processing is completed, the command outputs the `KFAA81001-I` message to the message log file.
8. For information on reducing this command's execution time, see the topic *Reducing the execution time of the adbgetcst command* in *Tuning to shorten command execution time* in *Tuning* in the *HADB Setup and Operation Guide*.

16 adbidxrebuild (Rebuild Indexes)

This chapter explains the functionality provided by the `adbidxrebuild` command and how to use the command.

Note that the term *tables* in this chapter means base tables.

16.1 Description

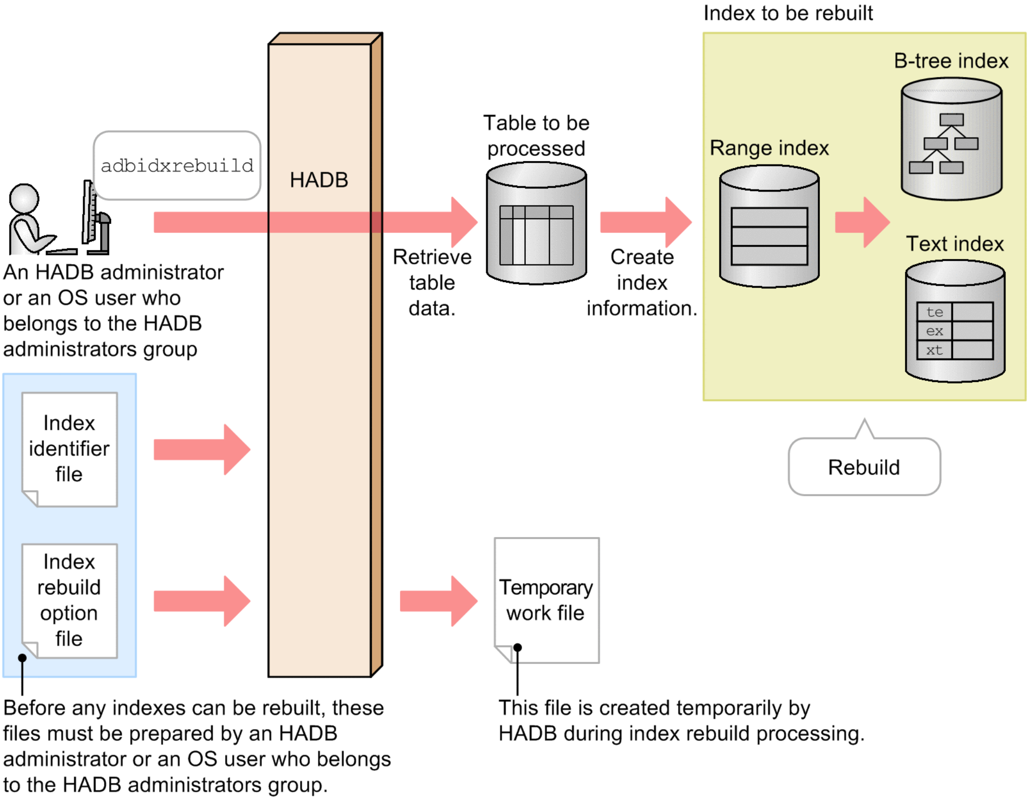
This section explains the functionality provided by the adbidxrebuild command and the threads it uses.

When executed, the adbidxrebuild command establishes a connection to the HADB server.

16.1.1 Functional overview

The adbidxrebuild command retrieves table data and creates index information. The adbidxrebuild command uses that index information to rebuild the indexes that have been defined for the table. This series of processing steps is called *index rebuild processing*. The following figure provides an overview of index rebuild processing.

Figure 16-1: Overview of index rebuild processing


Index to be rebuilt
B-tree index
Table to be processed
Range index
adbidxrebuild
HADB
Text index
te
ex
xt
Retrieve table data.
Create index information.
An HADB administrator or an OS user who belongs to the HADB administrators group
Index identifier file
Rebuild
Index rebuild option file
Temporary work file
Before any indexes can be rebuilt, these files must be prepared by an HADB administrator or an OS user who belongs to the HADB administrators group.
This file is created temporarily by HADB during index rebuild processing.

Explanation:

When the adbidxrebuild command is executed, table data is retrieved and index information is created. The indexes defined for the table are rebuilt based on that information. If B-tree indexes, text indexes, and range indexes are defined for the table, the range indexes are rebuilt first, and then the B-tree and text indexes are rebuilt.

You can release indexes from unfinished status by using the adbidxrebuild command to rebuild them.

The following explains the files:

- **Index identifier file**

  This file is used to specify the identifiers of the indexes that are to be rebuilt by the adbidxrebuild command.

- **Index rebuild option file**

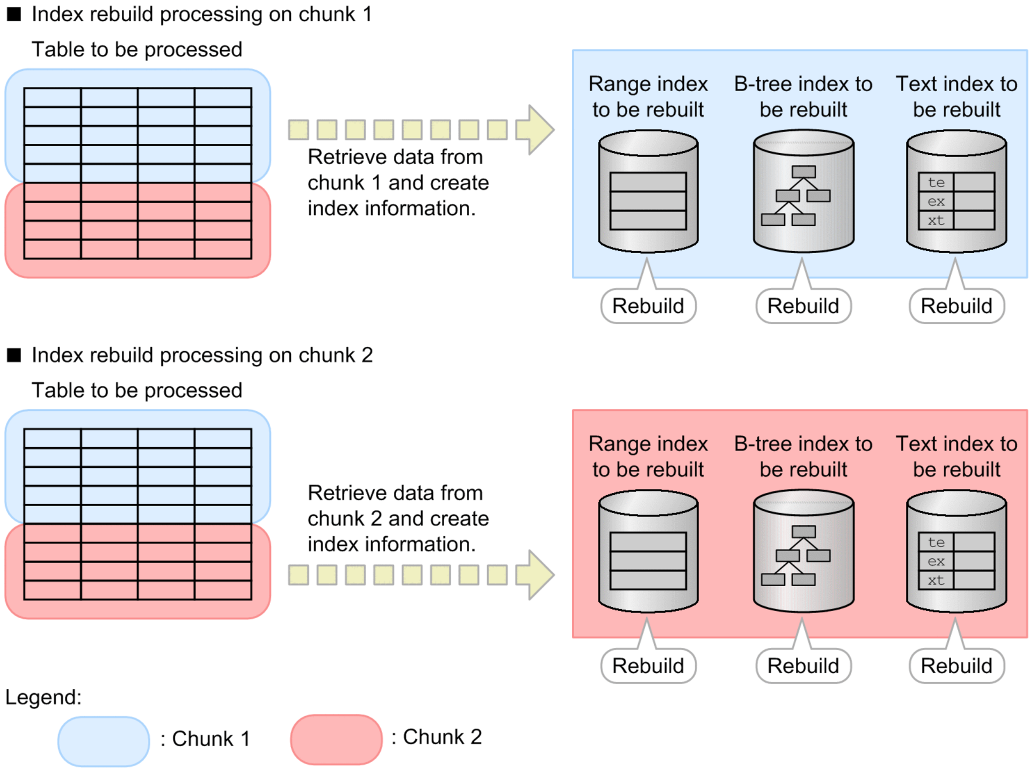
  This file contains options that are to be used to set up an execution environment for the adbidxrebuild command.

- **Temporary work files**
  These are work files that are created temporarily during index rebuild processing. They are deleted automatically when index rebuild processing terminates normally.

Note

When the `adbidxrebuild` command is executed on a multi-chunk table, index rebuild processing is performed separately for each chunk. A message displays the chunk ID of the chunk for which index rebuild processing is underway. Note that index rebuild processing is not performed for archived chunks because they have no indexes.

Figure 16-2: Index rebuild processing on a multi-chunk table


■ Index rebuild processing on chunk 1
Table to be processed
Retrieve data from chunk 1 and create index information.
Range index to be rebuilt
B-tree index to be rebuilt
Text index to be rebuilt
te
ex
xt
Rebuild
Rebuild
Rebuild
■ Index rebuild processing on chunk 2
Table to be processed
Retrieve data from chunk 2 and create index information.
Range index to be rebuilt
B-tree index to be rebuilt
Text index to be rebuilt
te
ex
xt
Rebuild
Rebuild
Rebuild
Legend:
: Chunk 1
: Chunk 2

16.1.2 Benefits of rebuilding indexes

The `adbidxrebuild` command enables you to rebuild indexes that are in the following statuses:

- Indexes in unfinished status
- Unique indexes on which a uniqueness constraint violation occurred
- Indexes in normal status

The following subsections describes the benefits of rebuilding these indexes.

(1) Rebuilding indexes in unfinished status (releasing indexes from unfinished status)

When a new index is defined for a table to which segments for storing rows have been assigned, the newly defined index is placed in unfinished status. Retrieval processing using an index in unfinished status results in an error.

When the `adbidxrebuild` command is executed on a table for which any indexes that are in unfinished status are defined, the indexes are released from unfinished status, thereby enabling them to be used for retrieval processing.

Figure 16-3: Rebuilding indexes in unfinished status

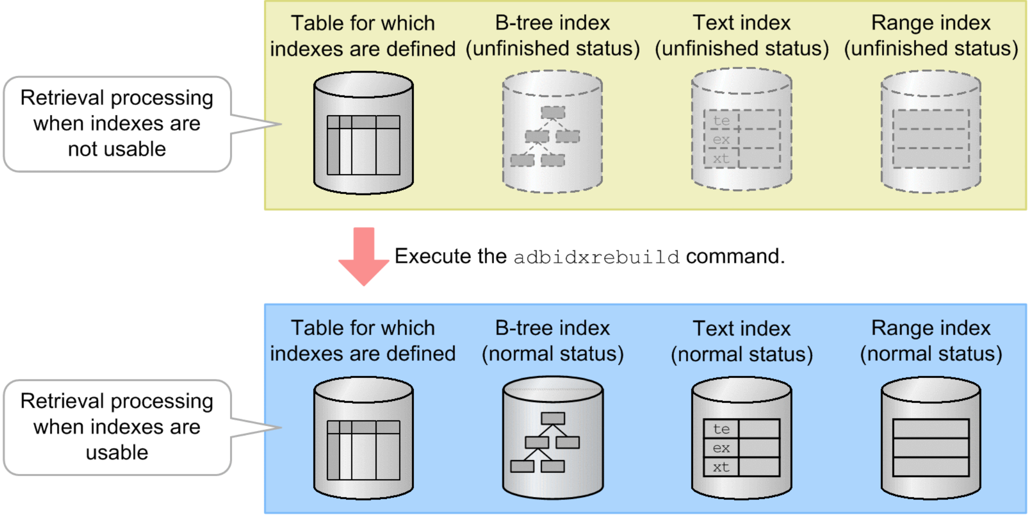


Table for which indexes are defined
B-tree index (unfinished status)
Text index (unfinished status)
Range index (unfinished status)
Retrieval processing when indexes are not usable
te ex xt
Execute the adbidxrebuild command.
Table for which indexes are defined
B-tree index (normal status)
Text index (normal status)
Range index (normal status)
Retrieval processing when indexes are usable
te ex xt

For details about the status in which segments for storing rows are assigned, see *Notes on defining B-tree indexes (unfinished status of B-tree indexes)* in the *HADB Setup and Operation Guide*.

(2) Rebuilding unique indexes on which a uniqueness constraint violation occurred (releasing the uniqueness constraint violation)

If a unique index is in violation of the uniqueness constraint (*uniqueness constraint violation*), retrieval efficiency might be lower than when the uniqueness constraint is satisfied.

In such a case, you can release the uniqueness constraint violation by deleting the rows causing the uniqueness constraint violation and then using the `adbidxrebuild` command to rebuild the unique index.

For details about how to release a uniqueness constraint violation, see *Steps to take when the uniqueness constraint is violated (when the KFAA61205-W message is output)* in *Problems related to B-tree indexes* in *Troubleshooting* in the *HADB Setup and Operation Guide*.

(3) Rebuilding indexes in normal status

You can also use the `adbidxrebuild` command to rebuild indexes that are in normal status (indexes that are not in unfinished status or on which a uniqueness constraint violation occurred). You can expect the following benefits from rebuilding such indexes.

For B-tree indexes

If you rebuild B-tree indexes (including those corresponding to the primary key) defined for a table in which rows have been updated, added, or deleted repeatedly, you can obtain the following benefits:

- Data that has become invalid as a result of updating or deleting rows can be deleted from the B-tree indexes. This reduces the amount of DB area required to store the B-tree indexes.
- Unused areas in index pages that have been used for adding or updating rows can be reallocated. This reduces the frequency of index page splitting. For details about the unused areas in index pages, see *Allocating an unused area inside a B-tree index page (PCTFREE)* in *Designing a B-tree index* in *Designing a Database* in the *HADB Setup and Operation Guide*.

For text indexes

If you rebuild the text indexes defined for a table in which rows have been updated, added, or deleted repeatedly, you can obtain the following benefits:

- You can delete from the text indexes links to data pages that contain no actual data, and that were generated as a result of updating or deleting rows. This deletion improves the performance of retrievals that use text indexes. It also reduces the amount of DB area required to store the text indexes.
- Unused areas in index pages that have been used to add or update rows can be reallocated. This reduces the frequency of index page splitting. For details about the unused areas in index pages, see *Allocating an unused area inside a text index page (PCTFREE)* in *Designing a text index* in *Designing a Database* in the *HADB Setup and Operation Guide*.

For range indexes

If you rebuild the range indexes defined for a table in which rows have been updated or deleted repeatedly, you can obtain the following benefits:

- The ranges of the data included in the range indexes become smaller, thereby improving retrieval performance.
- The amount of DB area required for storing the range indexes can be reduced because range information that is not needed for the range indexes is deleted.

16.1.3 Optional functions of the adbidxrebuild command

The `adbidxrebuild` command provides the following optional functions:

- Re-execute facility
- Inheritance of index creation processing from the `adbimport` command

(1) Re-execute facility

When the `adbidxrebuild` command is re-executed after having terminated abnormally, it resumes the index rebuild processing from the location where the index contents are guaranteed from the previous session of index rebuild processing. This reduces the amount of time required for re-execution of index rebuild processing.

Use of the re-execute facility is appropriate when the following conditions are satisfied:

- The indexes to be rebuilt include one of the following types of indexes:#
    - B-tree index defined for a single-chunk table
    - Text index defined for a single-chunk table
    - Index defined for a multi-chunk table

  #

  If an index for the table to be processed is deleted by using the `DROP INDEX` statement before the `adbidxrebuild` command is re-executed, the deleted index will not be rebuilt. In addition, if all indexes to be rebuilt are deleted before the adbidxrebuild command is re-executed, only command termination processing is performed.

- The processing stopped for one of the following reasons:
  - The adbidxrebuild command terminated abnormally.
  - The adbidxrebuild command was terminated forcibly by the adbcancel command.
  - The HADB server was terminated forcibly by the adbstop --force command.
  - The server process was terminated forcibly by the OS's kill command or because of a power outage.

In the following cases, even if the above conditions are satisfied, the re-execute facility is not applied (index rebuild processing is forcibly re-executed from the beginning):

- The adbidxrebuild command is executed with the --force option specified.
- After the adbimport command is stopped, the adbidxrebuild command that was executed with the --force option specified is canceled.
- An error occurred during execution of the adbunarchivechunk command.

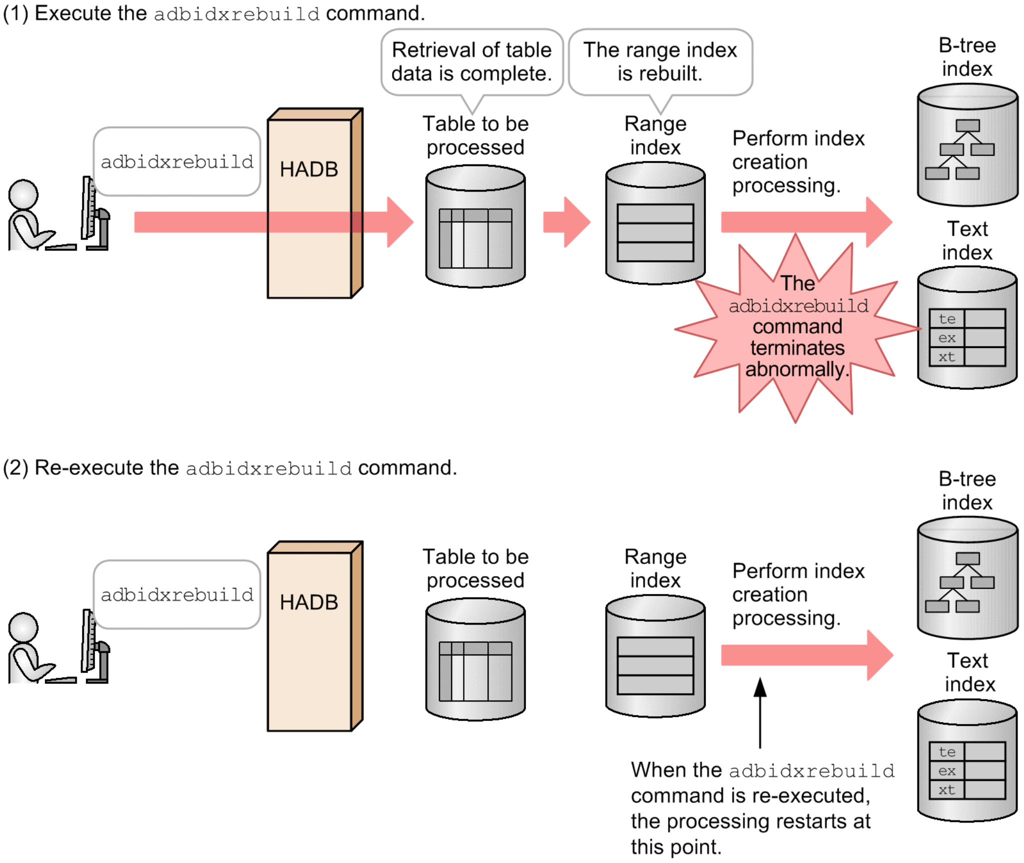
The conditions under which the re-execute facility is applied and the processing that is performed during command re-execution depend on whether CHUNK was specified in the definition of the target table, which contains the indexes to be rebuilt. That is, the conditions and processing depend on whether the target table is a single-chunk table or multi-chunk table.

If the target table is a single-chunk table

In index rebuild processing, table data is retrieved and range indexes are rebuilt first, and then B-tree and text indexes are rebuilt.

If the adbidxrebuild command is re-executed after retrieval of the table's data has been completed (and the KFAA80233-I message has been issued, indicating that retrieval of the table's data has been completed), the re-execute facility is applied and index rebuild processing restarts with rebuilding of B-tree and text indexes

Figure 16-4: Overview of the re-execute facility for a single-chunk table


(1) Execute the adbidxrebuild command.
Retrieval of table data is complete.
The range index is rebuilt.
B-tree index
adbidxrebuild
HADB
Table to be processed
Range index
Perform index creation processing.
Text index
The adbidxrebuild command terminates abnormally.
te
ex
xt
(2) Re-execute the adbidxrebuild command.
B-tree index
adbidxrebuild
HADB
Table to be processed
Range index
Perform index creation processing.
Text index
te
ex
xt
When the adbidxrebuild command is re-executed, the processing restarts at this point.

Legend:

Index creation processing: Creation of B-tree and text indexes

Note that the re-execute facility is not applied in the following cases:

- Only range indexes have been rebuilt.
- The `adbidxrebuild` command was canceled while table data was being retrieved (before the `KFAA80233-I` message was issued).
- The `adbidxrebuild` command was executed with the `--create-temp-file` option specified.

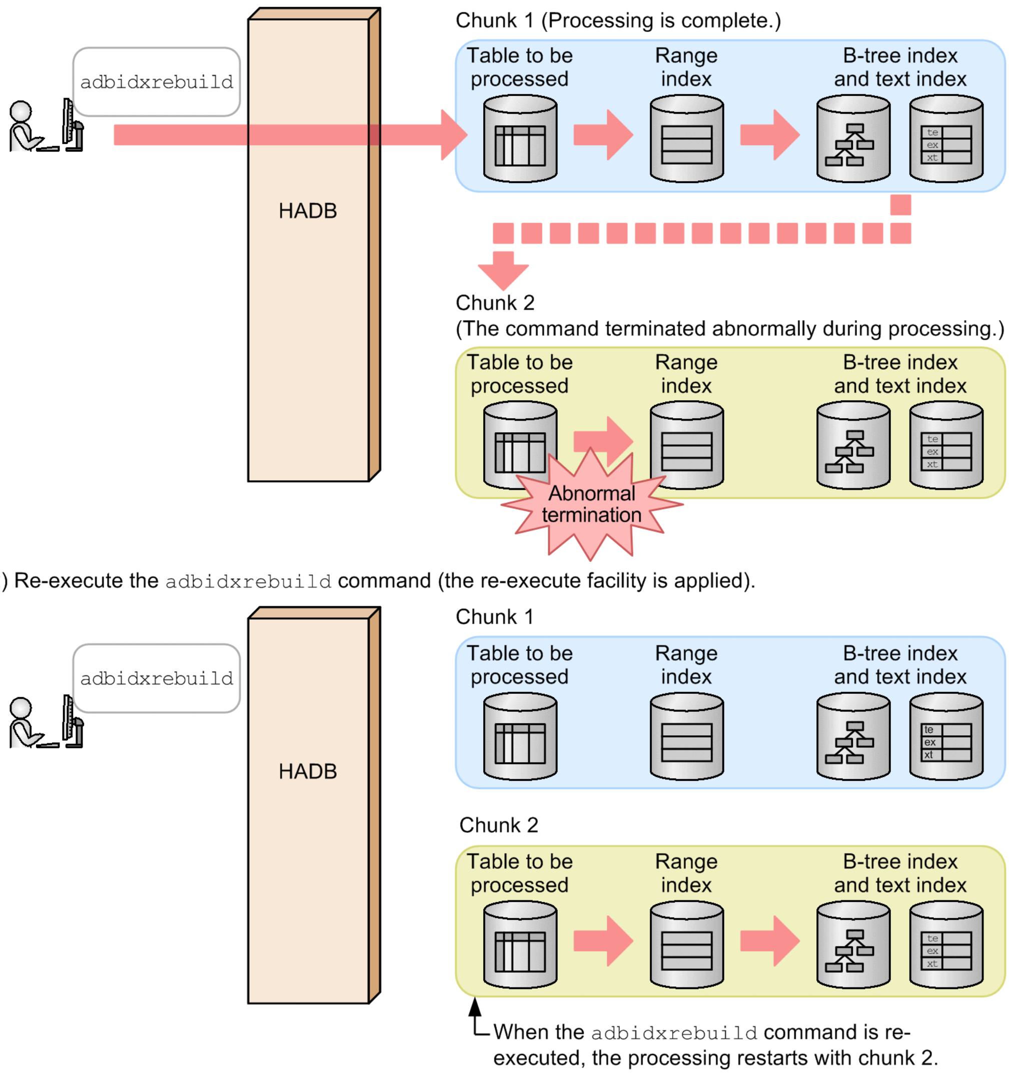
If the target table is a multi-chunk table

If the target table is a multi-chunk table, the index rebuild processing is performed for each chunk.

Therefore, if the `adbidxrebuild` command is re-executed after processing on a multi-chunk table stops, the re-execute facility is applied. Thus, the index rebuild processing restarts with the chunk for which processing was underway when the previous command's execution was canceled.

Note that if the `adbidxrebuild` command was stopped while it was retrieving table data from the first chunk (before the first instance of the `KFAA80233-I` was issued), the re-execute facility is not applied.

Figure 16-5: Overview of the re-execute facility for a multi-chunk table



(1) Execute the `adbidxrebuild` command on a table containing two chunks.

adbidxrebuild
HADB
Chunk 1 (Processing is complete.)
Table to be processed
Range index
B-tree index and text index
Chunk 2
(The command terminated abnormally during processing.)
Table to be processed
Range index
B-tree index and text index
Abnormal termination

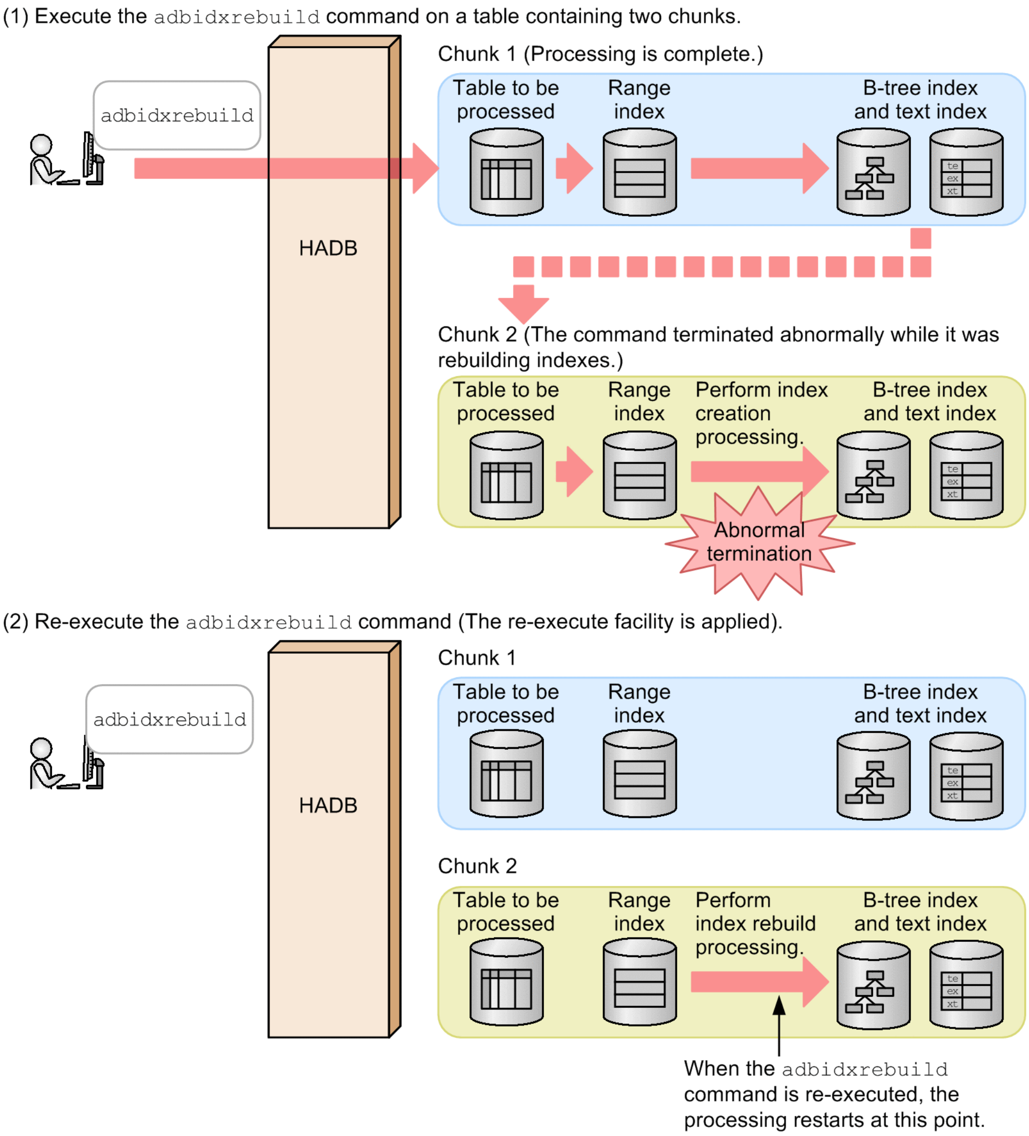
(2) Re-execute the `adbidxrebuild` command (the re-execute facility is applied).

adbidxrebuild
HADB
Chunk 1
Table to be processed
Range index
B-tree index and text index
Chunk 2
Table to be processed
Range index
B-tree index and text index
When the adbidxrebuild command is re-executed, the processing restarts with chunk 2.

If the following three conditions are satisfied, index rebuild processing restarts from the point of rebuilding B-tree and text indexes in the chunk where the previous processing was canceled:

- The indexes to be rebuilt include B-tree or text indexes.
- All table data has been retrieved from the chunk where the previous processing was canceled (when retrieval of table data from a chunk is completed, the `KFAA80233-I` message is issued).
- The `adbidxrebuild` command was executed without the `--create-temp-file` option specified (if the `adbidxrebuild` command was executed with the `--create-temp-file` option specified, index rebuild processing restarts from the point of retrieving table data in the chunk where the previous processing was canceled).

Figure 16-6: Restarting from rebuilding of B-tree and text indexes in the chunk where the previous processing was canceled


(1) Execute the adbidxrebuild command on a table containing two chunks.
adbidxrebuild
HADB
Chunk 1 (Processing is complete.)
Table to be processed
Range index
B-tree index and text index
Chunk 2 (The command terminated abnormally while it was rebuilding indexes.)
Table to be processed
Range index
Perform index creation processing.
B-tree index and text index
Abnormal termination
(2) Re-execute the adbidxrebuild command (The re-execute facility is applied).
adbidxrebuild
HADB
Chunk 1
Table to be processed
Range index
B-tree index and text index
Chunk 2
Table to be processed
Range index
Perform index rebuild processing.
B-tree index and text index
When the adbidxrebuild command is re-executed, the processing restarts at this point.

Legend:

Index creation processing: Creation of B-tree and text indexes

(2) Inheriting index creation processing from the adbimport command

The adbidxrebuild command can inherit index creation processing from the adbimport command.

If you execute the adbidxrebuild command because the adbimport command could not be executed for any of the following reasons, the adbidxrebuild command inherits index creation processing:#

- No temporary work file exists (the KFAA50247-E message is issued during re-execution of the adbimport command).

  If the temporary work file is deleted while the adbimport command is stopped, the adbimport command cannot be re-executed. In such a case, executing the adbidxrebuild command with the --create-temp-file option specified inherits index creation processing from the adbimport command.
- No command status file (a file under the $DBDIR/ADBSYS/ADBUTL directory) exists (the KFAA50244-E message is issued during re-execution of the adbimport command).

If an attempt is made to delete the command status file while the adbimport command is stopped, the adbimport command cannot be re-executed. In such a case, executing the adbidxrebuild command with the --force option specified inherits index creation processing from the adbimport command.

- An index is defined while the adbimport command is stopped, and as a result, an index in unfinished status is created in the table to be imported (the KFAA50209-E message is issued during re-execution of the adbimport command).

  If an index is defined while the adbimport command is stopped, you might not be able to re-execute the adbimport command. In such a case, executing the adbidxrebuild command with the --force option specified inherits index creation processing from the adbimport command.

#

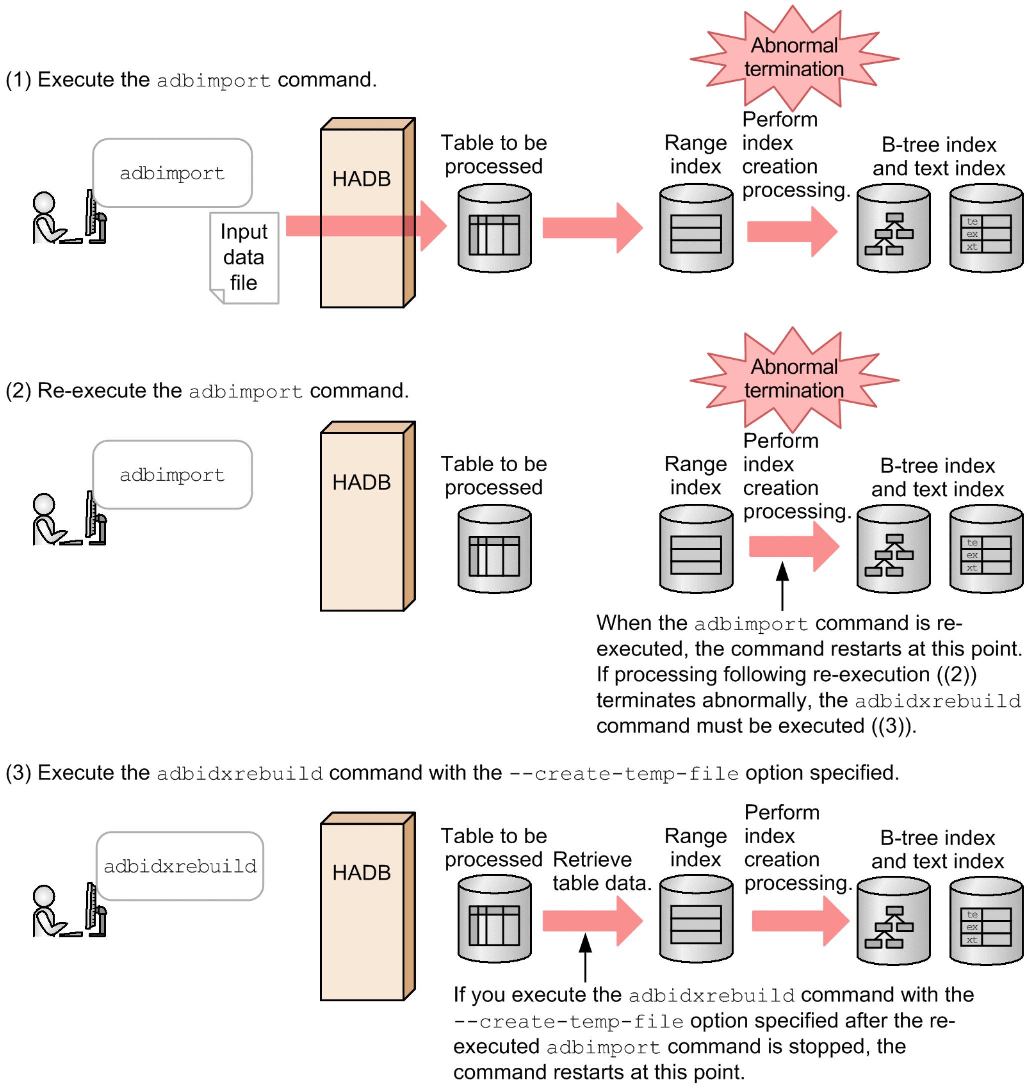
If an index for the table to be processed is deleted by using the DROP INDEX statement before the adbidxrebuild command is executed, the deleted index will not be rebuilt. In addition, if all indexes being created are deleted before the adbidxrebuild command is executed, only command termination processing is performed.

Processing by the adbidxrebuild command differs depending on the option specified for it.

■ If the **adbidxrebuild** command is executed with the **--create-temp-file** option specified

If you execute the adbidxrebuild command with the --create-temp-file option specified, the adbidxrebuild command starts with retrieval of table data for which a temporary work file is to be created.

Figure 16-7: Inheriting index creation processing from the adbimport command (with the --create-temp-file option specified)


(1) Execute the adbimport command.
Abnormal termination
adbimport
Input data file
HADB
Table to be processed
Range index
Perform index creation processing.
B-tree index and text index
(2) Re-execute the adbimport command.
Abnormal termination
adbimport
HADB
Table to be processed
Range index
Perform index creation processing.
B-tree index and text index
When the adbimport command is re-executed, the command restarts at this point. If processing following re-execution ((2)) terminates abnormally, the adbidxrebuild command must be executed ((3)).
(3) Execute the adbidxrebuild command with the --create-temp-file option specified.
adbidxrebuild
HADB
Table to be processed
Retrieve table data.
Range index
Perform index creation processing.
B-tree index and text index
If you execute the adbidxrebuild command with the --create-temp-file option specified after the re-executed adbimport command is stopped, the command restarts at this point.

Legend:

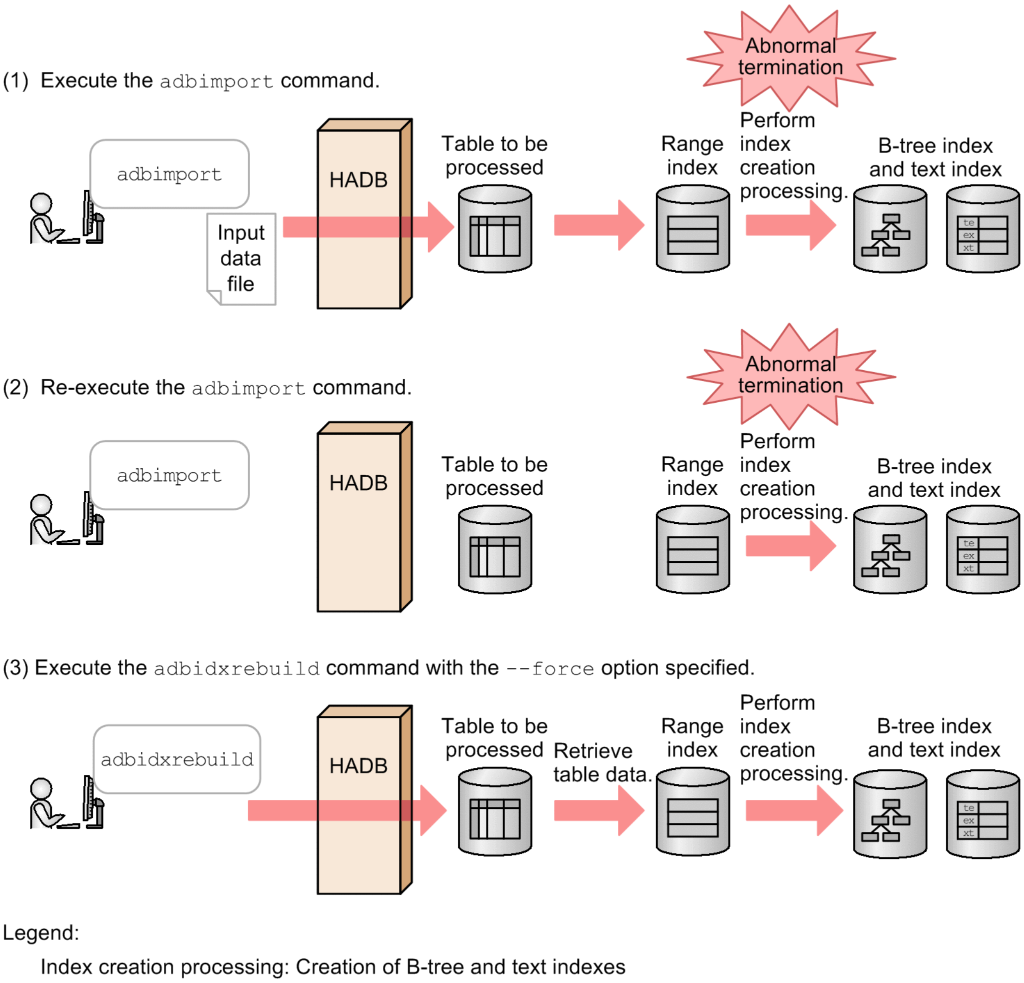
Index creation processing: Creation of B-tree and text indexes

If the table to be processed is a multi-chunk table, the chunk that was being processed when the adbimport command was stopped becomes the processing target of the adbidxrebuild command. The target chunk is not changed until the adbidxrebuild command terminates normally.

■ If the **adbidxrebuild** command is executed with the **--force** option specified

If you execute the adbidxrebuild command with the --force option specified, the adbidxrebuild command starts with the beginning of index rebuild processing.

Figure 16-8: Inheriting index creation processing from the adbimport command (with the --force option specified)


(1) Execute the adbimport command.
Abnormal termination
adbimport
Input data file
HADB
Table to be processed
Range index
Perform index creation processing.
B-tree index and text index
(2) Re-execute the adbimport command.
Abnormal termination
adbimport
HADB
Table to be processed
Range index
Perform index creation processing.
B-tree index and text index
(3) Execute the adbidxrebuild command with the --force option specified.
adbidxrebuild
HADB
Table to be processed
Retrieve table data.
Range index
Perform index creation processing.
B-tree index and text index
Legend:
Index creation processing: Creation of B-tree and text indexes

16.1.4 Relationships among threads used by the adbidxrebuild command

The `adbidxrebuild` command uses processing real threads in the following processing:

1. Retrieving table data and rebuilding range indexes

   One or more processing real threads are used.

2. Creating temporary work files

   One or more processing real threads are used.

3. Sorting records of B-tree and text indexes

   One or more processing real threads are used.

4. Rebuilding B-tree and text indexes

   One or more processing real threads are used.

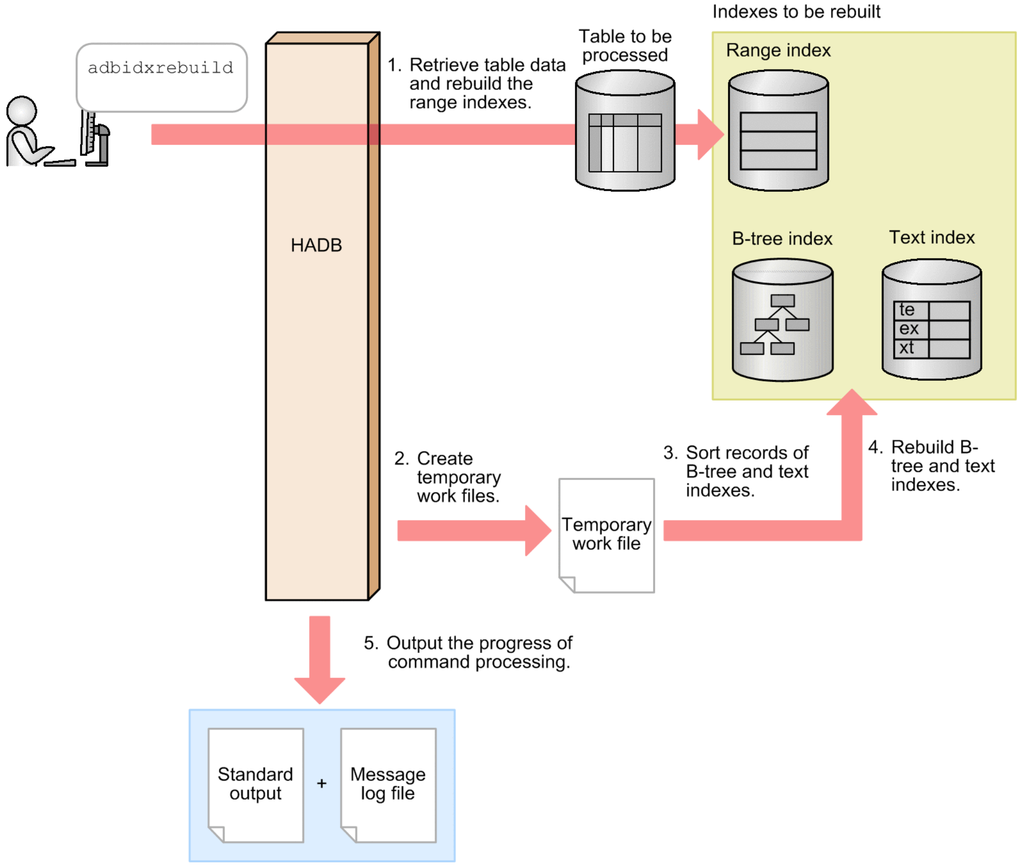
5. Outputting the progress of command processing

   One processing real thread is used.

Steps 1 and 2 are performed at the same time. Then, step 3 and then step 4 are performed. Step 5 is always performed while steps 1 to 4 are in progress.

The following figure shows the processing of the `adbidxrebuild` command using processing real threads. The preceding numbers correspond to the numbers in the figure.

Figure 16-9: Processing of the adbidxrebuild command using processing real threads


Indexes to be rebuilt
adbidxrebuild
1. Retrieve table data and rebuild the range indexes.
Table to be processed
Range index
HADB
B-tree index
Text index
te
ex
xt
2. Create temporary work files.
Temporary work file
3. Sort records of B-tree and text indexes.
4. Rebuild B-tree and text indexes.
5. Output the progress of command processing.
Standard output
+
Message log file

The number of processing real threads used by the `adbidxrebuild` command for each step is automatically calculated from the *number of processing real threads to be used by the* `adbidxrebuild` *command*. For details about the *number of processing real threads to be used by the* `adbidxrebuild` *command*, see the description of the index rebuild option adb_idxrebuild_rthd_num in (2) Explanation of specification format in 16.2.2 Format of index rebuild options.

16.2 Specification format for the adbidxrebuild command

This section explains the specification format for the `adbidxrebuild` command and how to specify the index rebuild options.

The following OS users can execute the `adbidxrebuild` command:

- HADB administrator
- OS users who belong to the HADB administrators group

16.2.1 Explanation of the specification format and options

(1) Specification format

```
adbidxrebuild
  -u authorization-identifier
  [-p password]
  [-g output-interval-for-index-rebuild-processing-progress-message]
  [-w {temporary-work-file-storage-directory-name
       |name-of-directory-path-file-specifying-temporary-work-file-storage-locations}
]
  [-z index-rebuild-option-file-name]
  [-i index-identifier-file-name]
  [{--create-temp-file|--force}]
  table-to-be-processed
```

> **Important**
>
> You must specify the table to be processed as the last option that is specified.

(2) Explanation of options

- `-u` *authorization-identifier*

  ~<character string>((1 to 100 bytes))

  Specifies the authorization identifier of the HADB user who executes the `adbidxrebuild` command. You must have the following two privileges to execute the `adbidxrebuild` command:

  - `CONNECT` privilege
  - `REBUILD INDEX` privilege for the table to be processed

  For this option, specify an authorization identifier in the range from 1 to 100 bytes. Note that the byte count (1 to 100 bytes) does not include the double quotation marks used to enclose the authorization identifier.

  > **Important**
  >
  > If the character string used as the authorization identifier includes any lowercase letter or a backslash (`\`), make sure that you check the rules for specifying authorization identifiers. For the rules for specifying authorization identifiers, see 1.4.2 Rules for specifying authorization identifiers and passwords.

Important

The authorization identifier specified in the `-u` option must not be changed while you are using the re-execute facility provided by the `adbidxrebuild` command. Do not delete a user who is executing the `adbidxrebuild` command or revoke the `REBUILD INDEX` privilege while indexes are being rebuilt.

If you have made a mistake, take one of the following steps:

- As the same user, make the `adbidxrebuild` command executable, and then re-execute this command. For details about the corrective action to take, see the topic *Managing HADB users* or *Managing access privileges* in *Unscheduled Operations* in the *HADB Setup and Operation Guide*.
- As a different user who has `REBUILD INDEX` privileges for the table to be processed, execute the `adbidxrebuild` command with the `--force` option specified. Note that the re-execute facility of the `adbidxrebuild` command is not applied in this case.

- `-p` *password*

  ~<character string>((1 to 255 bytes))

  Specifies the password for the authorization identifier that is specified in the `-u` option.

  Important

  If the password includes a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), make sure that you check the rules for specifying passwords. For the rules for specifying passwords, see 1.4.2 Rules for specifying authorization identifiers and passwords.

  If the `adbidxrebuild` command is executed with this option omitted, a message prompting the user to enter a password is displayed. In an environment in which a password cannot be entered from the standard input, such as when the command is executed in the background, make sure that you do not omit this option.

- `-g` *output-interval-for-index-rebuild-processing-progress-message*

  ~<integer>((0 to 1,000,000,000))<<0>>(in 1,000 rows)

  Specifies the interval (in units of 1,000 rows) at which the index rebuild processing progress message is to be output. For example, if `2` is specified in this option, the `KFAA80205-I` progress message is output each time 2,000 rows of table data have been retrieved.

  If this option is omitted or `0` is specified in this option, no progress message is output.

- `-w` `{`*temporary-work-file-storage-directory-name* `|` *name-of-directory-path-file-specifying-temporary-work-file-storage-locations*`}`

  ~<OS path name>((2 to 518 bytes))

  Specifies a storage location for the temporary work files that are created during index rebuild processing, expressed as either of the following:

  1. Absolute path name of the directory that stores temporary work files
  2. Absolute path name of a file (directory path file) that specifies directories for storing temporary work files

  If frequent input/output operations are performed on the temporary work files stored under a single directory, performance might be adversely affected. We recommend that you use method 2 and store temporary work files in directories on multiple disks to distribute the workload of input/output operations on the disk. The following shows an example specification using method 2 (specifying a directory path file).

  Example of directory path file

```
/mnt/diska/wwwww
/mnt/diskb/xxxxx
/mnt/diskc/yyyyy
/mnt/diskd/zzzzz
```

The following is a guideline for the number of directories to be specified:

Guideline-for-number-of-directories-to-be-specified = *A* × *B*

A: Maximum number of scan processing threads

Use the following formula to determine the value:

↓ (Value of the index rebuild option adb_idxrebuild_rthd_num - 1) ÷ 2↓

B: Number of B-tree and text indexes to be rebuilt

If you do not know the number of B-tree and text indexes, check the number of B-tree and text indexes that are defined for the table in the following SQL statement:

```
SELECT "N_INDEX"-"N_RANGE_INDEX"
    FROM "MASTER"."SQL_TABLES"
        WHERE "TABLE_SCHEMA"='authorization-identifier'
           AND "TABLE_NAME"='name-of-table-to-be-processed'
```

- A maximum of 255 directories can be specified in a directory path file.
- The directories specified in the directory path file are used as the storage locations of temporary work files in the order specified. When the last directory specified in the directory path file is used (in this example, `/mnt/diskd/zzzzz`), the first directory (in this example, `/mnt/diska/wwwww`) is then used again.

The following specification rules apply to this option (for both methods 1 and 2):

- Do not specify a forward slash (`/`) at the end of a path name.

For details about how to estimate the sizes of the temporary work files that will be created under the directories specified in the `-w` option, see *Estimating the size of the temporary work file for executing the adbidxrebuild command* in the *HADB Setup and Operation Guide*. If this option is omitted, temporary work files are created under `$DBDIR/ADBWORK`.

It is advisable to provide a dedicated directory to store temporary work files. If a directory storing other files is used for temporary work files, data in the existing files might be corrupted. Also make sure that symbolic links and relative paths are specified correctly. For details about the files whose names must differ from the temporary work file storage directory name, see 16.6 Notes.

If the disk that stores temporary work files runs out of free space, specify as the storage destination another disk with a larger capacity. For details about the actions to take if the disk that stores temporary work files runs out of free space, see *Steps to take in the event of a shortage of disk space for storing temporary work files during command execution* in *Command-related problems* in *Troubleshooting* in the *HADB Setup and Operation Guide*.

- `-z` *index-rebuild-option-file-name*

  ~<OS path name>((2 to 510 bytes))

  Specifies the absolute path name of the index rebuild option file that contains the index rebuild options. If this option is omitted, all index rebuild options are set to their default values because no index rebuild option file is specified.

  For details about the index rebuild option, see 16.2.2 Format of index rebuild options.

  Make sure that the index rebuild option file name is unique from all other file names. If the index rebuild option file has the same name as another file, data in the file might become corrupted. For details about the files whose names must differ from the index rebuild option file name, see 16.6 Notes.

- `-i` *index-identifier-file-name*

  ~<OS path name>((2 to 510 bytes))

Specifies the absolute path name of a file containing the identifiers of the indexes that are to be subject to index rebuild processing. All indexes specified in the index identifier file will be rebuilt regardless of their status.

If this option is omitted, only indexes in unfinished status that are defined for the table to be processed will be rebuilt.

After the cancelation of the adbimport command, if you use the adbidxrebuild command to inherit index creation processing, all indexes that are defined for the table to be processed are rebuilt.

The following shows a specification example of an index identifier file.

Example of the adbidxrebuild command

```
adbidxrebuild -u ADBUSER01 -i /home/adbmanager/rebuild_file/table01_idx.txt TABLE0
1
```

Specification example of an index identifier file (/home/adbmanager/rebuild_file/table01_idx.txt)

```
INDEX02
INDEX03
```

This example specifies as a command option an index identifier file containing the identifiers of two indexes that are to be subject to index rebuild processing. When the adbidxrebuild command is executed, INDEX02 and INDEX03 will be rebuilt.

The following are the rules for specifying an index identifier file:

- An index that is not defined for the table to be processed cannot be specified in the file.
- The same index identifier cannot be specified more than once.
- If an index identifier contains a lowercase letter, enclose the entire index identifier in double quotation marks ("). If an index identifier is not enclosed in double quotation marks ("), it is treated as being in all uppercase letters.
- To rebuild a B-tree index that corresponds to a primary key, specify the index identifier (same as the constraint name) that has been generated automatically. For details about how to check the index identifiers of the B-tree indexes that correspond to primary keys, see *Searching a dictionary table* in the *HADB Setup and Operation Guide*.
- Null rows in an index identifier file are ignored. If the specified index identifier file is empty, the adbidxrebuild command will result in an error.

- --create-temp-file

Normally, omit this option.

Specify this option if the message KFAA50247-E is issued when the adbimport or adbidxrebuild command is re-executed. The adbidxrebuild command is executed from the point of data retrieval for the table for which temporary work files will be created. After the adbimport command is canceled, if the adbidxrebuild command is executed with this option specified, it can inherit index creation processing from the adbimport command.

For an overview of the process of inheriting index creation processing from the adbimport command, see (2) Inheriting index creation processing from the adbimport command in 16.1.3 Optional functions of the adbidxrebuild command. In this case, careful consideration is necessary when specifying command options and index rebuild options. For details, see (1) Executing the adbidxrebuild command with the --create-temp-file option specified in 17.10.7 If an error occurs during re-execution of the adbimport command.

Important

- If the table to be processed is not non-updatable, you cannot specify this option (the KFAA50248-E message is output to the message log file).

- It is assumed that this option will be specified if the temporary work file necessary for re-execution does not exist (index record file or sort result file). If the adbidxrebuild command is executed when a temporary work file exists, the existing file is re-built. However, if you do not have write or read privileges for the applicable file, an access error might occur during execution of the adbidxrebuild command. (The KFAA30959-E, KFAA40204-E, KFAA40205-E, KFAA40214-E, KFAA41205-E, or KFAA41206-I message is issued. 13 (EACCES) is output for the error number.) In such a case, first delete the applicable file using the OS's rm command or a similar method, and then re-execute the adbidxrebuild command.

If you execute the adbidxrebuild command without specifying this option, index rebuild processing is executed from the point at which the content integrity was guaranteed during the previous index rebuild processing.

By contrast, if you execute the adbidxrebuild command with this option specified, index rebuild processing is executed from the point of data retrieval for the table for which temporary work files will be created. If a range index has been built, it is re-built during table data retrieval.

- --force

Normally, omit this option.

In the following cases, execute the adbidxrebuild command with this option specified. Index rebuild processing is forcibly executed from the beginning.

- If the KFAA50244-E message is issued during re-execution of the adbimport or adbidxrebuild command:

  - The command status files necessary for re-execution (files under the $DBDIR/ADBSYS/ADBUTL directory) do not exist.

  - The HADB server has been upgraded from version **04-00** or earlier without releasing tables from non-updatable status.

- An index was defined for the table to be processed while the adbimport command was stopped, and the KFAA50209-E message was issued during re-execution of the adbimport command (an index in unfinished status exists).

After the adbimport command is canceled, if you execute the adbidxrebuild command with this option specified, you can inherit index creation processing from the adbimport command.

Before executing the adbidxrebuild command with this option specified, delete the files (temporary work files) under the directories listed below using the OS's rm command or a similar method. If there are any files for which you do not have write or read privileges, you might not be able to execute the adbidxrebuild command.

- If the -w option was specified during execution of the previous command that was canceled:

  The temporary work file storage directory specified in the -w option

- If the -w option was omitted during execution of the previous command that was canceled:

  The $DBDIR/ADBWORK directory

For an overview of the process of inheriting index creation processing from the adbimport command, see (2) Inheriting index creation processing from the adbimport command in 16.1.3 Optional functions of the adbidxrebuild command. In this case, careful consideration is necessary when specifying command options and index rebuild options. For details, see (2) Executing the adbidxrebuild command with the --force option specified in 17.10.7 If an error occurs during re-execution of the adbimport command.

If you execute the adbidxrebuild command without specifying this option, index rebuild processing is executed from the point at which the content integrity was guaranteed during the previous index rebuild processing.

When the adbidxrebuild command is executed with this option specified, the command restarts index rebuild processing from the beginning.

- *table-to-be-processed*

  Specifies the table for which the index to be rebuilt is defined.

  The following rules apply to specifying this option:

  - Specify in the format *schema-name*.*table-identifier*.

    If the user owns the table, the schema name can be omitted. If another HADB user owns this table, the schema name must be specified.

  - For details about the specification rules that apply when a schema name or table identifier includes any lowercase letter or backslash (\), see 1.4.3 Table name specification rules.

  - The user can specify only a table for which the user has the `REBUILD INDEX` privilege.

  - A viewed table cannot be specified.

16.2.2 Format of index rebuild options

(1) Specification format

```
[set adb_idxrebuild_rthd_num = number-of-processing-real-threads-to-be-used-by-adbidx
rebuild-command]
[set adb_idxrebuild_scan_buff_size = table-data-scan-buffer-size]
[set adb_idxrebuild_buff_blk_num = number-of-buffer-blocks-for-rebuilding-indexes]
[set adb_idxrebuild_sort_buff_size = sort-buffer-size]
[set adb_idxrebuild_dvix_rdbuff_size = B-tree-and-text-index-input-buffer-size]
[set adb_idxrebuild_dvix_wtbuff_size = B-tree-and-text-index-output-buffer-size]
[set adb_idxrebuild_txt_buff_size = text-index-creation-buffer-size]
```

The specification rules for these index rebuild options are the same as for the server definition. For details about the syntax rules for the server definition, see the topic *Syntax rules for the server definition* in the *HADB Setup and Operation Guide*.

(2) Explanation of specification format

- `adb_idxrebuild_rthd_num` = *number-of-processing-real-threads-to-be-used-by-adbidxrebuild-command*

  ~<integer>((3 to 4,096))

  Specifies the number of processing real threads to be used for executing the `adbidxrebuild` command.

  Specification of this option is recommended. The `adbidxrebuild` command uses the number of processing real threads specified in this option. Therefore, specify an appropriate number of processing real threads for use by the `adbidxrebuild` command. For details about the processing of the `adbidxrebuild` command that uses processing real threads and the number of processing real threads to be used for each step, see 16.1.4 Relationships among threads used by the adbidxrebuild command.

  If this option is omitted, the number of processing real threads to be used for executing the `adbidxrebuild` command is determined by the `adb_sql_exe_max_rthd_num` operand in the server definition. The following table shows the relationship between the value of this option or the `adb_sql_exe_max_rthd_num` operand in the server definition and the number of processing real threads used for executing the command.

Table 16-1: Number of processing real threads used for executing the command

No.	Value in this option or in the server definition		Number of processing real threads used for executing the command
1	When this option is omitted	If the `adb_sql_exe_max_rthd_num` operand in the server definition is not specified	The default value of the `adb_sql_exe_max_rthd_num` operand in the server definition is used.
2		If a value in the range from `0` to `2` is specified in the `adb_sql_exe_max_rthd_num` operand in the server definition	--
3		If `3` or a greater value is specified for the `adb_sql_exe_max_rthd_num` operand in the server definition	The value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition is used.
4	If `3` or a greater value is specified in this option		The value specified in this option is used.

Legend:

--: The `adbidxrebuild` command results in an error.

Note

For details about the `adb_sql_exe_max_rthd_num` operand in the server definition, see *Operands related to performance (set format)* in the *HADB Setup and Operation Guide*.

- `adb_idxrebuild_scan_buff_size` = *table-data-scan-buffer-size*

~<integer>((1 to 2,048))<<8>> (megabytes)

Specifies the size (in megabytes) of the buffer to be used for retaining table data retrieval results. A larger value can reduce overhead in retrieving table data.

As a guideline, specify the value obtained from the following formula:

Formula

```
Key length of the B-tree indexes to be rebuilt × 100,000
```

For details about how to determine the key length of a B-tree index, see *Determining the key length (KEYSZ) of a B-tree index* in *Estimating the size of the data DB area* in *Designing a Database* in the *HADB Setup and Operation Guide*.

- `adb_idxrebuild_buff_blk_num` = *number-of-buffer-blocks-for-rebuilding-indexes*

~<integer>((5 to 4,096))<<64>>

Specifies the number of buffer blocks to be used for index rebuild processing. You can use the default value, but a larger value might improve the speed of index rebuild processing.

- `adb_idxrebuild_sort_buff_size` = *sort-buffer-size*

~<integer>((1 to 2,048))<<256>> (megabytes)

Consider specifying this option when you rebuild B-tree or text indexes.

Specifies the size (in megabytes) of the sort buffer to be used for building B-tree and text indexes.

Specify a large value if there is sufficient memory. Better performance can be expected as this option's value becomes larger.

- `adb_idxrebuild_dvix_rdbuff_size` = *B-tree-and-text-index-input-buffer-size*

~<integer>((32 to 51,200))<<1,024>> (kilobytes)

Consider specifying this option when you rebuild B-tree or text indexes.

Specifies the size (in kilobytes) of the input buffer to be used for building B-tree and text indexes.

Specify a large value if there is sufficient memory. Better performance can be expected as this option's value becomes larger.

- `adb_idxrebuild_dvix_wtbuff_size` = *B-tree-and-text-index-output-buffer-size*

  ~<integer>((32 to 51,200))<<1,024>> (kilobytes)

  Consider specifying this option when you rebuild B-tree or text indexes.

  Specifies the size (in kilobytes) of the output buffer to be used for building B-tree and text indexes.

  Specify a large value if there is sufficient memory. Better performance can be expected as this option's value becomes larger.

- `adb_idxrebuild_txt_buff_size` = *text-index-creation-buffer-size*

  ~<integer>((1 to 2,048))<<256>> (megabytes)

  Consider specifying this option when you rebuild text indexes.

  Specifies the size (in megabytes) of the buffer to be used for building text indexes.

  Increasing the value specified for this option often improves performance. However, the amount of buffer memory specified by this option will be used for all processing real threads on which text indexes will be built. Therefore, be careful when you increase the value specified for this option.

16.3 Examples

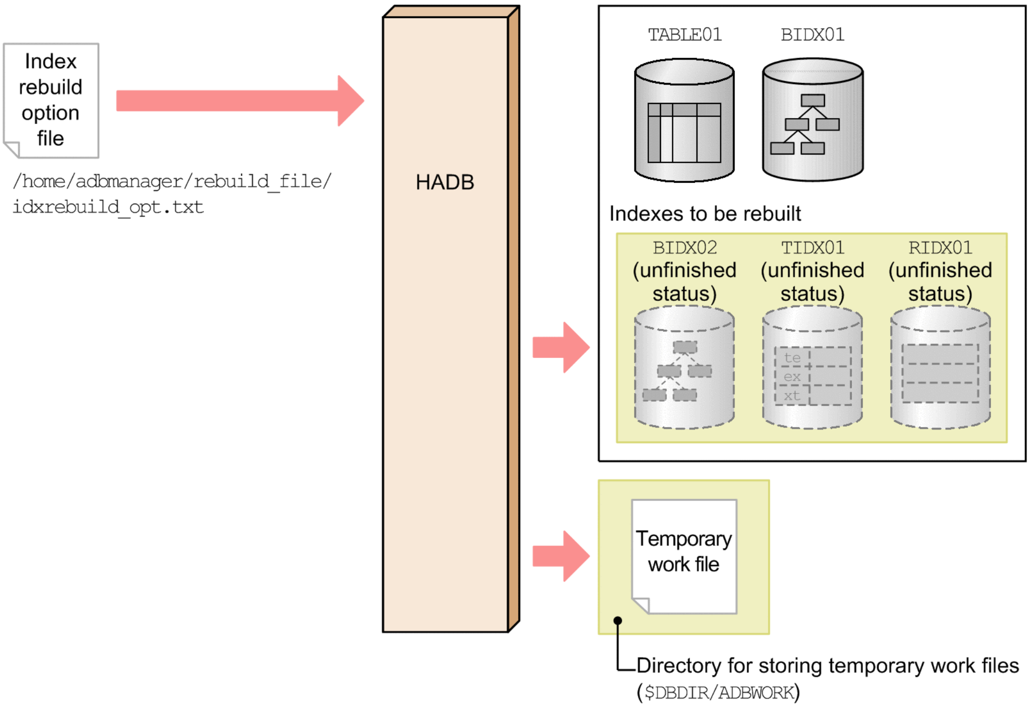
This section explains by way of examples how to use the adbidxrebuild command to rebuild indexes.

If the adbidxrebuild command terminates abnormally during execution, take appropriate action based on the explanation in 16.5 Handling abnormal termination of the adbidxrebuild command.

16.3.1 Rebuilding indexes in unfinished status

The example below shows the rebuilding of indexes BIDX02, TIDX01, and RIDX01, which are in unfinished status, so that they can be used. The following conditions apply:

1. B-tree index BIDX01 is defined for table TABLE01.
2. Data is stored in table TABLE01.
3. After data was stored in table TABLE01, definitions for B-tree index BIDX02, text index TIDX01, and range index RIDX01 were added to table TABLE01 (BIDX02, TIDX01, and RIDX01 are in unfinished status).


Index rebuild option file
/home/adbmanager/rebuild_file/idxrebuild_opt.txt
HADB
TABLE01
BIDX01
Indexes to be rebuilt
BIDX02 (unfinished status)
TIDX01 (unfinished status)
RIDX01 (unfinished status)
Temporary work file
Directory for storing temporary work files ($DBDIR/ADBWORK)

This example uses the following procedure to execute the adbidxrebuild command.

Procedure

1. Create an index rebuild option file.
2. Execute the adbidxrebuild command.

Using an example, the following subsections explain each of these steps.

(1) Creating an index rebuild option file

This example creates an index rebuild option file and specifies index rebuild options in that file.

Specification example of index rebuild option file

```
set adb_idxrebuild_rthd_num = 3
set adb_idxrebuild_scan_buff_size = 2
      :
```

(2) Executing the adbidxrebuild command

```
adbidxrebuild -u ADBUSER01                                          ...1
              -p '#HelloHADB_01'                                    ...2
              -g 10                                                 ...3
              -z /home/adbmanager/rebuild_file/idxrebuild_opt.txt   ...4
              TABLE01                                               ...5
```

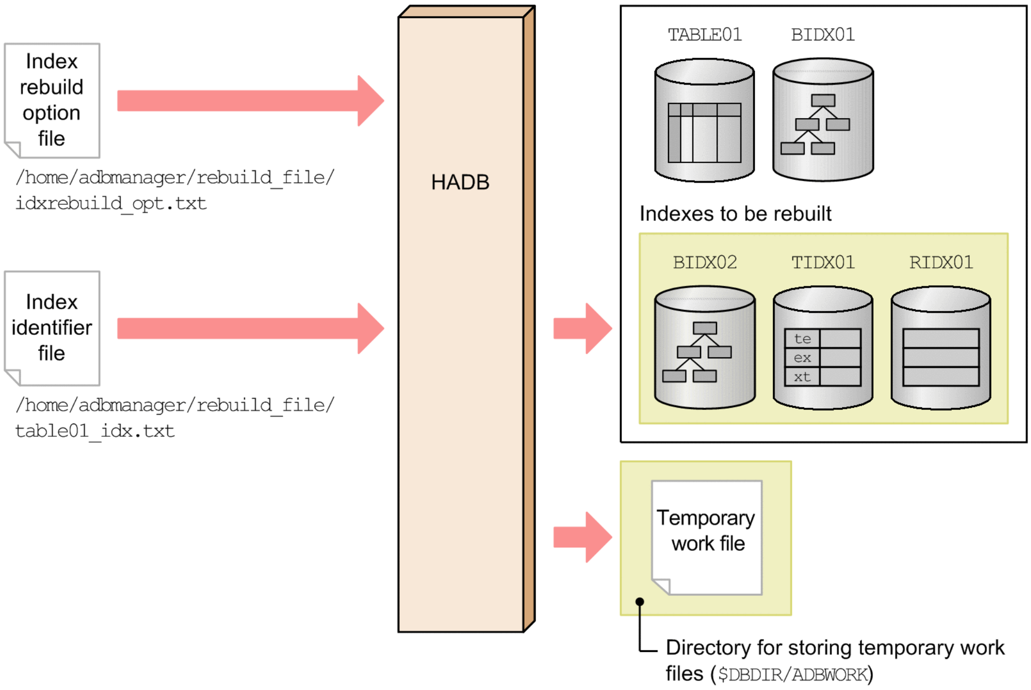
Explanation:

1. Specifies the authorization identifier of the HADB user who executes the adbidxrebuild command.
2. Specifies the password for the authorization identifier that is specified in the -u option.
3. Specifies that the index rebuild processing progress message (KFAA80205-I) is to be output after every 10,000 rows.
4. Specifies the absolute path name of the index rebuild option file created in (1) Creating an index rebuild option file.
5. Specifies the table to be processed (TABLE01).
   Because no index identifier file is specified, the indexes in unfinished status (BIDX02, TIDX01, and RIDX01) that have been defined for the target table are rebuilt.

16.3.2 Rebuilding specified indexes

The example below shows the rebuilding of indexes BIDX02, TIDX01, and RIDX01 under the following conditions:

- B-tree indexes BIDX01 and BIDX02, text index TIDX01, and range index RIDX01 have been defined for table TABLE01.
- The indexes are all in normal status.


Index rebuild option file
/home/adbmanager/rebuild_file/idxrebuild_opt.txt
Index identifier file
/home/adbmanager/rebuild_file/table01_idx.txt
HADB
TABLE01
BIDX01
Indexes to be rebuilt
BIDX02
TIDX01
RIDX01
te
ex
xt
Temporary work file
Directory for storing temporary work files ($DBDIR/ADBWORK)

This example uses the following procedure to execute the `adbidxrebuild` command.

Procedure

1. Create an index identifier file.
2. Create an index rebuild option file.
3. Execute the `adbidxrebuild` command.

Using examples, the following subsections explain each of these steps.

(1) Creating an index identifier file

This example creates an index identifier file and specifies in that file the identifiers of the indexes that are to be subject to index rebuild processing.

Specification example of index identifier file

```
BIDX02
TIDX01
RIDX01
```

(2) Creating an index rebuild option file

This example creates an index rebuild option file and specifies index rebuild options in that file.

Specification example of index rebuild option file

```
set adb_idxrebuild_rthd_num = 3
set adb_idxrebuild_scan_buff_size = 2
      :
```

(3) Executing the adbidxrebuild command

```
adbidxrebuild -u ADBUSER01                                           ...1
              -p '#HelloHADB_01'                                     ...2
              -g 10                                                  ...3
              -z /home/adbmanager/rebuild_file/idxrebuild_opt.txt    ...4
              -i /home/adbmanager/rebuild_file/table01_idx.txt       ...5
              TABLE01                                                ...6
```

Explanation:

1. Specifies the authorization identifier of the HADB user who executes the `adbidxrebuild` command.
2. Specifies the password for the authorization identifier that is specified in the `-u` option.
3. Specifies that the index rebuild processing progress message (`KFAA80205-I`) is to be output after every 10,000 rows.
4. Specifies the absolute path name of the index rebuild option file created in (2) Creating an index rebuild option file.
5. Specifies the absolute path name of the index identifier file created in (1) Creating an index identifier file.
6. Specifies the table to be processed (`TABLE01`).

16.4 Return codes

The following table lists and describes the return codes of the `adbidxrebuild` command.

Table 16-2: Return codes of the adbidxrebuild command

Return code	Description
`0`	The `adbidxrebuild` command terminated normally.
`4`	The `adbidxrebuild` command terminated normally, but a warning message was issued. Check the displayed message or the message output to a message log file and take appropriate action. If the `KFAA61205-W` message (warning message indicating detection of duplicate key values) was issued, the table and a B-tree index subject to index rebuild processing violate the uniqueness constraint (a B-tree index that violates the uniqueness constraint is in the status called uniqueness constraint violation). Locate the duplicate key values and delete the corresponding rows. Then re-execute the `adbidxrebuild` command to rebuild the B-tree indexes. For details about how to handle such an error, see *Steps to take when the uniqueness constraint is violated (when the KFAA61205-W message is output)* in *Problems related to B-tree indexes* in *Troubleshooting* in the *HADB Setup and Operation Guide*.
`8`	One of the following errors occurred: • The specified `adbidxrebuild` command was not executed because it was invalid. • An error occurred during execution of the `adbidxrebuild` command. For the cause of the error and corrective action to take, see the displayed error message or the message that has been output to a message log file. For details, see 16.5 Handling abnormal termination of the adbidxrebuild command.

16.5 Handling abnormal termination of the adbidxrebuild command

This subsection describes the actions to take if the `adbidxrebuild` command terminates abnormally.

16.5.1 First thing to check

Check if the following messages have been issued:

- `KFAA50225-E`
- `KFAA96606-E`

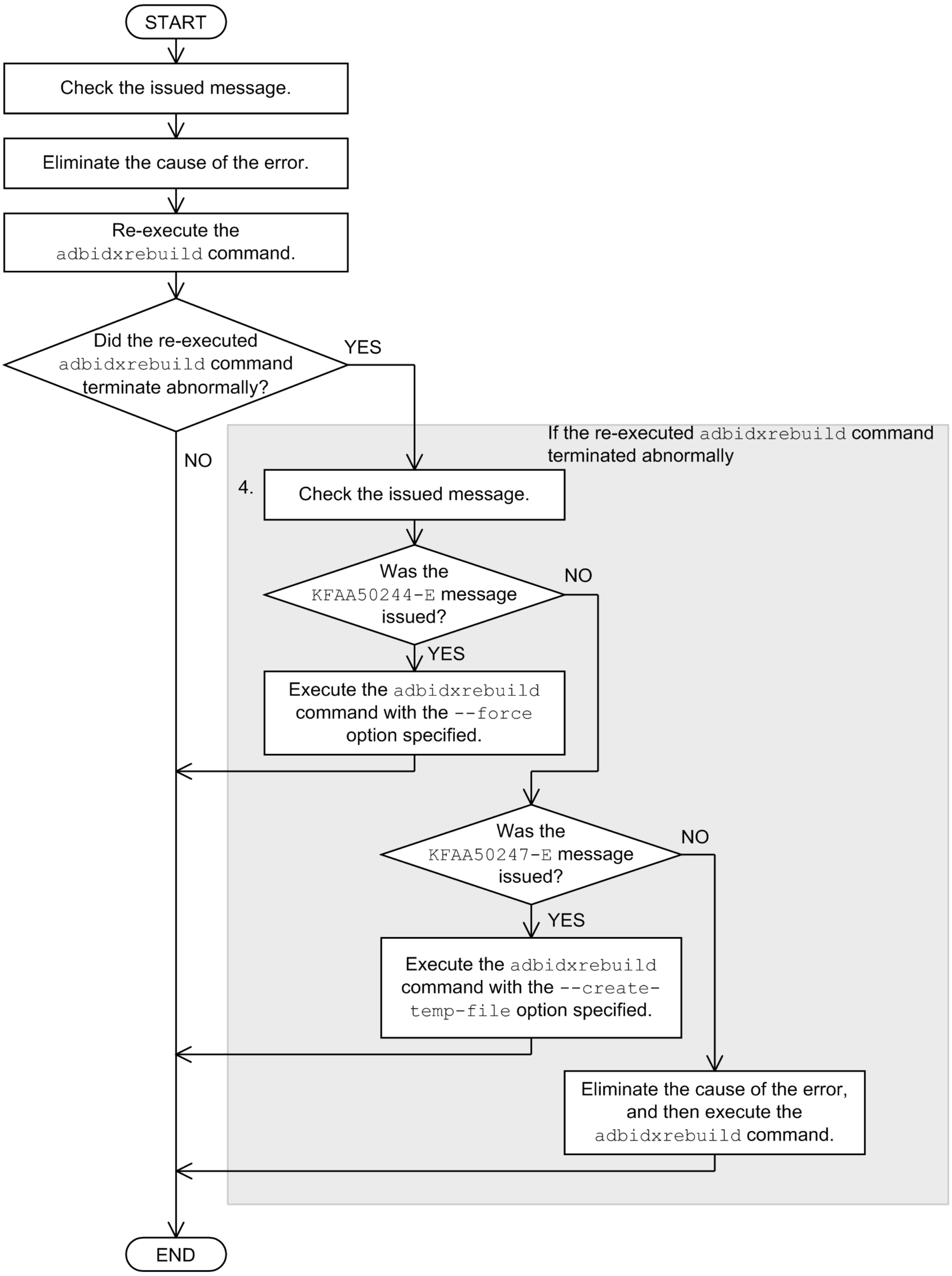
If either of these messages has been output, there are errors in specified command options or index rebuild options. Correct the errors in the command options or index rebuild options, and then re-execute the `adbidxrebuild` command.

If neither of these messages was issued, go to 16.5.2 Identifying the cause of the error and taking corrective action.

16.5.2 Identifying the cause of the error and taking corrective action

The following figure shows the procedural flow to follow if the `adbidxrebuild` command terminates abnormally.

Figure 16-10: Procedural flow to follow if the adbidxrebuild command terminates abnormally


START
1. Check the issued message.
2. Eliminate the cause of the error.
3. Re-execute the adbidxrebuild command.
Did the re-executed adbidxrebuild command terminate abnormally?
YES
NO
If the re-executed adbidxrebuild command terminated abnormally
4. Check the issued message.
Was the KFAA50244-E message issued?
NO
YES
Execute the adbidxrebuild command with the --force option specified.
Was the KFAA50247-E message issued?
NO
YES
Execute the adbidxrebuild command with the --create-temp-file option specified.
Eliminate the cause of the error, and then execute the adbidxrebuild command.
END

Procedure

1. Check the messages that have been issued and identify the cause of the error.
2. Take appropriate action to eliminate the cause of the error.
3. Re-execute the `adbidxrebuild` command.
4. If the re-executed `adbidxrebuild` command terminates abnormally, check the message that was issued and execute the `adbidxrebuild` command.

   For details, see 16.5.6 If an error occurs during re-execution of the adbidxrebuild command.

When the following indexes are being rebuilt, see 16.5.3 If the target table is a single-chunk table:

- B-tree index defined for a single-chunk table
  Includes the B-tree index corresponding to the primary key of the single-chunk table.
- Text index defined for a single-chunk table

If the indexes defined for the multi-chunk table are being rebuilt, see 16.5.4 If the target table is a multi-chunk table.

■ When the multi-node function is used

If the adbidxrebuild command is canceled and the table to be processed becomes non-updatable, release the base table from updatable status by re-executing the canceled adbidxrebuild command.

If the master node was switched due to abnormal termination of the HADB server on the master node, execute the adbidxrebuild command on the new master node to release the base table from non-updatable status. At this time, execute the adbidxrebuild command with the --create-temp-file option specified.

For details about how to switch the master node to another node, and how to start the HADB server without using the multi-node function, see the topic *Problems related to startup or termination of the HADB servers in the multi-node configuration* in *Troubleshooting (when the multi-node function is being used)* in *Operations When Using the Multi-Node Function* in the *HADB Setup and Operation Guide*.

For details about the --create-temp-file option of the adbidxrebuild command, see --create-temp-file in (2) Explanation of options in 16.2.1 Explanation of the specification format and options.

16.5.3 If the target table is a single-chunk table

If you are rebuilding a B-tree index defined for a single-chunk table (including a B-tree index corresponding to the primary key) or a text index, you might be able to use the re-execute facility of the adbidxrebuild command. This facility resumes index rebuild processing from the point of B-tree and text index creation processing.

(1) Notes before re-execution

The re-execute facility is applied automatically when the adbidxrebuild command is re-executed by the same HADB user who executed the adbidxrebuild command that terminated abnormally. No special operation is needed.

If you want to change command options or index rebuild options when you re-execute the adbidxrebuild command, see 16.5.5 Changing command options and index rebuild options during re-execution.

Important

- If an access error occurred on a command status file (file under the $DBDIR/ADBSYS/ADBUTL directory) and the KFAA50244-E message is issued, re-execute the adbidxrebuild command with the --force option specified. (In this case, index rebuild processing is executed from the beginning.)
  For details about how to execute the adbidxrebuild command with the --force option specified, see (2) Executing the adbidxrebuild command with the --force option specified in 16.5.6 If an error occurs during re-execution of the adbidxrebuild command.
- If an access error occurred on a temporary work file (the KFAA50247-E message is issued), re-execute the adbidxrebuild command with the --create-temp-file option specified. (In this case, index rebuild processing is executed from the point of table data retrieval.)

For details about how to execute the `adbidxrebuild` command with the `--create-temp-file` option specified, see (1) Executing the adbidxrebuild command with the --create-temp-file option specified in 16.5.6 If an error occurs during re-execution of the adbidxrebuild command.

(2) Whether the re-execute facility is applied (for a single-chunk table)

The table below describes whether the re-execute facility is applied depending on when the `adbidxrebuild` command was canceled.

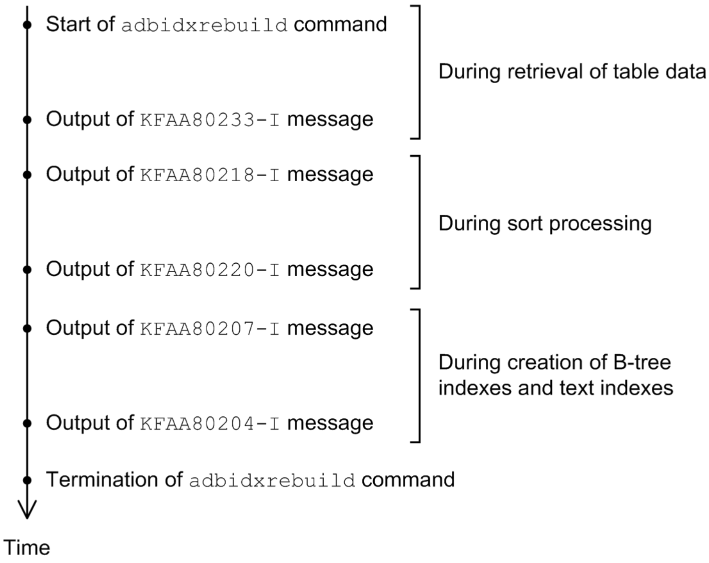
Note that the database recovery method is the same regardless of when cancellation occurred.

Table 16-3: Whether the re-execute facility is applied depending on when the adbidxrebuild command was canceled (for a single-chunk table)

When canceled#1		
During table data retrieval processing	**During sort processing**	**While creating B-tree and text indexes#2**
Whether the re-execute facility is applied Not applied.#3 **Table status** One of the following: • Non-updatable • Status before the `adbidxrebuild` command was executed **Index status** B-tree, text, and range indexes are in one of the following statuses: • Unfinished status • Status before the `adbidxrebuild` command was executed	**Whether the re-execute facility is applied** Applied.#4 **Table status** Non-updatable **Index status** B-tree and text indexes are in unfinished status. Range indexes have been rebuilt (range indexes have been released from unfinished status).	

#1

You can determine when the command was canceled by the messages that are issued, as shown in the following:


Start of adbidxrebuild command
Output of KFAA80233-I message
During retrieval of table data
Output of KFAA80218-I message
Output of KFAA80220-I message
During sort processing
Output of KFAA80207-I message
Output of KFAA80204-I message
During creation of B-tree indexes and text indexes
Termination of adbidxrebuild command
Time

#2

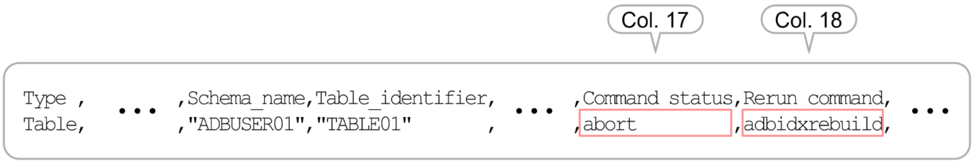
If the `KFAA80204-I` message has been issued with return code `0` or `4`, index rebuild processing has been completed. In such a case, there is no need to re-execute the `adbidxrebuild` command.

If the `KFAA80204-I` message has not been issued, execute the command shown below and check line 2 in the output results (table summary information) (line 1 in the output results contains the column titles).

```
adbdbstatus -c table -n name-of-table-for-which-adbidxrebuild-command-was-canceled
```

Specify *name-of-table-for-which-adbidxrebuild-command-was-canceled* in the format *schema-name*`.`*table-identifier*.

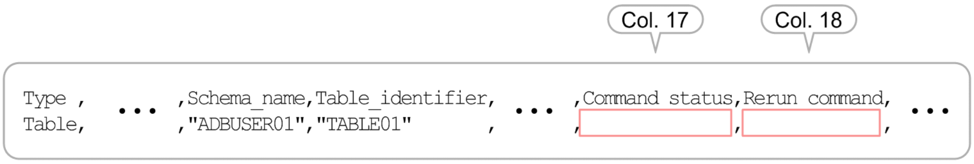
■ When the value of the `Command_status` column (column 17) is `abort` and the value of the `Rerun_command` column (column 18) is `adbidxrebuild`



```
Type ,   ...  ,Schema_name,Table_identifier,  ...  ,Command_status,Rerun_command,  ...
Table,        ,"ADBUSER01","TABLE01"       ,       ,abort         ,adbidxrebuild,
```

Index rebuild processing was not completed. Re-execute the `adbidxrebuild` command.

■ When no values are displayed in the `Command_status` column (column 17) and the `Rerun_command` column (column 18)



```
Type ,   ...  ,Schema_name,Table_identifier,  ...  ,Command_status,Rerun_command,  ...
Table,        ,"ADBUSER01","TABLE01"       ,       ,              ,             ,
```

Index rebuild processing has been completed. There is no need to re-execute the `adbidxrebuild` command.

#3

When the `adbidxrebuild` command is re-executed, index rebuild processing will be restarted from the beginning.

However, if all indexes to be rebuilt are deleted before the `adbidxrebuild` command is re-executed, only command termination processing is performed.

#4

When the `adbidxrebuild` command is re-executed, the command resumes index rebuild processing from the point of B-tree and text index creation processing.

In the following cases, the command resumes index rebuild processing from the point of table data retrieval:

- The `adbidxrebuild` command is executed with the `--create-temp-file` option specified
- After the `adbimport` command is stopped, the `adbidxrebuild` command executed with the `--create-temp-file` option specified is canceled

In addition, if all indexes to be rebuilt are deleted before the `adbidxrebuild` command is re-executed, only command termination processing is performed.

16.5.4 If the target table is a multi-chunk table

If you are rebuilding indexes defined for a multi-chunk table, you can use the re-execute facility of the adbidxrebuild command to restart index rebuild processing with the chunk that was being processed when the previous processing was canceled.

(1) Notes before re-execution

The re-execute facility is applied automatically when the adbidxrebuild command is re-executed by the same HADB user who executed the adbidxrebuild command that terminated abnormally. No special operation is needed.

Make sure that you do not change the -i option when you re-execute the adbidxrebuild command. If you want to change other command options or index rebuild options when you re-execute the adbidxrebuild command, see 16.5.5 Changing command options and index rebuild options during re-execution.

Important

- If an access error occurred on a command status file (file under the $DBDIR/ADBSYS/ADBUTL directory) and the KFAA50244-E message was issued, re-execute the adbidxrebuild command with the --force option specified. (In this case, index rebuild processing is executed from the beginning.)

  For details about how to execute the adbidxrebuild command with the --force option specified, see (2) Executing the adbidxrebuild command with the --force option specified in 16.5.6 If an error occurs during re-execution of the adbidxrebuild command.
- If an access error occurred on a temporary work file (the KFAA50247-E message is issued), re-execute the adbidxrebuild command with the --create-temp-file option specified. (In this case, index rebuild processing is executed from the point of table data retrieval.)

  For details about how to execute the adbidxrebuild command with the --create-temp-file option specified, see (1) Executing the adbidxrebuild command with the --create-temp-file option specified in 16.5.6 If an error occurs during re-execution of the adbidxrebuild command.

Note

The chunk ID of the chunk where index rebuild processing was started is displayed in the KFAA80211-I message, and the chunk ID of the chunk where the processing was completed is displayed in the KFAA80212-I message.

(2) Restart processing (for a multi-chunk table)

If you are rebuilding indexes defined for a multi-chunk table, the location where the processing is restarted when the adbidxrebuild command is re-executed depends on when the adbidxrebuild command was canceled.

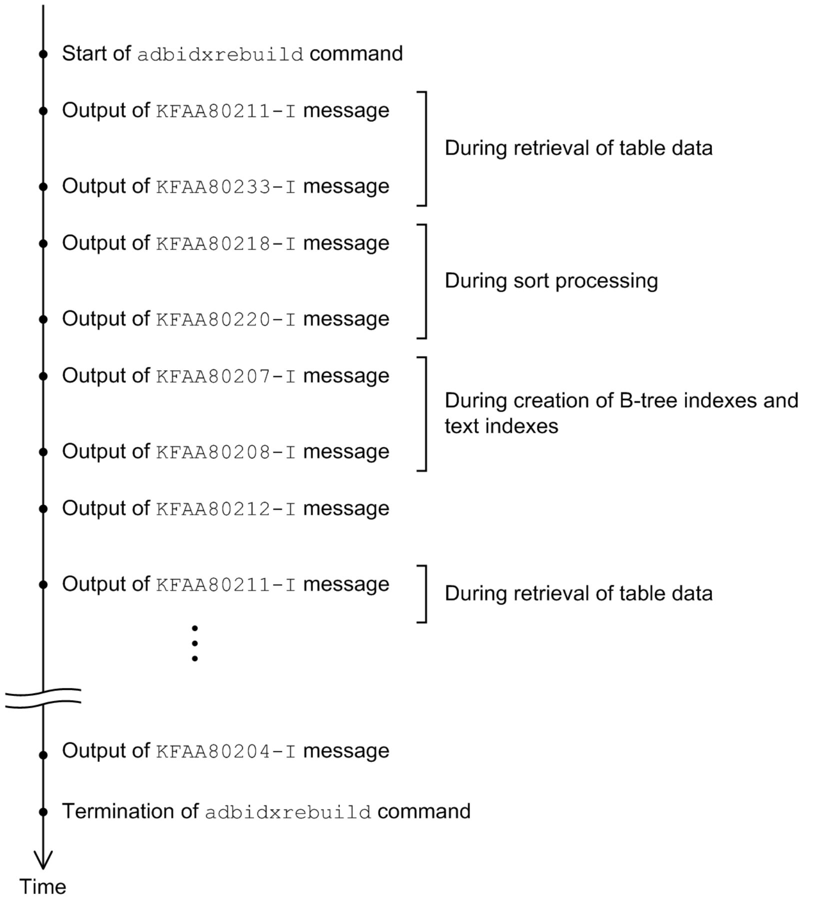
Note that the database recovery method is the same regardless of when cancellation occurred.

Table 16-4: Restart processing depending on when the adbidxrebuild command was canceled (for a multi-chunk table)

When canceled#1		
During table data retrieval processing	**During sort processing**	**While creating B-tree and text indexes#2**
Processing at the restart Index rebuild processing is restarted from the point of table data retrieval for the chunk where the previous processing was canceled.#3 **Table status** One of the following: • Non-updatable • Status before the `adbidxrebuild` command was executed **Index status** B-tree, text, and range indexes are in one of the following statuses: • Unfinished status • Status before the `adbidxrebuild` command was executed	**Processing at the restart** Index rebuild processing resumes from the point of building the B-tree and text indexes of the chunk in which processing stopped.#4 **Table status** Non-updatable **Index status** B-tree, text, and range indexes are in unfinished status. If the command was canceled while it was processing the final chunk,#5 the range index has been rebuilt (the range index is released from unfinished status).	

#1

You can determine when the command was canceled by the messages that are issued, as shown below. The `KFAA80211-I` through `KFAA80212-I` messages are output for each chunk ID of the table.


Start of adbidxrebuild command
Output of KFAA80211-I message
Output of KFAA80233-I message
During retrieval of table data
Output of KFAA80218-I message
Output of KFAA80220-I message
During sort processing
Output of KFAA80207-I message
Output of KFAA80208-I message
During creation of B-tree indexes and text indexes
Output of KFAA80212-I message
Output of KFAA80211-I message
During retrieval of table data
Output of KFAA80204-I message
Termination of adbidxrebuild command
Time

#2

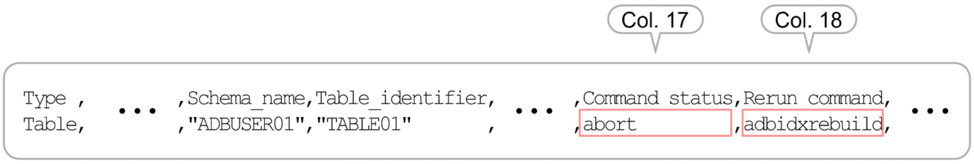
If the `KFAA80204-I` message has been issued with return code `0` or `4`, index rebuild processing has been completed. In such a case, there is no need to re-execute the `adbidxrebuild` command.

If the `KFAA80204-I` message has not been issued, execute the command shown below and check line 2 in the output results (table summary information) (line 1 in the output results contains the column titles).

```
adbdbstatus -c table -n name-of-table-for-which-adbidxrebuild-command-was-canceled
```

Specify *name-of-table-for-which-adbidxrebuild-command-was-canceled* in the format *schema-name*`.`*table-identifier*.

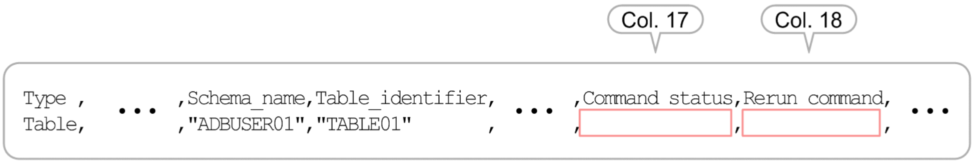
■ When the value of the `Command_status` column (column 17) is `abort` and the value of the `Rerun_command` column (column 18) is `adbidxrebuild`



```
Type ,   ... ,Schema_name,Table_identifier,  ... ,Command_status,Rerun_command,  ...
Table,       ,"ADBUSER01","TABLE01"       ,      ,abort         ,adbidxrebuild,
```

Index rebuild processing was not completed. Re-execute the `adbidxrebuild` command.

■ When no values are displayed in the `Command_status` column (column 17) and the `Rerun_command` column (column 18)



```
Type ,   ... ,Schema_name,Table_identifier,  ... ,Command_status,Rerun_command,  ...
Table,       ,"ADBUSER01","TABLE01"       ,      ,              ,             ,
```

Index rebuild processing has been completed. There is no need to re-execute the `adbidxrebuild` command.

#3

If the `adbidxrebuild` command was canceled while it was retrieving table data from the first chunk, the index rebuild processing is restarted from the beginning when the `adbidxrebuild` command is re-executed (the re-execute facility is not applied).

However, if all indexes to be rebuilt are deleted before the `adbidxrebuild` command is re-executed, only command termination processing is performed.

#4

In the following cases, index rebuild processing resumes from the point of table data retrieval in the chunk in which processing stopped:

- The `adbidxrebuild` command is re-executed with the `--create-temp-file` option specified.
- After the `adbimport` command is stopped, the `adbidxrebuild` command executed with the `--create-temp-file` option specified is canceled.

However, if all indexes to be rebuilt are deleted before the `adbidxrebuild` command is re-executed, only command termination processing is performed.

#5

To determine whether the `adbidxrebuild` command was canceled while it was processing the final chunk for the target table, check the `KFAA80211-I` message to see whether the current chunk ID value is the same as the total number of chunks.

Example of the KFAA80211-I message (when the command was canceled while it was processing the final chunk for the target table)

```
KFAA80211-I Processing for the chunk will start. (chunk id = 3) (3/3)
```

16.5.5 Changing command options and index rebuild options during re-execution

This subsection explains the effects of changing command options and index rebuild options when the `adbidxrebuild` command is re-executed.

(1) Changing command options

Whether changed command options take effect depends on the cancellation timing of the `adbidxrebuild` command. The following table shows the processing at the restart when command options have been changed.

Table 16-5: Processing at the restart when command options have been changed

No.	Command option that has been changed	Timing of the cancellation[#1]		
		During table data retrieval processing	During sort processing	During creation of B-tree and text indexes
1	`-u` option	N	N	N
2	`-p` option	N	N	N
3	`-g` option	Y	C	C
4	`-w` option	Y	C	C
5	`-z` option	Y	Y	Y
6	`-i` option[#2]	Y[#3]	--	--
7	`--create-temp-file`	Y[#4]	--	--
8	`--force`	Not applicable (because when `--force` is specified, the processing is restarted forcibly from the beginning)[#5]		
9	*table-to-be-processed*	N	N	N

Legend:

Y: Change takes effect.

C: For a single-chunk table, change is ignored (the setting in effect before the change is used for execution). For a multi-chunk table, change takes effect beginning with the processing of the next chunk. If the `adbidxrebuild` command is canceled while the final chunk is being processed, the change is ignored.

N: If a change is made, the command will not re-execute.

--: Change is ignored (the setting in effect before the change is used for execution).

#1

For details about the timing of cancellation, see the following:

- Single-chunk table: Table 16-3: Whether the re-execute facility is applied depending on when the adbidxrebuild command was canceled (for a single-chunk table)

- Multi-chunk table: Table 16-4: Restart processing depending on when the adbidxrebuild command was canceled (for a multi-chunk table)

#2

Do not change the value of the -i option when you use the re-execute facility for a multi-chunk table. If the -i option value is changed in such a case, the change might or might not take effect, depending on conditions, and the re-execute facility might become unavailable.

#3

In the case of a multi-chunk table, the change takes effect only when the processing is restarted at the point of data retrieval from the first chunk. If the processing is restarted at a subsequent chunk, the change is ignored.

#4

If, after the adbimport command is stopped, the adbidxrebuild command executed with the --create-temp-file option specified is canceled, the adbidxrebuild command will have the --create-temp-file option specified the next time it is executed. In this case, index rebuild processing resumes from the point of table data retrieval. (If the table to be processed is a multi-chunk table, index rebuild processing resumes from the point of table data retrieval in the chunk that was being processed when the adbimport command stopped.)

Even if you change the specification of the --create-temp-file option when re-executing the adbidxrebuild command, the change is invalid. However, if you specify the --force option, this specification goes into effect.

#5

If, after the adbimport command is stopped, the adbidxrebuild command executed with the --force option specified is canceled, the adbidxrebuild command will have the --force option specified the next time it is executed. In this case, index rebuild processing is forcibly re-executed from the beginning. (The KAFF80215-I message is issued.)

Even if you change the specification of the --force option when you re-execute the adbidxrebuild command, the change is invalid.

(2) Changing index rebuild options

Whether changed index rebuild options take effect depends on the cancellation timing of the adbidxrebuild command. The following table shows the processing at the restart when index rebuild options have been changed.

Table 16-6: Processing at the restart when index rebuild options have been changed

No.	Index rebuild option that has been changed	Timing of the cancellation#		
		During table data retrieval processing	During sort processing	While creating B-tree and text indexes
1	adb_idxrebuild_rthd_num	Y	Y	Y
2	adb_idxrebuild_scan_buff_size	Y	C	C
3	adb_idxrebuild_buff_blk_num	Y	C	C
4	adb_idxrebuild_sort_buff_size	Y	Y	C
5	adb_idxrebuild_dvix_rdbuff_size	Y	Y	Y
6	adb_idxrebuild_dvix_wtbuff_size	Y	Y	Y
7	adb_idxrebuild_txt_buff_size	Y	Y	Y

Legend:

Y: Change takes effect.

C: For a single-chunk table, change is ignored but checking of the number of threads will be performed. For a multi-chunk table, change takes effect beginning with the processing of the next chunk. If the `adbidxrebuild` command is canceled while the final chunk is being processed, the change is ignored.

#

For details about the timing of cancellation, see the following:

- Single-chunk table: Table 16-3: Whether the re-execute facility is applied depending on when the adbidxrebuild command was canceled (for a single-chunk table)

- Multi-chunk table: Table 16-4: Restart processing depending on when the adbidxrebuild command was canceled (for a multi-chunk table)

16.5.6 If an error occurs during re-execution of the adbidxrebuild command

If an error occurs during re-execution of the `adbidxrebuild` command, identify the cause of the error and take the necessary actions according to the following procedure.

Procedure

1. Check whether the `KFAA50244-E` message has been issued.

   If the `KFAA50244-E` message has been issued, an access error has occurred on a command status file (file under the `$DBDIR/ADBSYS/ADBUTL` directory). Execute the `adbidxrebuild` command with the `--force` option specified.

   For details, see (2) Executing the adbidxrebuild command with the --force option specified.

   If the `KFAA50244-E` message has not been issued, proceed to step 2.

2. Check whether the `KFAA50247-E` message has been issued.

   If the `KFAA50247-E` message has been issued, an access error occurred on a temporary work file. Execute the `adbidxrebuild` command with the `--create-temp-file` option specified.

   For details, see (1) Executing the adbidxrebuild command with the --create-temp-file option specified.

   If the `KFAA50247-E` message has not been issued, proceed to step 3.

3. Check whether a message other than `KFAA50244-E` and `KFAA50247-E` has been issued.

   Check the message that was issued and identify the cause of the error. After removing the cause of the error, execute the `adbidxrebuild` command.

(1) Executing the adbidxrebuild command with the --create-temp-file option specified

If the `KFAA50247-E` message is issued during re-execution of the `adbidxrebuild` command, an access error has occurred on a temporary work file. Execute the `adbidxrebuild` command with the `--create-temp-file` option specified.

Important

It is assumed that the `--create-temp-file` option will be specified if the temporary work file necessary for re-execution (index record file or sort result file) does not exist. If the `adbidxrebuild` command is executed when a temporary work file exists, the existing file is re-built. However, if you do not have write or read privileges for the applicable file, an access error might occur during execution of the adbidxrebuild command. (The KFAA30959-E, KFAA40204-E, KFAA40205-E, KFAA40214-E, KFAA41205-E, or KFAA41206-I message is issued. 13 (EACCES) is output for the error number.) In such a case, first delete the applicable file using the OS's rm command or a similar method, and then re-execute the adbidxrebuild command.

(2) Executing the adbidxrebuild command with the --force option specified

If the KFAA50244-E message is issued during re-execution of the adbidxrebuild command, an access error has occurred on a command status file (file under the $DBDIR/ADBSYS/ADBUTL directory).

In this case, take the necessary actions according to the following procedure.

Procedure

1. Delete the temporary work files.
   Delete the files under the directories listed below (temporary work files) using the OS's rm command or a similar method. If there are any files for which you do not have write or read privileges, you might not be able to execute the adbidxrebuild command.
   - If the -w option was specified during execution of the previous command that was canceled:
     The temporary work file storage directory specified in the -w option
   - If the -w option was omitted during execution of the previous command that was canceled:
     The $DBDIR/ADBWORK directory
2. Execute the adbidxrebuild command with the --force option specified.
   Execute the adbidxrebuild command with the --force option specified.

16.6 Notes

1. Because the `adbidxrebuild` command uses SQL statements to retrieve table data, we recommend that you execute the command in an environment in which out-of-order execution is applied. For details about environments in which out-of-order execution is applied, see *Database retrieval method (out-of-order execution)* in the *HADB Setup and Operation Guide*.

2. Use `X'0A'` (`LF`) or `X'0D0A'` (`CRLF`) for the line break code in any file specified in the options of the `adbidxrebuild` command.

3. Index rebuild processing continues even if there is a duplicated key value in a unique index (B-tree index for the primary key or B-tree index for which `UNIQUE` was specified in the `CREATE INDEX` statement). The `KFAA61205-W` warning message indicating detection of duplicate key values is issued. In such a case, the table and the B-tree index subject to index rebuild processing violate the uniqueness constraint. Locate the duplicate key values and delete the corresponding rows (a B-tree index that violates the uniqueness constraint is in the status called uniqueness constraint violation). Then re-execute the `adbidxrebuild` command and rebuild the corresponding B-tree index.

4. Use the `adbcancel` command to cancel execution of the `adbidxrebuild` command. If you want to use anything other than the `adbcancel` command, use a method such as the `adbstop --force` command or the OS's `kill` command to forcibly terminate the server process. After canceling the `adbidxrebuild` command, re-execute the `adbidxrebuild` command as needed based on the explanation in 16.5 Handling abnormal termination of the adbidxrebuild command.

   For details about how to use the `adbcancel` command to suspend an executing command, see 1.6 When suspending (forcibly terminating) an executing command.

5. Index rebuild processing is executed in server processes. Therefore, using the **Ctrl**+**C** keys to forcibly terminate the `adbidxrebuild` command process will not interrupt index rebuild processing that is being performed by server processes (this processing continues). When the index rebuild processing is completed, the command outputs the `KFAA81001-I` message to the message log file.

6. Work files (temporary work files) are created during index rebuild processing. These temporary work files are deleted automatically when the `adbidxrebuild` command terminates normally (the return code is `0` or `4`). If the `adbidxrebuild` command does not terminate normally, these work files might not be deleted.

   Because some of the temporary work files will be needed when the re-execute facility is used, do not delete temporary work files before you use the re-execute facility. If you do delete them, the re-execute facility will not function correctly.

7. Because temporary work files might cause a shortage of disk space, we recommend that you delete temporary work files from time to time in the following cases:

   - There will no need to re-execute the `adbidxrebuild` command.
   - There is no need to re-execute the `adbimport` command.

   Check the following directories periodically for any remaining work files and delete any that are found:

   - `$DBDIR/ADBSYS/ADBUTL`
   - Temporary work file storage directories specified in the `-w` option (or in `$DBDIR/ADBWORK` when the `-w` option was omitted)

8. The following table shows how index creation processing is inherited from the `adbimport` command.

Table 16-7: Inheriting index creation processing from the adbimport command

No.	Canceled command[#1]	Command to be executed following cancelation			Whether processing is inherited
		Command name	Option specification		
			--create-temp-file option	--force option	
1	adbimport	adbidxrebuild	Specified	Not specified	Y[#2, #3]
2			Not specified	Specified	Y[#2, #4, #5]
3			Not specified	Not specified	N

Legend:

Y: Index creation processing can be inherited.

N: Index creation processing cannot be inherited. (The KFAA50201-E message is issued.)

#1

To identify the canceled command, output the table summary information by executing the adbdbstatus command. The name of the command that was canceled is output to the Rerun_command column in the output results. For details about the adbdbstatus command, see 13. adbdbstatus (Analyze the Database Status).

If the adbidxrebuild command executed after the cancelation of the adbimport command is canceled, adbidxrebuild is displayed in the Rerun_command column. If the --create-temp-file option was specified for the canceled adbidxrebuild command, re-execute the adbidxrebuild command with the --create-temp-file option specified. In this case, the information about the cancelation of the adbimport command is saved in a command status file (file under the $DBDIR/ADBSYS/ADBUTL directory), and therefore indexes are rebuilt for the chunk that was being processed when the adbimport command was canceled.

#2

Executing the adbidxrebuild command rebuilds all indexes that were defined for the table to be processed. The -i option of the adbidxrebuild command is ignored even if it is specified.

#3

If the KFAA50247-E message is issued during re-execution of the adbimport command (no temporary work file exists), execute the adbidxrebuild command with the --create-temp-file option specified. In this way, index creation processing can be inherited.

#4

When an index is defined for the table to be processed while the adbimport command is stopped, and the KFAA50209-E message is issued during re-execution of the adbimport command (an index in unfinished status exists), you can inherit index creation processing by executing the adbidxrebuild command with the --force option specified.

To identify the indexes in unfinished status, output the index summary information by executing the adbdbstatus command, and check the Unfinished column in the output results. If an index is in unfinished status, unfinished is output to the Unfinished column in the output results. For details about the adbdbstatus command, see 13. adbdbstatus (Analyze the Database Status).

If the adbimport command has not been canceled, execute the adbidxrebuild command without the --force option specified.

#5

If the KFAA50244-E message is issued while the adbimport command is stopped (no command status file exists), execute the adbidxrebuild command with the --force option specified. In this way, index creation processing can be inherited.

9. Note the following if the `adbidxrebuild` command is executed and canceled after cancelation of the `adbimport` command:

   - When the `adbidxrebuild` command with the `--create-temp-file` option specified is canceled after cancelation of the `adbimport` command

     If you re-execute the `adbidxrebuild` command, the command resumes index rebuild processing at the point of table retrieval in the chunk for which processing was canceled even if you do not specify the `--create-temp-file` option. Note, however, that if you specify the `--force` option when you re-execute the `adbidxrebuild` command, specification of the `--force` option goes into effect.

   - When the `adbidxrebuild` command with the `--force` option specified is canceled after cancelation of the `adbimport` command

     If you re-execute the `adbidxrebuild` command, index rebuild processing is forcibly re-executed from the beginning even if you do not specify the `--force` option. Therefore, even if you specify the `--create-temp-file` option when you execute the command, the option is ignored. When index rebuild processing is forcibly re-executed from the beginning, the `KAFF80215-I` message is issued.

10. Make sure that there is no duplication of the absolute paths for the directories and files specified in the `adbidxrebuild` command's options. This applies to the following directories and files:

    - Temporary work file storage directories (specified in the `-w` option or specified in the directory path file specified in the `-w` option)
    - Directory path files specifying the storage locations of temporary work files (specified in the `-w` option)
    - Index identifier file (specified in the `-i` option)
    - Index rebuild option file (specified in the `-z` option)

    Make sure that the absolute paths of these directories and files are unique from those of any file names used by any programs other than the `adbidxrebuild` command. If a duplicate file name is used, data in the file might be corrupted. Also make sure that symbolic links and relative paths are specified correctly.

11. If the `adbidxrebuild` command is executed while another command or application program is running, HADB might not be able to allocate the required number of processing real threads. If this happens, HADB waits until the required number of processing real threads can be allocated before executing the `adbidxrebuild` command. If allocation of processing real threads is placed in wait status, see the topic *Points to consider about the number of processing real threads to be used during command execution* in *Points to consider when executing commands concurrently* in *Preparing Resources* in the *HADB Setup and Operation Guide* and take appropriate action.

12. To reduce this command's execution time, see the topic *Reducing the execution time of the adbidxrebuild command* in *Tuning to shorten command execution time* in *Tuning* in the *HADB Setup and Operation Guide*.

13. Do not delete a user who is executing the `adbidxrebuild` command or revoke such a user's `REBUILD INDEX` privilege while indexes are being rebuilt. If the executing user is deleted or the user's privileges are revoked by mistake and the `adbidxrebuild` command terminates abnormally, it is not possible to use the re-execute facility provided by the `adbidxrebuild` command.

    If the executing user is deleted or the user's privileges are revoked by mistake, take either of the following corrective actions:

    - As the same user, make the `adbidxrebuild` command executable, and then re-execute this command. For details about the corrective action to take, see the topic *Managing HADB users* or *Managing access privileges* in *Unscheduled Operations* in the *HADB Setup and Operation Guide*.
    - As a different user who has `REBUILD INDEX` privileges for the table to be processed, execute the `adbidxrebuild` command with the `--force` option specified.

17 adbimport (Import Data)

This chapter explains the functionality provided by the `adbimport` command and how to use the command.

Note that the term *tables* in this chapter means base tables.

17.1 Description

This section explains the functionality provided by the adbimport command and the threads it uses.

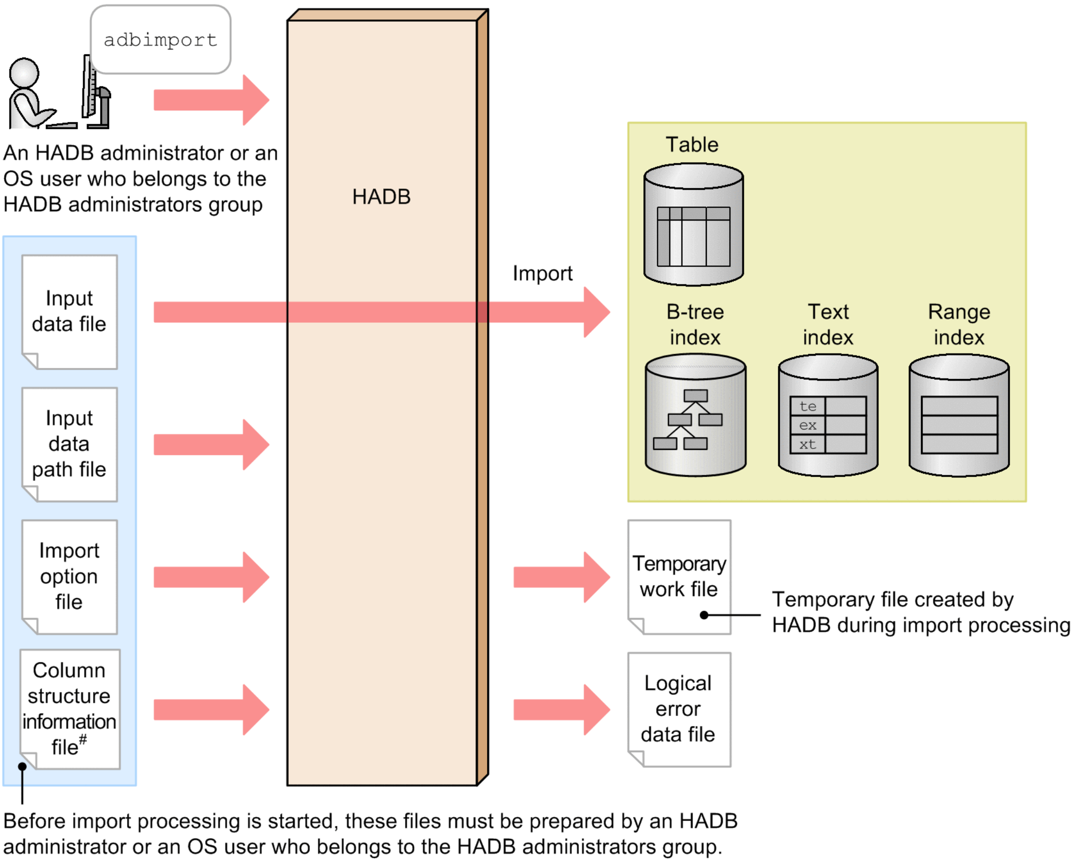
When executed, the adbimport command establishes a connection to the HADB server.

17.1.1 Functional overview

The adbimport command is used to import data into a table. If a range index has been defined for the target table, the command creates the range index while it is importing the table data. If a B-tree index or a text index has been defined for the target table, the command creates the B-tree index or text index after table data has been imported (if a primary key has been defined for the target table, the command creates the B-tree index corresponding to the primary key after the table data has been imported).

This series of processing steps is called the *data import processing*. The following figure provides an overview of data import processing.

Figure 17-1: Overview of data import processing


adbimport
An HADB administrator or an OS user who belongs to the HADB administrators group
HADB
Input data file
Input data path file
Import option file
Column structure information file#
Import
Table
B-tree index
Text index
te
ex
xt
Range index
Temporary work file
Temporary file created by HADB during import processing
Logical error data file
Before import processing is started, these files must be prepared by an HADB administrator or an OS user who belongs to the HADB administrators group.

#: A column structure information file is specified when the sort order of column definitions or the number of columns in the input data does not match those in the target table.

Explanation:

- When the adbimport command is executed, the command imports the data contained in the input data files into a table.

If the table to be imported is a row store table, data in the input data files is stored in the database in row store format. If the table to be imported is a column store table, data in the input data files is stored in the database in column store format.

Note that if the table to be imported is a column store table, data in each column is compressed before being stored in the database. Multiple column-data compression types are available. The HADB server automatically selects the compression type based on the data in the input data files. However, if the column-data compression type is specified when a column store table is defined, column data is compressed with the specified compression type.

> **Note**
>
> - For details about row store tables and column store tables, see *Row store tables and column store tables* in the *HADB Setup and Operation Guide*.
> - For details about the types of column-data compression, see *Column-data compression types for column store tables* in *Criteria for selecting row store tables and column store tables* in the *HADB Setup and Operation Guide*.

- If a range index has been defined for the table, the command creates the range index when it has finished storing the table data in each segment. If a B-tree index or a text index has been defined for the table, the command creates the B-tree index or text index after table data has been imported (if a primary key is defined for the table, the command creates the B-tree index corresponding to the primary key after it has finished importing the table data).

The following explains the files:

- *Input data files*

  These files store the data that is to be imported. You must create input data files in one of the following formats:

  - CSV format
  - Fixed-length format

  For details about the CSV format, see 17.5 Format of input data files (CSV format specifications).

  For details about the fixed-length format, see 17.6 Format of input data files (fixed-length format specifications).

  The following compression files can also be used as input data files:

  - CSV-format file compressed in GZIP format
  - Fixed-length format file compressed in GZIP format

  Files compressed in any other format cannot be used. For details about the compression format of input data files, see 17.7 Compression format of input data files.

  > **Note**
  >
  > The format of the input data files is the same regardless of whether the table to be imported is a row store table or column store table.

- *Input data path file*

  This file is used to specify a list of path names of the storage locations of the input data files.

- *Import option file*

  This file contains the options that are to be used to set up an execution environment for the `adbimport` command. The options specify information such as the output destination for the logical error data file.

- *Column structure information file*

This file specifies the relationship between the target table and the input data. It is specified when there is a difference between the input data and the sort order of column definitions or the number of columns in the target table.

- *Temporary work files*

  These are work files that are created temporarily during data import processing. These files are deleted automatically when data import processing terminates normally.

- *Logical error data file*

  This file is used for the output of logical error data (such as data with invalid data lengths) contained in the input data files. For details about the logical error data file, see (1) Logical error data output function in 17.1.2 Optional functions of the adbimport command.

17.1.2 Optional functions of the adbimport command

The `adbimport` command provides the following optional functions:

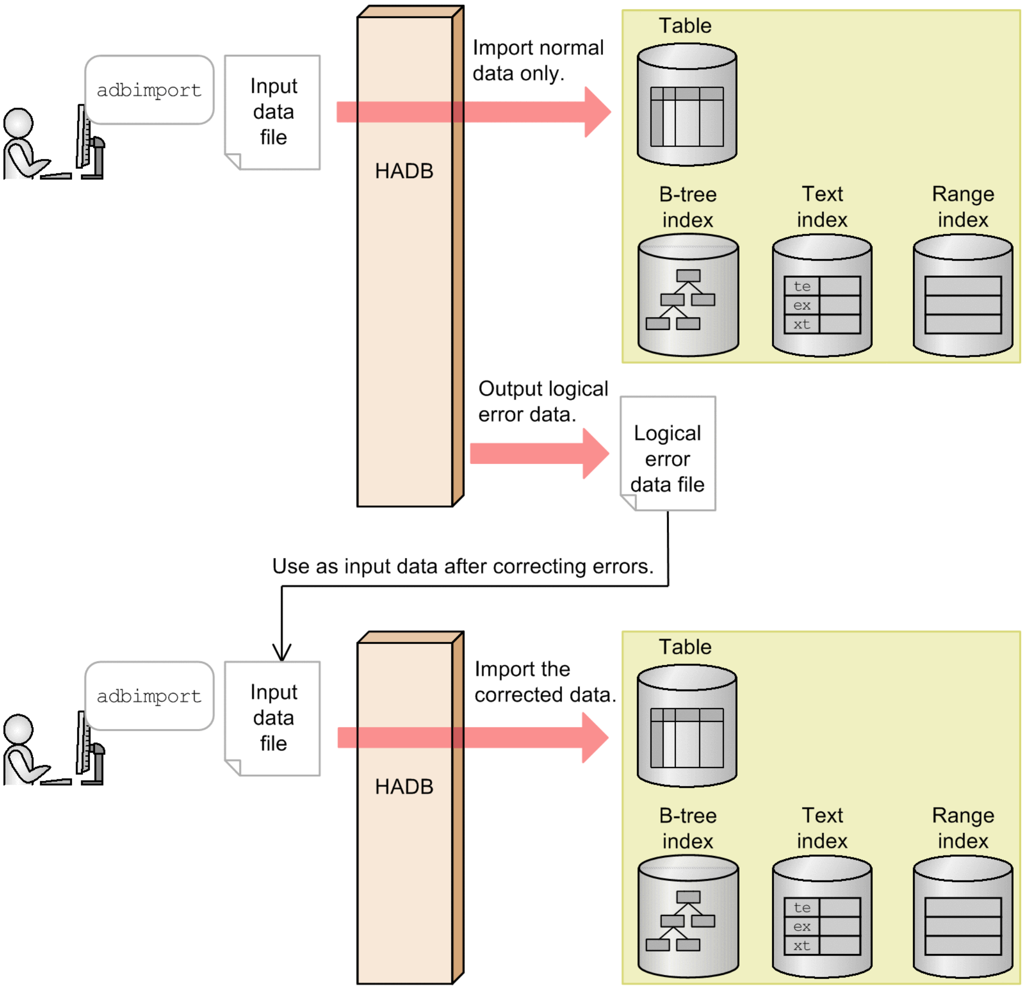
- Logical error data output function
- Re-execute facility that starts from the processing to create B-tree indexes and text indexes
- Background-import facility

(1) Logical error data output function

When the `adbimport` command imports data, it imports only valid data. It does not import invalid data (logical error data). Any row of data that contains logically erroneous data is output to the logical error data file. You can correct logical error data that has been output to the logical error data file, and then re-execute the `adbimport` command to import the corrected data.

The following figure provides an overview of the logical error data output function.

Figure 17-2: Overview of the logical error data output function



■ **About logical error data**

The following constitute logical error data:

- Data with an invalid data length
- Data with invalid enclosing characters
- Data resulting in an error during input data conversion processing
- Input data to be imported does not exist
- Input data containing more or fewer columns than exist in the table
- Column data subject to the NOT NULL constraint that contains an empty string#
- Input data files with an invalid format

#

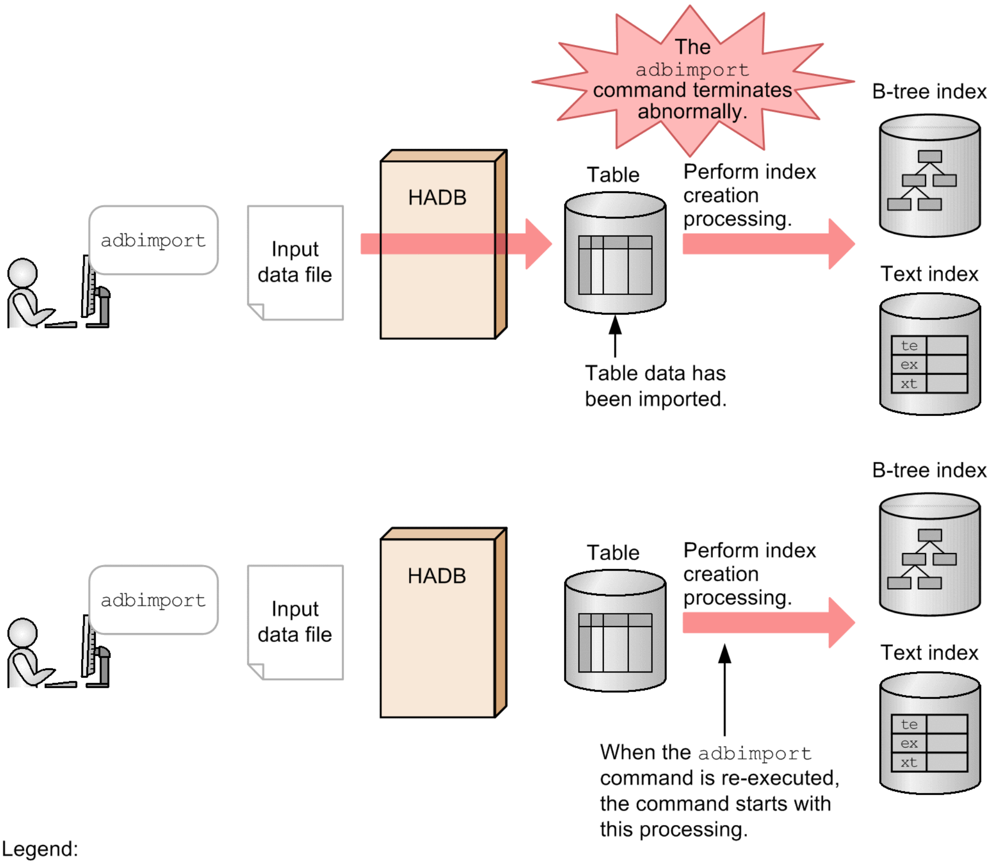
An *empty string* is a string that does not contain any field data in the input data.

(2) Re-execute facility that starts from the processing to create B-tree indexes and text indexes

Data import processing involves importing table data and then creating B-tree and text indexes. If the adbimport command terminates abnormally during B-tree and text index creation, the command restarts data import processing beginning with B-tree and text index creation when the command is re-executed. This reduces the time required to re-execute data import processing.

The following figure provides an overview of the re-execute facility that starts from the processing to create B-tree indexes and text indexes.

Figure 17-3: Re-execute facility that starts from the processing to create B-tree indexes and text indexes


The adbimport command terminates abnormally.
B-tree index
adbimport
Input data file
HADB
Table
Perform index creation processing.
Text index
te ex xt
Table data has been imported.
B-tree index
adbimport
Input data file
HADB
Table
Perform index creation processing.
Text index
te ex xt
When the adbimport command is re-executed, the command starts with this processing.

Legend:

Index creation processing: Creation of B-tree and text indexes

The re-execute facility that starts from the processing to create B-tree indexes and text indexes is applied when all of the following conditions are satisfied:

- Either or both of the following indexes have been defined for the table to be imported:#
    - B-tree indexes (including when a primary key has been defined for the target table).
    - Text indexes

  #

  If all the B-tree indexes and text indexes for the table to be processed are deleted before the `adbimport` command is re-executed, only command termination processing is performed.
- The table data has been imported (when table data import processing is completed, the `KFAA80203-I` message is issued).
- The processing was canceled for one of the following reasons:
    - The `adbimport` command terminated abnormally.
    - The `adbcancel` command was used to forcibly terminate the `adbimport` command.
    - The `adbstop --force` command was used to forcibly terminate the HADB server.

- The server process was forcibly terminated by the OS's `kill` command or because of a power shutdown.

If the `adbimport` command stops while the table data is being imported (before the `KFAA80203-I` message is issued), the re-execute facility that starts from the processing to create B-tree indexes and text indexes is not applied.

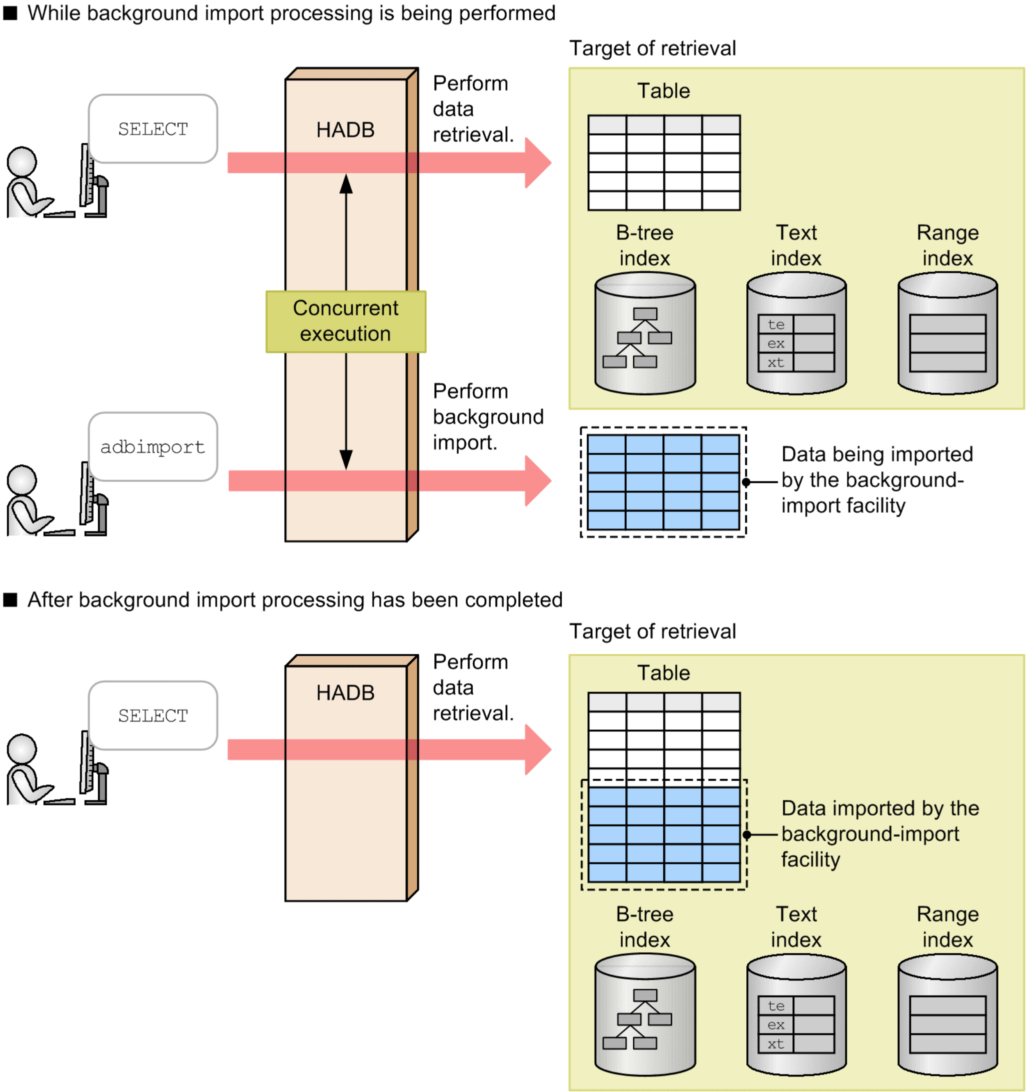
If the `adbimport` command is re-executed with the `--force` option specified, data import processing is forcibly executed from the beginning, and therefore the re-execute facility that starts from the processing to create B-tree indexes and text indexes is not applied.

(3) Background-import facility

The background-import facility enables you to perform data retrieval and import processing on the same table concurrently (data is not subject to retrieval while it is undergoing background import processing).

The following figure provides an overview of the background-import facility.

Figure 17-4: Overview of the background-import facility.


■ While background import processing is being performed
Target of retrieval
SELECT
HADB
Perform data retrieval.
Table
B-tree index
Text index
Range index
te
ex
xt
Concurrent execution
adbimport
Perform background import.
Data being imported by the background-import facility
■ After background import processing has been completed
Target of retrieval
SELECT
HADB
Perform data retrieval.
Table
Data imported by the background-import facility
B-tree index
Text index
Range index
te
ex
xt

The background-import facility is applied only if all the following conditions are satisfied:

- `CHUNK` was specified when the table to be imported was defined. (The table to be imported is a multi-chunk table.)

- The data import processing is performed in the addition mode (the `-d` option is omitted from the `adbimport` command).
- The `-b` option is specified in the `adbimport` command.
- The total of the number of chunks that will be created during the background import processing added to the number of chunks already created for the table so far will not exceed the value specified for `CHUNK` when the table was defined.

Note

The following information is output to the message log file while the background import processing is being performed:

- The remaining number of chunks that can be created for the table
- The chunk IDs of the chunks that have been created during background import processing

Note

■ **Background import for creating chunks in wait status**

If you want to exclude data imported by background import as a target of processing by data manipulation SQL statements, specify the `--status wait` option in addition to the `-b` option. Data is imported to chunks that are in wait status during background import processing.

Note that the `PURGE CHUNK` and `TRUNCATE TABLE` data manipulation SQL statements can be executed on chunks that are in wait status.

17.1.3 Relationships among threads used by the adbimport command

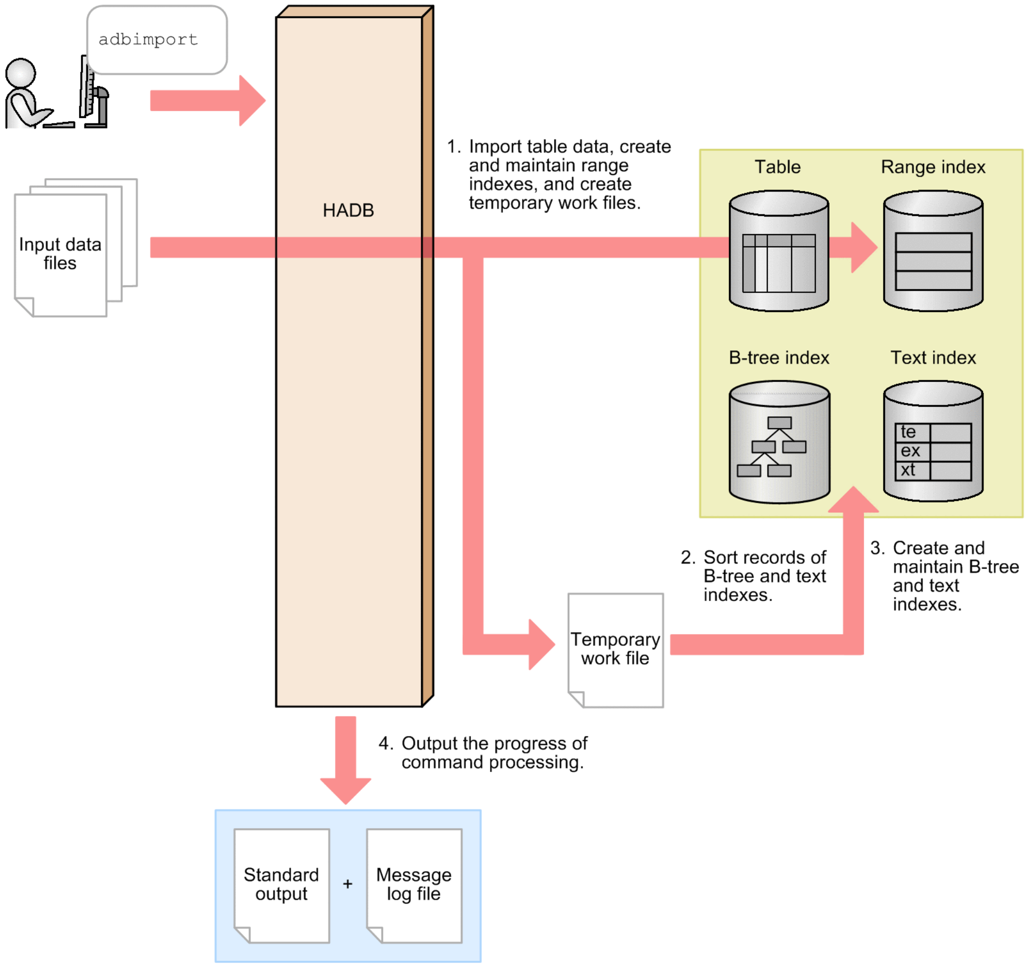
The `adbimport` command uses processing real threads in the following processing:

1. Importing table data, creating and maintaining range indexes, and creating temporary work files
   One or more processing real threads are used.
2. Sorting records of B-tree and text indexes
   One or more processing real threads are used.
3. Creating and maintaining B-tree and text indexes
   One or more processing real threads are used.
4. Outputting the progress of command processing
   One processing real thread is used.

Steps 1 to 3 are performed in sequence. Step 4 is always performed while steps 1 to 3 are in progress.

The following figure shows the processing of the `adbimport` command using processing real threads. The preceding numbers correspond to the numbers in the figure.

Figure 17-5: Processing of the adbimport command using processing real threads


adbimport
HADB
Input data files
1. Import table data, create and maintain range indexes, and create temporary work files.
Table
Range index
B-tree index
Text index
te
ex
xt
2. Sort records of B-tree and text indexes.
3. Create and maintain B-tree and text indexes.
Temporary work file
4. Output the progress of command processing.
Standard output
+
Message log file

The number of processing real threads used by the `adbimport` command for each step is automatically calculated from the *number of processing real threads to be used by the `adbimport` command*. For details about the *number of processing real threads to be used by the `adbimport` command*, see the description of the import option adb_import_rthd_num in (b) Options related to performance in (1) Specification format in 17.2.2 Format of import options.

17.2 Specification format for the adbimport command

This section explains the specification format, the import option format, and the column structure information option format for the `adbimport` command.

The following OS users can execute the `adbimport` command:

- HADB administrator
- OS users who belong to the HADB administrators group

17.2.1 Explanation of the specification format and options

The values of the options listed below and whether the options are to be specified depend on the format of the input data files. Specify the required options as appropriate for the input data files to be used for importing data. For details about the individual options, see -f, -r, -k, and -s.

When the input data files are in CSV format

- Omit the `-f` option or specify `csv` in the `-f` option.
- Specify the `-r` option according to the relationship between the target table and input data.
- You can specify the `-k` and `-s` options.

When the input data files are in fixed-length format

- Specify `fix` in the `-f` option.
- You must specify the `-r` option.
- Do not specify the `-k` or `-s` option.

(1) Specification format

```
adbimport
    -u authorization-identifier
  [-p password]
  [-d]
  [-k {enclosing-character|none}]
  [-s {delimiter |tab}]
  [-g output-interval-for-data-import-processing-progress-messages]
  [-w {temporary-work-file-storage-directory-name
        |name-of-directory-path-file-specifying-temporary-work-file-storage-locations
}]
  [-z import-option-file-name]
  [-e]
  [--force]
  [-b]
  [--status wait]
  [-m comment-to-be-set-in-chunk]
  [-f {csv|fix}]
  [-r column-structure-information-file-name]
  table-to-be-processed
  input-data-path-file-name
```

Important

You must specify the table to be processed and the input data path file name in this order as the last options. Make sure that the table to be processed is specified before the input data path file name.

(2) Relationship between data import patterns and options to be specified (types of chunks to which data is to be added)

The combination of options you need to specify depends on the data import pattern you want to perform.

When data is imported to a multi-chunk table, the chunk to which data is to be added varies depending on the specification of the `-d` and `-b` options. If the `-b` option is specified together with the `--status wait` option, chunks in wait status are created and then data is added.

Note

For details about chunks, see *Managing data in data-import units (chunks)* in the *HADB Setup and Operation Guide*.

The following table shows the relationships among data import patterns and options to be specified (types of chunks to which data is to be added).

For details about the options, see the descriptions of -d, -b, and --status wait in (3) Explanation of options.

Table 17-1: Relationship between data import patterns and options to be specified (types of chunks to which data is to be added)

No.	Data import pattern		Options to specify			Chunk to which data is to be added when data is imported to a multi-chunk table #1
			-d option	-b option	--status wait option	
1	Importing data to a newly defined table		N	N	N	Current chunk
2	Importing data to an existing table without using background import		N	N	N	Current chunk
3	Importing data to an existing table in the background#2	Including chunks for data that has been imported to the target of processing by data manipulation SQL statements (placing the chunks in normal status)	N	Y	N	Chunks in the normal status that were newly created during background import processing
4		Excluding chunks for data that has been imported from the target of processing by data manipulation SQL statements (placing the chunks in wait status)#3	N	Y	Y	Chunks in the wait status that were newly created during background import processing#4
5	Importing new data to an existing table after all existing data has been deleted		Y	N	N	Newly created chunks (because data is imported in creation mode, all the existing chunks are deleted).

Legend:

Y: The option is specified.

N: The option is omitted.

Note:

Specifying the `-d` option together with the `-b` option will result in an error.

Specifying the `--status wait` option without specifying the `-b` option will result in an error.

#1

If the `-m` option is specified, a comment can be set in the chunk to which data is to be added.

#2

For a single-chunk table, an error results (background import processing cannot be applied to a single-chunk table).

#3

The `PURGE CHUNK` and `TRUNCATE TABLE` statement can be executed.

#4

Chunks in wait status cannot be current chunks.

(3) Explanation of options

In option explanations, options indicated with [Multi-chunk table] can be specified when data is imported to a multi-chunk table. Those options cannot be specified when data is imported to a single-chunk table.

- `-u` *authorization-identifier*

  ~<character string>((1 to 100 bytes))

  Specifies the authorization identifier of the HADB user who executes the `adbimport` command. You must have the following two privileges to execute the `adbimport` command:

  - `CONNECT` privilege
  - `IMPORT TABLE` privilege for the table to be processed

  For this option, specify an authorization identifier in the range from 1 to 100 bytes. Note that the byte count (1 to 100 bytes) does not include the double quotation marks used to enclose the authorization identifier.

  > **Important**
  >
  > If the character string used as the authorization identifier includes any lowercase letter or a backslash (\\), make sure that you check the rules for specifying authorization identifiers. For the rules for specifying authorization identifiers, see 1.4.2 Rules for specifying authorization identifiers and passwords.

  > **Important**
  >
  > The authorization identifier that is specified in the `-u` option must not be changed while the re-execute facility that starts from the processing to create B-tree indexes and text indexes is being used. Do not delete a user who is executing the `adbimport` command or revoke that user's `IMPORT TABLE` privilege while data is being imported.
  >
  > If the executing user is deleted or the user's privileges are revoked by mistake, take either of the following corrective actions:

- As the same user, make the `adbimport` command executable and re-execute this command. For details about the corrective action to take, see the topic *Managing HADB users* or *Managing access privileges* in *Unscheduled Operations* in the *HADB Setup and Operation Guide*.
- As a different user who has `REBUILD INDEX` privileges for the table to be processed, execute the `adbidxrebuild` command with the `--force` option specified.

  For details about how to execute the `adbidxrebuild` command with the `--force` option specified and related notes, see (2) Executing the adbidxrebuild command with the --force option specified in 17.10.7 If an error occurs during re-execution of the adbimport command.

- `-p` *password*

  ~<character string>((1 to 255 bytes))

  Specifies the password for the authorization identifier that is specified in the `-u` option.

  Important

  If the password includes a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), make sure that you check the rules for specifying passwords. For the rules for specifying passwords, see 1.4.2 Rules for specifying authorization identifiers and passwords.

  If the `adbimport` command is executed with this option omitted, a message prompting the user to enter a password is displayed. In an environment in which a password cannot be entered from the standard input, such as when the command is executed in the background, make sure that you do not omit this option.

- `-d`

  Specifies that data import processing is to be performed in creation mode. If this option is omitted, the command performs data import processing in addition mode.

  In creation mode, the command deletes all existing data from the table and then imports the input data. In addition mode, the command imports the input data while retaining any data that already exists in the table.

  When you import data to a newly defined table, there is no need to specify this option.

- `-k {`*enclosing-character*`|none}`

  ~<character string>((1 byte))<<`"`>>

  Consider specifying this option for input data files in CSV format. Do not specify this option for input data files in fixed-length format.

  Specifies the character that is to be used as the enclosing character for demarcating field data in the input data files. If this option is omitted, the double quotation mark (`"`) is assumed to be the enclosing character.

  If no enclosing character is used for the field data in the input data files, specify `none` in this option.

  The enclosing character must be a single-byte character. A character string of 2 or more bytes cannot be specified. None of the following characters are permitted to be the enclosing character:

  - Space, tab, asterisk (`*`)
  - The character that is used as the delimiter (the value of or default value for the `-s` option)

  If numeric data, datetime data, or binary data is defined for a table that is to be imported, do not specify any character used in that numeric, datetime, or binary data as the enclosing character. If such a character is specified as the enclosing character, an input data conversion error might occur.

Important

- If you specify a single quotation mark (`'`) as the enclosing character, enclose it in double quotation marks (`"`).

  Example: If you specify a single quotation mark (`'`) as the enclosing character:

  ```
  adbimport ... -k "'" ...
  ```

- If you specify as the enclosing character a special character used by the shell, such as the vertical bar (`|`), enclose it in either double quotation marks (`"`) or single quotation marks (`'`).

  Example: If you specify the vertical bar (`|`) as the enclosing character:

  ```
  adbimport ... -k "|" ...
  ```

- When you specify this option, either omit the `-f` option or specify `csv` in the `-f` option. If `fix` is specified in the `-f` option, the `adbimport` command will result in an error.
- If you specify an enclosing character in this option, and if a character that is the same as that enclosing character exists in the field data, specify the enclosing character twice in succession in the field data.

  If you specify `none` in this option, no enclosing character for input data is used in any data type. Therefore, there is no need to specify the character twice in succession in the field data.
- You cannot specify `none` in this option if the field data contains a character that is the same as the delimiter or if you want to treat line-ending code as data.
- The following table shows the relationship between input data and the data to be stored according to the value specified in this option.

Table 17-2: Relationship between input data and the data to be stored

No.	Input data	Data to be stored	
		When the enclosing character (double quotation mark (")) is specified in the -k option	When "none" is specified for the -k option
1	`ABC,DEF`	• `ABC` • `DEF`	• `ABC` • `DEF`
2	`"ABC""","DEF"`	• `ABC"` • `DEF`	• `"ABC"""` • `"DEF"`
3	`"ABC,DEF"`	`ABC,DEF`	• `"ABC` • `DEF"`
4	`"ABC,DEF`	--	• `"ABC` • `DEF`
5	`"ABC","DEF\n"`	• `ABC` • `DEF\n`	• `"ABC"` • `"DEF` • `"`

Legend:

--: A logical error occurs.

- `-s` {*delimiter*|`tab`}

  ~<character string>((1 byte))<<,>>

  Consider specifying this option for input data files in CSV format. Do not specify this option for input data files in fixed-length format.

Specifies the character that is to be used as the delimiter for delimiting field data in the input data files. To specify the tab character as the delimiter, specify `tab`. If this option is omitted, the comma (`,`) is assumed to be the delimiter.

Only `tab` or a single-byte character can be specified as the delimiter (except for `tab`, a character string consisting of two or more bytes cannot be specified). None of the following characters can be specified as the delimiter:

- Alphanumeric characters (upper-case letters `A` to `Z`, lower-case letters `a` to `z`, numeric characters `0` to `9`), underscore (_), double quotation mark (`"`), asterisk (`*`)
- The character that is used as the enclosing character (the value of or default value for the `-k` option)

None of the following characters is suitable as the delimiter because the character codes representing these characters might also be used in the input data to represent special meanings:

- Signs (`+`, `-`), forward slash (`/`), colon (`:`), period (`.`), vertical bar (`|`), backslash (`\`), `[`, `]`, `(`, `)`, `{`, `}`, swung dash (~)

> **Important**
>
> - If you specify a single quotation mark (`'`) as the delimiter, enclose it in double quotation marks (`"`).
>
>   Example: If you specify a single quotation mark (`'`) as the delimiter:
>
>   `adbimport ... -s "'" ...`
>
> - If you specify as the delimiter a special character used by the shell, such as the vertical bar (`|`), enclose it in either double quotation marks (`"`) or single quotation marks (`'`).
>
>   Example: If you specify the vertical bar (`|`) as the delimiter:
>
>   `adbimport ... -s "|" ...`
>
> - When you specify this option, either omit the `-f` option or specify `csv` in the `-f` option. If `fix` is specified in the `-f` option, the `adbimport` command will result in an error.

- `-g` *output-interval-for-data-import-progress-messages*

  ~<integer>((0 to 1,000,000,000))<<0>>(in 1,000 rows)

  Specifies the interval at which the data import progress messages are to be output. For example, if you specify `2` in this option, a progress message (`KFAA80205-I`) is output each time 2,000 rows of data have been imported.

  If this option is omitted or `0` is specified in this option, no progress message is output.

- `-w {`*temporary-work-file-storage-directory-name*`|`*name-of-directory-path-file-specifying-temporary-work-file-storage-locations*`}`

  ~<OS path name>((2 to 518 bytes))

  Specifies a storage location for the temporary work files that are created during data import processing, expressed as either of the following:

  1. Absolute path name of the directory that stores temporary work files
  2. Absolute path name of a file (directory path file) that specifies directories for storing temporary work files

  If frequent input/output operations are performed on the temporary work files stored under a single directory, performance might be adversely affected. We recommend that you store temporary work files in directories on multiple disks by using method 2 so that the workload of input/output operations will be distributed among disks. The following shows an example specification using method 2 (specifying a directory path file).

  Example of directory path file

```
/mnt/diska/wwwww
/mnt/diskb/xxxxx
/mnt/diskc/yyyyy
/mnt/diskd/zzzzz
```

The following is a guideline for the number of directories to be specified:

Guideline-for-number-of-directories-to-be-specified = *A* × *B*

A: Number of data storage threads

Use the following formula to determine the value:

value-of-the-import-option adb_import_rthd_num - 1

B: Number of B-tree and text indexes defined for the table subject to import processing

If the number of B-tree and text indexes is not known, use the following SQL statement to determine the number of B-tree and text indexes defined for the table:

```
SELECT "N_INDEX"-"N_RANGE_INDEX"
    FROM "MASTER"."SQL_TABLES"
        WHERE "TABLE_SCHEMA"='authorization-identifier'
           AND "TABLE_NAME"='name-of-table-to-be-imported'
```

- A maximum of 255 directories can be specified in a directory path file.
- The directories specified in the directory path file are used as the storage locations of temporary work files in the order specified. When the last directory specified in the directory path file is used (in this example, `/mnt/diskd/zzzzz`), the first directory (in this example, `/mnt/diska/wwwww`) is then used again.

The following specification rules apply to this option (for both methods 1 and 2):

- Do not specify a forward slash (`/`) at the end of a path name.

For details about how to estimate the sizes of the temporary work files that will be created under the directories specified in the `-w` option, see the topic *Estimating the size of the temporary work file for executing the adbimport command* in the *HADB Setup and Operation Guide*. If this option is omitted, temporary work files are created under `$DBDIR/ADBWORK`.

Provide a dedicated directory to store the temporary work files. If a directory storing other files is used to store temporary work files, data in the existing files might be corrupted. For details about the files whose names must differ from the temporary work file storage directory name, see 17.11 Notes.

If the disk that stores temporary work files runs out of free space, specify another disk with a larger capacity as the storage destination. For details about the actions to take if the disk that stores temporary work files runs out of free space, see *Steps to take in the event of a shortage of disk space for storing temporary work files during command execution* in *Command-related problems* in *Troubleshooting* in the *HADB Setup and Operation Guide*.

- `-z` *import-option-file-name*

~<OS path name>((2 to 510 bytes))

Specifies the absolute path name of the import option file that contains the import options. If this option is omitted, all import options are set to their default values because no import option file is specified.

For details about the import options, see 17.2.2 Format of import options.

Make sure that the import option file name differs from all other file names. If the import option file has the same name as another file, data in the file might be lost. For details about the files whose names must differ from the import option file name, see 17.11 Notes.

- `-e`

Normally, you will omit this option.

Specifies that data import processing is to be canceled if a logical error is detected in the input data.

If you specify this option and a logical error is detected in the input data, the transaction is rolled back (no data is imported). In this case, the return code is `8`.

If you omit this option and a logical error is detected in the input data, only normal data is imported. The data in which the logical error occurred is not imported. In this case, the return code is `4`.

- --force

  Specifies that the adbimport command is to be executed forcibly after all data has been deleted from the table. Consider using this option after the adbimport command was canceled and you want to delete all table data and re-execute data import processing.

  When you re-execute the adbimport command without specifying this option, any data stored in the table during previous data import processing is retained and only the remaining data is imported during re-execution.

  On the other hand, if you re-execute the adbimport command with this option specified, all data stored in the table is deleted and then data import processing is performed from the beginning.

  Important

  - When you specify this option, make sure that you also specify the -d option (creation mode). If the -d option is omitted, the adbimport command will result in an error.
  - When you re-execute the adbimport command with this option and with the -d option specified, all data stored in the table is deleted. Therefore, do not specify this option if you want to release a table from non-updatable status. For details about how to release a table from non-updatable status, see *Steps to take when a base table becomes non-updatable* in *Problems related to base tables* in *Troubleshooting* in the *HADB Setup and Operation Guide*.

  To execute the adbimport command with this option specified:

  1. Execute the adbexport command to output to a file the data stored in the table.

     As needed, output data from each chunk separately.
  2. Use the OS's rm command or a similar method to delete the temporary work files that were created the last time the adbimport command was executed.

     Delete the files under the directories listed below. If there are any files for which you do not have write or read privileges, you might not be able to execute the adbimport command.

     • If the -w option was specified during execution of the previous command that was canceled: The temporary work file storage directories specified in the -w option

     • If the -w option was omitted during execution of the previous command that was canceled: The $DBDIR/ADBWORK directory
  3. Using the files that were output in step 1 as input data files, execute the adbimport command with the -d and --force options specified.

- -b [Multi-chunk table]

  Specifies that data is to be added by background import. When this option is specified, the command adds the data to a new chunk.

  - If only this option is specified, chunks in normal status are created and then data is added.
  - If this option is specified together with the --status wait option, chunks in wait status are created and then data is added.

  Specify this option for a multi-chunk table. An error results if this option is specified for a single-chunk table.

  The first time you import data to a multi-chunk table, do not specify this option. If this option is specified at such a time, the data will be placed in a new chunk, not the one created during table definition, thereby wasting a chunk.

  For details about the chunk to which data is added, see (2) Relationship between data import patterns and options to be specified (types of chunks to which data is to be added).

- `--status wait` [Multi-chunk table]

  Specifies that chunks in wait status are to be created and then data is to be added by background import. Data added to chunks in wait status is excluded from processing by data manipulation SQL statements (note that `PURGE CHUNK` and `TRUNCATE TABLE` statements can be executed).

  Specify this option for a multi-chunk table. An error results if this option is specified for a single-chunk table.

  > **Important**
  >
  > When you specify this option, make sure that you also specify the `-b` option. If the `-b` option is omitted, the `adbimport` command will result in an error.

- `-m` *comment-to-be-set-in-chunk* [Multi-chunk table]

  ~<character string>((1 to 1,024 bytes))

  Specifies a comment that is to be set in the chunk to which data is to be added.

  Specify this option for a multi-chunk table. An error results if this option is specified for a single-chunk table.

  A comment that has already been set in the chunk to which data is to be added will be overwritten.

  For details about the chunk to which data is added, see (2) Relationship between data import patterns and options to be specified (types of chunks to which data is to be added).

  The following rules apply to specifying this information:

  - The specified comment is case sensitive.
  - If the comment contains a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), specify the escape character (`\`) immediately before that character. Alternatively, enclose the entire comment in single quotation marks (`'`).

    Example 1: `-m abc\|def`

    Example 2: `-m 'abc|def'`

    In both examples, `abc|def` is set as the comment.
  - If a character string is enclosed in double quotation marks (`"`) and then the entire string is enclosed in single quotation marks (`'`), both the character string and the double quotation marks are treated as a comment.

    Example: `-m '"abcdef"'`

    In this example, `"abcdef"` is set as a comment.
  - The character encoding specified in the HADB server's `ADBLANG` environment variable is used for the character string set as a comment.

- `-f {csv|fix}`

  Specifies the format of the input data files. When this option is omitted, `csv` is assumed.

  `csv`: CSV format

  `fix`: Fixed-length format

  For details about the CSV format, see 17.5 Format of input data files (CSV format specifications).

  For details about the fixed-length format, see 17.6 Format of input data files (fixed-length format specifications).

  > **Important**
  >
  > When the input data files are in the fixed-length format (when you specify `fix` in this option), make sure that you specify the `-r` option. If the `-r` option is omitted, the `adbimport` command will result in an error.

- `-r` *column-structure-information-file-name*

  ~<OS path name>((2 to 510 bytes))

  Specifies the absolute path of the column structure information file in which the column structure information options are specified

  A column structure information file is used to specify the relationship between the table to which import processing is being applied and the input data. For details about the specification format of the column structure information options, see 17.2.3 Format of column structure information options.

  Whether this option needs to be specified (whether a column structure information file is needed) depends on the file format of the input data files, as well as the sort order of field data and the number of field data items in the input data files. The following table shows the relationship between the input data files and the `-r` option.

Table 17-3: Relationship between input data files and the -r option

No.	Input data file format	Field data in the input data files	-r option specification
1	CSV format	Satisfies the following two conditions: • The field data sort order is the same as the column structure order in the table to which import processing is applied. • The number of field data items is the same as the number of column definitions in the table to which import processing is applied.	O
2		The field data sort order is different from the column structure order in the table to which import processing is applied.	R
3		The number of field data items is different from the number of column definitions in the table to which import processing is applied.	R
4		The field data contains binary data in binary format.	R
5	Fixed-length format	Satisfies the following two conditions: • The field data sort order is the same as the column structure order in the table to which import processing is applied. • The number of field data items is the same as the number of column definitions in the table to which import processing is applied.	R
6		The field data sort order is different from the column structure order in the table to which import processing is applied.	R
7		The number of field data items is different from the number of column definitions in the table to which import processing is applied.	R
8		The field data contains binary data in binary format.	R

Legend:

R: You must specify the `-r` option. (A column structure information file is required for importing data.)

O: Specification of the `-r` option is optional. (Data can be imported without a column structure information file.)

Make sure that the column structure information file name is unique among all other file names. If the specified column structure information file name duplicates an existing file name, data in the column structure information file might be lost. For details about the files whose names must differ from the name of the column structure information file, see 17.11 Notes.

Important

When you omit specification of this option, you need to specify for the input data file a file that satisfies the following conditions:

- A file in CSV format
- A file in which the field data sort order is the same as the column structure order in the table to which import processing is applied
- A file in which the number of field data items is the same as the number of column definitions in the table to which import processing is applied

- *table-to-be-processed*

  Specifies the table into which data is to be imported.

  The following rules apply to specifying this option:

  - Specify in the format *schema-name*`.`*table-identifier*.

    If the user owns the table, the schema name can be omitted. If another HADB user owns the table, the schema name must be specified.
  - For details about the specification rules that apply when a schema name or table identifier includes any lowercase letter or backslash (\), see 1.4.3 Table name specification rules.
  - The user can specify only a table for which the user has the `IMPORT TABLE` privilege.
  - A viewed table cannot be specified.

- *input-data-path-file-name*

  ~<OS path name>((2 to 510 bytes))

  Specifies the absolute path name of the input data path file. If a path name contains a space, the entire path name must be enclosed in single quotation marks (`'`).

  For the input data path file, specify the storage location of the input data files, expressed as an absolute path consisting of 2 to 510 bytes. The characters permitted in path names are the same as for the input data path file name.

  If frequent input operations are performed on the input data files stored under a single directory, performance might be adversely affected. We recommend that you specify directories on multiple disks as input data file storage locations to distribute the workload of input operations among disks.

  The following shows an example of an input data path file.

  Example of input data path file

  ```
  /mnt/diska/wwwww/imp1.csv
  /mnt/diskb/xxxxx/imp2.csv
  /mnt/diskc/yyyyy/imp3.csv
  /mnt/diskd/zzzzz/imp4.csv
  ```

  This example specifies the storage locations of four input data files.

  Make sure that the input data path file name and the input data file storage locations specified in the file are unique from all other file names. If the input data path file or any of the input data file storage locations in the file has the same name as another file, data in the input data path file or input data files might become corrupted. For details about the files whose names must differ from the names of the input data path file and input data file storage locations, see 17.11 Notes.

17.2.2 Format of import options

(1) Specification format

```
Options related to outputting logical error data
  [set adb_import_errdata_file_name = logical-error-data-file-name]
```

```
  [set adb_import_errdata_num = maximum-number-of-output-logical-error-data-items]
  [set adb_import_errmsg_lv = {0|1}]

Options related to performance
  [set adb_import_rthd_num = number-of-processing-real-threads-to-be-used-by-adbimpor
t-command]
  [set adb_import_read_size = buffer-size]
  [set adb_import_buff_blk_num = number-of-buffer-blocks]
  [set adb_import_sort_buff_size = sort-buffer-size]
  [set adb_import_dividx_rd_buff_size = B-tree-and-text-index-input-buffer-size]
  [set adb_import_dividx_wt_buff_size = B-tree-and-text-index-output-buffer-size]
  [set adb_import_txt_buff_size = text-index-creation-buffer-size]
  [set adb_import_decompress_buff_size = buffer-size-for-decompressing-input-data-fil
es-compressed-in-GZIP-format]

Options related to input data
  [set adb_import_null_string = {DEFAULT|NULL}]
```

The specification rules for these import options are the same as for the server definitions. For details about the syntax rules for the server definition, see the topic *Syntax rules for the server definition* in the *HADB Setup and Operation Guide*.

(2) Explanation of specification format

(a) Options related to output of logical error data

- adb_import_errdata_file_name = *logical-error-data-file-name*

  ~<OS path name>((2 to 510 bytes))

  Specifies the absolute path name of the logical error data file. Any row data in the input data that contains a logical error is output to the logical error data file.

  If this option is omitted, row data containing logical errors is not output to a file.

  Make sure that the logical error data file name differs from all other file names. If the logical error data file has the same name as another file, data in the file might be lost. For details about the files whose names must differ from the logical error data file name, see 17.11 Notes.

- adb_import_errdata_num = *maximum-number-of-logical-error-data-items-that-can-be-output*

  ~<integer>((1 to 4,294,967,295))<<100>>

  Specifies the maximum number of logical error data items that can be output to the logical error data file. For example, if the value 200 is specified, the first 200 logical error data items are output to the file, but no subsequent logical error data items are output.

  Note that the command continues import processing even after the number of logical error data items exceeds this option's value.

- adb_import_errmsg_lv = {0|1}

  Specifies whether messages indicating the causes of the logical errors are to be output when logical errors occur:

  0: Output messages.

  1: Do not output messages.

  If this option is omitted, 0 is assumed, in which case messages indicating the causes of logical errors are output.

  One message is output for each logical error. This means that many messages might be output if a lot of logical errors are detected.

Use the guidelines provided in the following table to determine how to specify the foregoing options related to output of logical error data.

Table 17-4: Guidelines for specifying import options

No.	Case	Related import options	Description
1	To correct logical error data and then import the corrected data	• adb_import_errdata_file_name • adb_import_errdata_num	Specify the options shown at the left to output logical error data to the logical error data file.
2	Logical error data is not required (there is no need to correct and then import the data)	• Omit specification of adb_import_errdata_file_name, thus suppressing output of logical error data. • Specify adb_import_errmsg_lv = 1, thus suppressing output of messages indicating the causes of logical errors.	Specify the options as shown at the left to suppress output of information about logical error data. If there are many logical error data items, a large amount of information related to logical error data would be output, thereby adversely affecting performance. In such a case, specify the import options as shown at the left to suppress output of information related to logical error data.
3	To cancel import processing when logical error data is detected	There is no need to specify import options.	Execute the adbimport command with the -e option specified. If a logical error data is detected, importing is canceled and the transaction is rolled back (no data is imported). In this case, the return code is 8. For details about the -e option, see the explanation of the -e option in (3) Explanation of options in 17.2.1 Explanation of the specification format and options.

(b) Options related to performance

- adb_import_rthd_num = *number-of-processing-real-threads-to-be-used-by-adbimport-command*

  ~<integer>((2 to 4,096))

  Specifies the number of processing real threads to be used for executing the adbimport command.

  Specification of this option is recommended. The adbimport command uses the number of processing real threads specified by this option. Therefore, specify an appropriate number of processing real threads for use by the adbimport command. For details about the processing of the adbimport command that uses processing real threads and the number of processing real threads to be used for each step, see 17.1.3 Relationships among threads used by the adbimport command.

  If this option is omitted, the number of processing real threads to be used for executing the adbimport command is determined by the adb_sql_exe_max_rthd_num operand in the server definition. The following table shows the relationship between the value of this option or the adb_sql_exe_max_rthd_num operand in the server definition and the number of processing real threads used for executing the command.

Table 17-5: Number of processing real threads used for executing the command

No.	Value in this option or in the server definition		Number of processing real threads used for executing the command
1	When this option is omitted	If the adb_sql_exe_max_rthd_num operand in the server definition is not specified	The default value of the adb_sql_exe_max_rthd_num operand in the server definition is used.
2		If 0 or 1 is specified in the adb_sql_exe_max_rthd_num operand in the server definition	--
3		If 2 or a greater value is specified for the adb_sql_exe_max_rthd_num operand in the server definition	The value specified for the adb_sql_exe_max_rthd_num operand in the server definition is used.
4	If 2 or a greater value is specified in this option		The value specified in this option is used.

Legend:

--: The `adbimport` command results in an error.

Note

For details about the `adb_sql_exe_max_rthd_num` operand in the server definition, see *Operands related to performance (set format)* in the *HADB Setup and Operation Guide*.

> **Important**
>
> When you perform background import, the data storage efficiency might vary depending on the value specified for `adb_import_rthd_num`. For details about how to obtain a value to specify for `adb_import_rthd_num` that will improve the data storage efficiency, see *Storing data in a multi-chunk table (background import)* in the *HADB Setup and Operation Guide*.

- `adb_import_read_size` = *buffer-size*

  ~<integer>((32 to 51,200)) <<1.024>> (kilobytes)

  Specifies the size (in kilobytes) of the buffer to be used for reading the input data files. As a guideline, specify the following value:

  (*Number of records in the data to be imported* × *length of input data per row*) ÷ 10,000

- `adb_import_buff_blk_num` = *number-of-buffer-blocks*

  ~<integer>((5 to 4,096))<<64>>

  Specifies the number of buffer blocks to be used for data import processing.

  In general, you can use the default value, but a larger value might improve the performance of data import processing.

- `adb_import_sort_buff_size` = *sort-buffer-size*

  ~<integer>((1 to 2,048))<<256>> (megabytes)

  Consider specifying this option when B-tree or text indexes are defined for the table subject to import processing. The size (in megabytes) of the sort buffer to be used for building B-tree and text indexes is specified in this option.

  Specify a large value if there is sufficient memory. Better performance can be expected as this option's value becomes larger.

- `adb_import_dividx_rd_buff_size` = *B-tree-and-text-index-input-buffer-size*

  ~<integer>((32 to 51,200))<<1,024>> (kilobytes)

  Consider specifying this option when B-tree and text indexes are defined for the table subject to import processing.

  The size (in kilobytes) of the input buffer to be used for building B-tree and text indexes is specified in this option. Specify a large value if there is sufficient memory. Better performance can be expected as this option's value becomes larger.

- `adb_import_dividx_wt_buff_size` = *B-tree-and-text-index-output-buffer-size*

  ~<integer>((32 to 51,200))<<1,024>> (kilobytes)

  Consider specifying this option when B-tree and text indexes are defined for the table subject to import processing.

  The size (in kilobytes) of the output buffer to be used for building B-tree and text indexes is specified in this option. Specify a large value if there is sufficient memory. Better performance can be expected as this option's value becomes larger.

- `adb_import_txt_buff_size` = *text-index-creation-buffer-size*

  ~<integer>((1 to 2,048))<<256>> (megabytes)

  Consider specifying this option when text indexes are defined for the table subject to import processing.

  The size (in megabytes) of the buffer to be used for building text indexes is specified in this option.

Increasing the value specified for this option often improves performance. However, the amount of buffer memory specified by this option will be used for all processing real threads on which text indexes will be built. Therefore, be careful when you increase the value specified for this option.

- `adb_import_decompress_buff_size` = *buffer-size-for-decompressing-input-data-files-compressed-in-GZIP-format*

  ~<integer>((32 to 51,200))<<1,024>> (kilobytes)

  Specify the size (in kilobytes) of the buffer used for decompressing input data files compressed in GZIP format.

  Use the value specified in the import option `adb_import_read_size` as a guideline for specifying this option.

  When data of input data files compressed in GZIP format is imported, only the buffer amount specified in this option is used for decompressing the compressed data.

(c) Options related to input data

- `adb_import_null_string` = `{DEFAULT|NULL}`

  Specifies whether the default value for a column or the null value is to be stored when the input data to be stored in a column for which the `DEFAULT` clause was specified during table definition is the empty string.

  DEFAULT

  Store the default value for the column. For details about the default values for columns, see the following topics:

  - *Specification format and rules for the DEFAULT clause* in *DEFAULT clause* in *Constituent Elements* in the manual *HADB SQL Reference*
  - *Setting a default value for a column (DEFAULT clause)* in *Designing a table* in *Designing a Database* in the *HADB Setup and Operation Guide*

  NULL

  Store the null value.

  If this option is omitted, `DEFAULT` is assumed; that is, when the input data is the empty string, the default value for the column will be stored.

  For details about how to specify empty strings in the input data, see 17.5.3 Specification format of field data (CSV format) or 17.6.3 Specification format of field data (fixed-length format).

17.2.3 Format of column structure information options

This subsection explains the column structure information options that are specified in the column structure information file.

To specify a column structure information option (when a column structure information file is needed), specify a column structure information file name in the `-r` option of the `adbimport` command. For details about the `-r` option, see -r.

(1) Specification format

If the file format of the input data files is CSV:

```
{{adbcolumninfo
    -n column-name
    [-r {field-data-number|empty_string}]
    [-b {hex|bin}]
}}
```

If the file format of the input data files is fixed-length:

```
set adb_import_input_record_size = input-record-length

{{adbcolumninfo
    -n column-name
    -p start-position,length
    [-c empty-string-comparison-value]
    [-b {hex|bin}]
}}
```

The specification rules for the above are the same as for the server definition. For details about the server definition specification rules, see the topic *Syntax rules for the server definition* in the *HADB Setup and Operation Guide*.

(2) Relationship between column structure information options and the file format of the input data files

The column structure information options to be specified in the column structure information file vary according to the file format of the input data files. The following table shows the relationship between column structure information options and the file format of the input data files.

For details about the individual options and operands, see (3) Explanation of specification format.

Table 17-6: Relationship between column structure information option and the file format of the input data files

No.	Option name or operand name		File format of the input data files	
			CSV	Fixed-length
1	`adb_import_input_record_size` operand		N	R
2	`adbcolumninfo` operand		O	R
3	When the `adbcolumninfo` operand is specified	`-n` option	R	R
4		`-p` option	N	R
5		`-c` option	N	O
6		`-r` option	O	N
7		`-b` option	O	O

Legend:

R: Specification of the option or operand is required.

O: Specification of the option or operand is optional.

N: The option or operand cannot be specified.

(3) Explanation of specification format

(a) Set format

- `adb_import_input_record_size` = *input-record-length*

  ~<integer>((1 to 536,870,912))(bytes)

  You cannot specify this option if the file format of the input data files is CSV.

When the file format of the input data files is fixed-length, this option specifies the record length (data length per row) in the input data files.

If the records contain a line break code, specify the value in bytes including the length of the line break code.

(b) Command format

- `adbcolumninfo`

  Specifies how to handle the input data in units of the columns defined in the table to be processed.

  You can omit this operand if the file format of the input data files is CSV.

  Specify this operand at least once if the file format of the input data files is fixed-length.

  Make sure that the number of times this operand is specified does not exceed the number of columns defined for the table to be processed.

  There is no need to use this operand to specify all columns defined in the table to be processed. For any columns not specified by this operand, the following data is assumed as the input data:

  - If the file format of the input data files is CSV:

    Data with a field data number that is the same as the column definition order is assumed.

    A field data number is a number that is assigned according to the order in which field data items within a single record are arranged. The first field data item in the input record is assigned number 1, and the remaining field data items are assigned successive numbers. Even if a field data item is an empty string, it is counted as one field data item.

  - If the file format of the input data files is fixed-length:

    An empty string is assumed.

  `-n` *column-name*

  ~<character string>((1 to 100 bytes))

  Specifies a column name defined for the table to be processed.

  Note the following:

  - If a column name contains a space, enclose the column name between double-quotation marks (`"`).

    Example: `"CΔ1"`

    Legend: Δ: Space

  - If a column name contains any lowercase letters, enclose the column name between sets of a backslash and a double-quotation mark (`\"`*column-name*`\"`). If the column name is not enclosed in sets of `\"`, all characters in the column name will be treated as uppercase letters.

    Example: `\"c1\"`

    If a column name contains both lowercase letters and spaces, enclose the column name between sets of a backslash and a double-quotation mark and then between double-quotation marks (`"`).

    Example: `"\"cΔ1\""`

    Legend: Δ: Space

  `-p` *start-position,length*

  You cannot specify this option if the file format of the input data files is CSV.

  Specification of this option is required if the file format of the input data files is fixed-length.

  - *start-position*

    ~<integer>((1 to 536,870,912))(bytes)

    Specifies the start position of the data that is to be stored in the column specified in the `-n` option. Specify the position relative to the beginning of the input record (where the beginning of the input record is 1).

- *length*

  ~<integer>((1 to 536,870,912))(bytes)

  Specifies the length in the input record of the data that is to be stored in the column specified in the `-n` option.

  The following cannot be specified for *length* in the `-p` option:

  • Any value that is greater than the maximum number of characters in the column with the column name specified in the `-n` option in character format

  • An odd value if the column with the column name specified in the `-n` option contains binary data

  • A value of `7` or smaller if `bin` is specified for the `-b` option

> **Important**
>
> - No error occurs if the range specified in this option overlaps the range of values specified in the `-p` option in any other `adbcolumninfo` operand.
> - When input data is enclosed in enclosing characters, the enclosing characters are also stored as part of the data. If you do not want the enclosing characters to be treated as data to be stored, specify this option so that ranges do not include enclosing characters.

`-c` *empty-string-comparison-value*

~<character string>((1 to 255 bytes))

You cannot specify this option if the file format of the input data files is CSV.

When the file format of the input data files is fixed-length, this option specifies data that is to be treated as an empty string. Any input data that matches this option value is treated as an empty string.

Make sure that the length of this option's value in characters matches the length specified in the `-p` option. If the length specified in the `-p` option is greater than 255 bytes and the first 255 bytes of data from the start position specified in the `-p` option matches the value specified in this option, the data is treated as an empty string.

If the specified value contains any spaces, enclose the value between double-quotation marks (`"`).

`-r` {*field-data-number*`|empty_string`}

- *field-data-number*

  ~<integer>((1 to 536,870,912))

  Specifies the field data number of the input data when the format of input data files is CSV.

  You cannot specify this option if the file format of the input data files is fixed-length.

  A field data number is a number that is assigned according to the order in which field data items within a single record are arranged. The first field data item in the input record is assigned number 1, and the remaining field data items are assigned successive numbers. Even if a field data item is an empty string, it is counted as one field data item.

- `empty_string`

  Specify this option to assume an empty string for the input data.

If this option is omitted, data with a field data number that is the same as the definition order of the column corresponding to the column name specified in the `-n` option is assumed as the input data.

> **Important**
>
> - Even if the field data number specified in this option matches the field data number of the `-r` option of another `adbcolumninfo` operand, no error occurs. The same field data is stored in multiple columns of the table to be processed.

- If you specify a field data number in this option and no applicable field data exists in the input record, an error occurs. If you omit this option and no data that corresponds to the assumed field number exists in the input record, an error occurs.

`-b {hex|bin}`

Specifies the format of the binary data in the input record when a binary type is defined for the column name specified in the `-n` option of the `adbcolumninfo` operand.

- `hex`

  Specifies that binary data in the input record is to be treated as binary data in hexadecimal format.

- `bin`

  Specifies that binary data in the input record is to be treated as binary data in binary format.

If this option is omitted, `hex` is assumed.

The following table shows the relationship between the value to be specified in this option and the data type defined for a column with a column name specified in the `-n` option of the `adbcolumninfo` operand.

Table 17-7: Relationship between the data type defined for a column and the value to be specified in the -b option

No.	Data type defined for a column with a column name specified in the -n option	Value specified in the -b option	Data format of the input data
1	Binary	--	Treated as binary data in hexadecimal format
2		`hex`	Treated as binary data in hexadecimal format
3		`bin`	Treated as binary data in binary format
4	Data type other than binary	--	Treated as the data type defined for the column
5		`hex`	An error occurs (the `KFAA50265-E` message is issued).
6		`bin`	

Legend:

--: Specification of the `-b` option is omitted.

(4) Error resulting from an invalid value in a column structure information option

The following are two possible scenarios in which the `adbimport` command will result in an error caused by a column structure information option value:

- An invalid value is specified for a column structure information option.
- There is a problem with consistency between a column structure information option value and the table to be processed.

In both cases, correct the column structure information option value and then re-execute the `adbimport` command.

(a) When an invalid value is specified for a column structure information option

The table below lists and describes the message IDs that are displayed when an invalid value is specified for a column structure information option and the possible errors. Correct the error and then re-execute the `adbimport` command.

Table 17-8: Message IDs that are displayed when an invalid value is specified for a column structure information option and the possible errors

No.	Message ID	File format of the input data files	Possible errors
1	KFAA50265-E	CSV	• The column name specified in the -n option of the adbcolumninfo operand is duplicated. • The -b option is specified for the adbcolumninfo operand in which empty_string is specified for the -r option.
2		Fixed-length	• The value for *start-position* + *length* - 1 specified in the -p option of the adbcolumninfo operand is greater than the input record length specified in the adb_import_input_record_size operand. • The length specified in the -p option of the adbcolumninfo operand does not match the length specified in the -c option of the adbcolumninfo operand. • The column name specified in the -n option of the adbcolumninfo operand is duplicated.
3	KFAA50266-E	• CSV • Fixed-length	More adbcolumninfo operands are specified than the maximum number of columns in the table. For details about the maximum number of columns in a table, see the topic *Maximum and minimum values related to database* in *Maximum and Minimum Values in HADB* in the *HADB Setup and Operation Guide*.

(b) When there is a problem with consistency between a column structure information option value and the table to be processed

The table below lists and describes the message IDs that are displayed when there is a problem with consistency between a column structure information option value and the table to be processed, and the possible errors. Correct the error and then re-execute the adbimport command.

Table 17-9: Message IDs that are displayed when there is a problem with consistency between a column structure information option value and the table to be processed, and the possible errors

No.	Message ID	File format of the input data files	Possible errors
1	KFAA50265-E	CSV	• The column name specified in the -n option of the adbcolumninfo operand does not exist in the table to be processed. • The data type defined for the column specified in the -b option of the adbcolumninfo operand is not binary. • The NOT NULL constraint is defined for a column for which empty_string is specified for the -r option of the adbcolumninfo operand, and for which the DEFAULT clause is not specified. • If multiple adbcolumninfo operands with the same value are specified for the -r option, the values specified in the individual -b options do not match (if the -r and -b options are omitted, whether there is a match is judged based on the value that is assumed).
2		Fixed-length	• The column name specified in the -n option of the adbcolumninfo operand does not exist in the table to be processed. • The length specified in the -p option of the adbcolumninfo operand is greater than the maximum number of characters that can be expressed

No.	Message ID	File format of the input data files	Possible errors
			in character format for the column specified in the `-n` option of the `adbcolumninfo` operand. • The length specified in the `-p` option of the `adbcolumninfo` operand is less than the number of characters that can be expressed in character format for the column specified in the `-n` option of the `adbcolumninfo` operand. • The data type defined for the column specified in the `-b` option of the `adbcolumninfo` operand is not binary. • If there are multiple `adbcolumninfo` operands with the same value specified for the `-p` option, the values specified in the individual `-b` options do not match (if the`-b` options is omitted, whether there is a match is judged based on the value that is assumed).
3	`KFAA50267-E`	Fixed-length	In the case of a column that is not specified in the `-n` option of the `adbcolumninfo` operand (column in which the input data is to be treated as an empty string), the `NOT NULL` constraint is defined and the `DEFAULT` clause is not specified.

17.2.4 Relationship between the column structure information file and the input data

Depending on the specified content of the column structure information file, you can store only part of the input data in the table to be processed.

This subsection explains the patterns of input data to be stored in the table to be processed, and the handling of input data that is not destined to be stored.

(1) Input data to be stored in the columns of the table to be processed

Depending on the specified content of the column structure information file, the data to be stored in the columns of the table to be processed varies.

The following two tables show, for each input data file, the relationship between the specified content of the column structure information file and the data to be stored.

Table 17-10: Input data (in CSV format) to be stored in the table to be processed

No.	Specification of the column structure information file	Specification of the column structure information option or the value specified			Field data number used	Whether the data indicated by the field data number used exists in the input record	Input data to be stored
		`adbcolumninfo` operand	**-n option**	**-r option**			
1	Specified	Specified	Column name	Field data number	Value specified in the `-r` option	Exists	Data indicated by the field data number used
2						Does not exist	N
3				`empty_string`	--	--	Empty string

No.	Specification of the column structure information file	Specification of the column structure information option or the value specified			Field data number used	Whether the data indicated by the field data number used exists in the input record	Input data to be stored
		`adbcolumninfo` operand	-n option	-r option			
4				Not specified	The same number as the definition order of the column with the column name specified in the `-n` option	Exists	Data indicated by the field data number used
5						Does not exist	N
6		Not specified	--	--	The same number as the definition order of the column for which the `adbcolumninfo` operand was not specified	Exists	Data indicated by the field data number used
7						Does not exist	N
8	Not specified	--	--	--	The same number as the definition order of the column defined in the table to be processed	Exists	Data indicated by the field data number used
9						Does not exist	N

Legend:

N: A logical error occurs and no value is stored.

--: Not applicable

Table 17-11: Input data (in fixed-length format) to be stored in the table to be processed

No.	Specification of the column structure information file	Specification of the column structure information option or the value specified			Input data to be stored
		`adbcolumninfo` operand	-n option	-p option	
1	Specified	Specified	Column name	Input data positional information	Data at the position indicated by the `-p` option
2				Not specified	N
3		Not specified	--	--	Empty string
4	Not specified	--	--	--	N

Legend:

N: Because specification of both the column structure information file and the `-p` option is required, an error occurs.

--: Not applicable

(2) Input data not destined to be stored in the table to be processed

When input data is not destined to be stored in the table to be processed, handling of the applicable data differs depending on the input data file format and whether the column structure information file is specified.

The following table shows handling of input data not destined to be stored in the table to be processed.

Table 17-12: Handling of input data not destined to be stored in the table to be processed

No.	File format of the input data files	Specification of the column structure information file	Handling of input data not destined to be stored
1	CSV	Specified	The applicable data is read but not used during data storage.
2		Not specified	A logical error occurs because the number of columns in the input data is greater than the number of columns in the table to be processed.
3	Fixed-length	Specified	The applicable data is read but not used during data storage.
4		Not specified	An error occurs because specification of the column structure information option is required.

Note:

If an input data logical error occurs, take the necessary corrective action based on the explanation in 17.9.1 How to handle logical errors in input data.

17.2.5 Specification patterns for the column structure information options

This subsection explains specification patterns for the column structure information options for each combination of the contents of input data files and the column definition for the table to be processed.

Note

For specification examples of column structure information options, see 17.3.2 Importing table data in CSV format (when the column structure information file is specified) or 17.3.3 Importing table data in fixed-length format.

(1) When the file format of the input data files is CSV

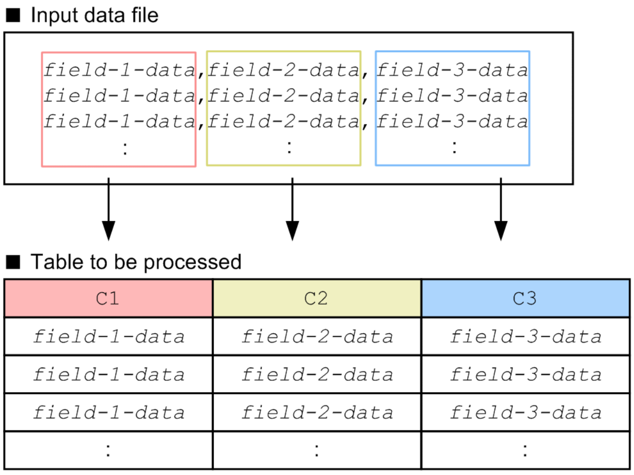
This subsection explains the specification pattern for column structure information options when the file format of the input data files is CSV.

(a) When the sort order of field data items in input data files matches the order of the column definitions for the table to be processed

There is no need to specify a column structure information file if both of the following conditions are satisfied:

- The number of field data items in input data files matches the number of column definitions for the table to be processed.
- The sort order of field data items in input data files matches the order in which the columns appear in the table to be processed.

Figure 17-6: When the sort order of field data items in input data files matches the order of the column definitions for the table to be processed



■ Input data file

```
field-1-data,field-2-data,field-3-data
field-1-data,field-2-data,field-3-data
field-1-data,field-2-data,field-3-data
     :            :            :
```

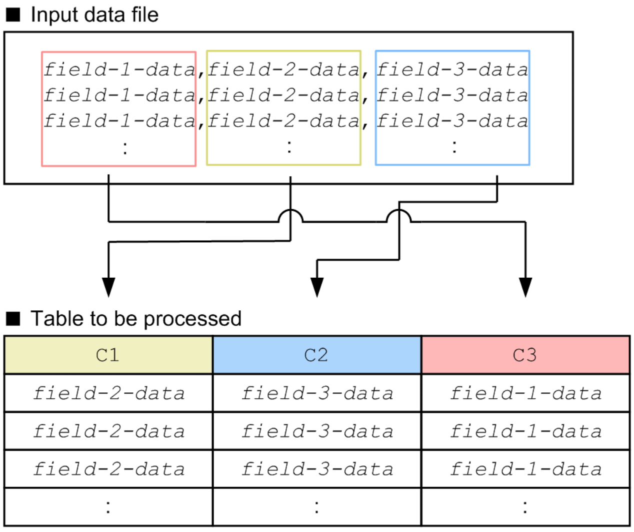
■ Table to be processed

C1	C2	C3
field-1-data	*field-2-data*	*field-3-data*
field-1-data	*field-2-data*	*field-3-data*
field-1-data	*field-2-data*	*field-3-data*
:	:	:

(b) When the sort order of field data items in input data files does not match the order of the column definitions for the table to be processed

The following figure shows the information to specify in the column structure information options when the sort order of field data items in input data files does not match the order of the column definitions for the table to be processed.

Figure 17-7: When the sort order of field data items in input data files does not match the order of the column definitions for the table to be processed



■ Input data file

```
field-1-data,field-2-data,field-3-data
field-1-data,field-2-data,field-3-data
field-1-data,field-2-data,field-3-data
     :            :            :
```

■ Table to be processed

C1	C2	C3
field-2-data	*field-3-data*	*field-1-data*
field-2-data	*field-3-data*	*field-1-data*
field-2-data	*field-3-data*	*field-1-data*
:	:	:

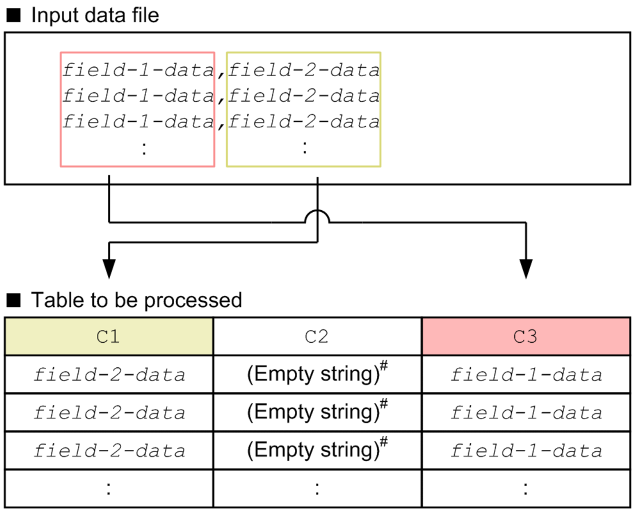
Information to be specified in the column structure information options

```
adbcolumninfo  -n C1  -r 2
adbcolumninfo  -n C2  -r 3
adbcolumninfo  -n C3  -r 1
```

(c) When the number of field data items in input data files is less than the number of column definitions for the table to be processed

The following figure shows the information to specify in the column structure information options when the number of field data items in input data files is less than the number of column definitions for the table to be processed.

Figure 17-8: When the number of field data items in input data files is smaller than the number of column definitions for the table to be processed


■ Input data file

```
field-1-data,field-2-data
field-1-data,field-2-data
field-1-data,field-2-data
     :            :
```

■ Table to be processed

C1	C2	C3
field-2-data	(Empty string)#	*field-1-data*
field-2-data	(Empty string)#	*field-1-data*
field-2-data	(Empty string)#	*field-1-data*
:	:	:

#

For details about the actual value to be stored in the target column when the input data is an empty string, see 17.8 Value that is stored for empty-string input data.

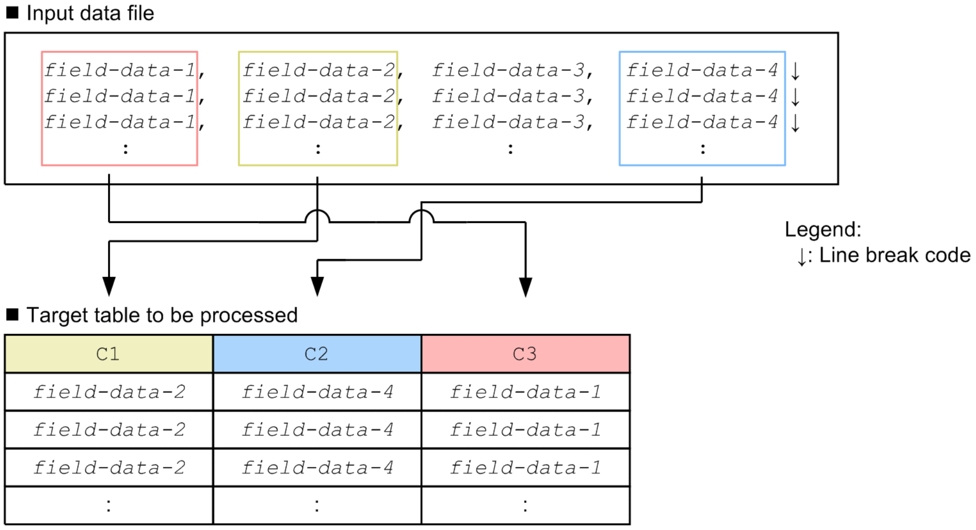
Information to be specified in the column structure information options

```
adbcolumninfo  -n C1  -r 2
adbcolumninfo  -n C2  -r empty_string
adbcolumninfo  -n C3  -r 1
```

(d) When the number of field data items in input data files is greater than the number of column definitions for the table to be processed

The following figure shows the information to specify in the column structure information options when the number of field data items in input data files is greater than the number of column definitions for the table to be processed.

Figure 17-9: When the number of field data items in input data files is greater than the number of column definitions for the table to be processed


■ Input data file

```
field-data-1, field-data-2, field-data-3, field-data-4↓
field-data-1, field-data-2, field-data-3, field-data-4↓
field-data-1, field-data-2, field-data-3, field-data-4↓
     :             :             :             :
```

Legend:
↓: Line break code

■ Target table to be processed

C1	C2	C3
field-data-2	*field-data-4*	*field-data-1*
field-data-2	*field-data-4*	*field-data-1*
field-data-2	*field-data-4*	*field-data-1*
:	:	:

Information to specify in the column structure information options

```
adbcolumninfo  -n C1  -r 2
adbcolumninfo  -n C2  -r 4
adbcolumninfo  -n C3  -r 1
```

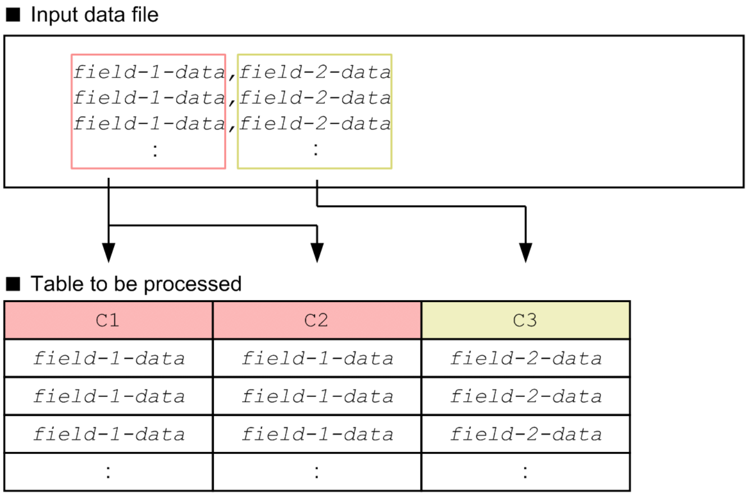
Note:

Field 3 data is not subject to data import.

(e) When a specific field data item in an input data file is stored in multiple columns of the table to be processed

The following figure shows the information to specify in the column structure information options when a specific field data item in the input data file is stored in multiple columns of the table to be processed.

Figure 17-10: When a specific field data item in the input data file is stored in multiple columns of the table to be processed


■ Input data file
field-1-data,field-2-data
field-1-data,field-2-data
field-1-data,field-2-data
:
:
■ Table to be processed

C1	C2	C3
field-1-data	*field-1-data*	*field-2-data*
field-1-data	*field-1-data*	*field-2-data*
field-1-data	*field-1-data*	*field-2-data*
:	:	:

Information to specify in the column structure information options (Example 1)

```
adbcolumninfo  -n C1  -r 1
adbcolumninfo  -n C2  -r 1
adbcolumninfo  -n C3  -r 2
```

Information to specify in the column structure information options (Example 2)

```
adbcolumninfo  -n C1
adbcolumninfo  -n C2  -r 1
adbcolumninfo  -n C3  -r 2
```

Note:

Field 1 data is stored in column `C1` both when `1` is specified in the `-r` option for column `C1`, and when the `-r` option is omitted.

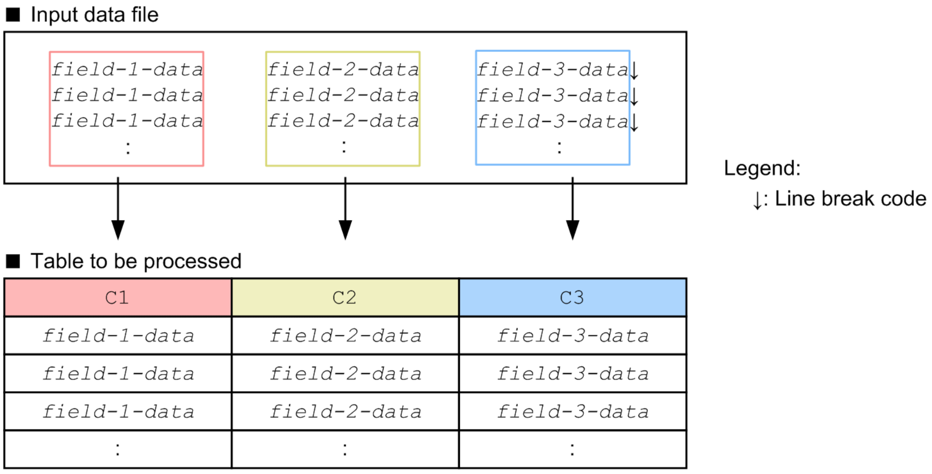
(2) When the file format of the input data files is fixed-length

This subsection explains the specification pattern for column structure information options when the file format of the input data files is fixed-length.

(a) When the sort order of field data in the input data files matches the order of the column definitions for the table to be processed

The following figure shows the information to be specified in the column structure information options when the sort order of field data in the input data files matches the order of the column definitions for the table to be processed.

Figure 17-11: When the sort order of field data in the input data files matches the order of the column definitions for the table to be processed



Information to be specified in the column structure information options

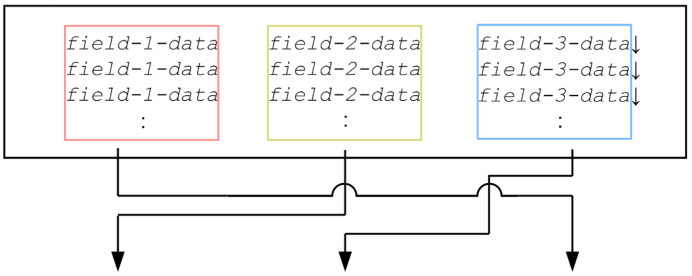
```
set adb_import_input_record_size = sum-of-the-lengths-in-bytes-of-field-1-data-thr
ough-field-3-data-and-of-line-break-codes

adbcolumninfo  -n C1  -p start-position-of-field-1-data,
                         length-of-field-1-data
adbcolumninfo  -n C2  -p start-position-of-field-2-data,
                         length-of-field-2-data
adbcolumninfo  -n C3  -p start-position-of-field-3-data,
                         length-of-field-3-data
```

(b) When the sort order of field data in the input data files does not match the order of the column definitions for the table to be processed

The following figure shows the information to be specified in the column structure information options when the sort order of field data in the input data files does not match the order of the column definitions for the table to be processed.

Figure 17-12: When the sort order of field data in the input data files does not match the order of the column definitions for the table to be processed

■ Input data file



Legend:
↓: Line break code

■ Table to be processed

C1	C2	C3
field-2-data	*field-3-data*	*field-1-data*
field-2-data	*field-3-data*	*field-1-data*
field-2-data	*field-3-data*	*field-1-data*
:	:	:

Information to be specified in the column structure information options

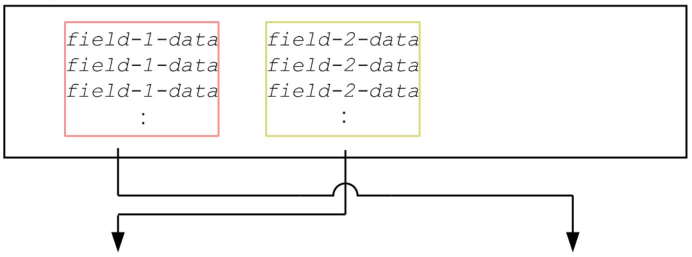
```
set adb_import_input_record_size = sum-of-the-lengths-in-bytes-of-field-1-data-thr
ough-field-3-data-and-of-line-break-codes

adbcolumninfo  -n C1  -p start-position-of-field-2-data,
                         length-of-field-2-data
adbcolumninfo  -n C2  -p start-position-of-field-3-data,
                         length-of-field-3-data
adbcolumninfo  -n C3  -p start-position-of-field-1-data,
                         length-of-field-1-data
```

(c) When the number of field data items in the input data files is fewer than the number of column definitions for the table to be processed

The following figure shows the information to be specified in the column structure information options when the number of field data items in the input data files is fewer than the number of column definitions for the table to be processed.

Figure 17-13: When the number of field data items in the input data files is fewer than the number of column definitions for the table to be processed

■ Input data file



■ Table to be processed

C1	C2	C3
field-2-data	(Empty string)#	*field-1-data*
field-2-data	(Empty string)#	*field-1-data*
field-2-data	(Empty string)#	*field-1-data*
:	:	:

#

For details about the actual value to be stored in the target column when the input data is an empty string, see 17.8 Value that is stored for empty-string input data.

Information to be specified in the column structure information options

```
set adb_import_input_record_size = sum-of-the-lengths-of-field-1-data-and-field-2-
data

adbcolumninfo  -n C1  -p start-position-of-field-2-data,
                         length-of-field-2-data
adbcolumninfo  -n C3  -p start-position-of-field-1-data,
                         length-of-field-1-data
```

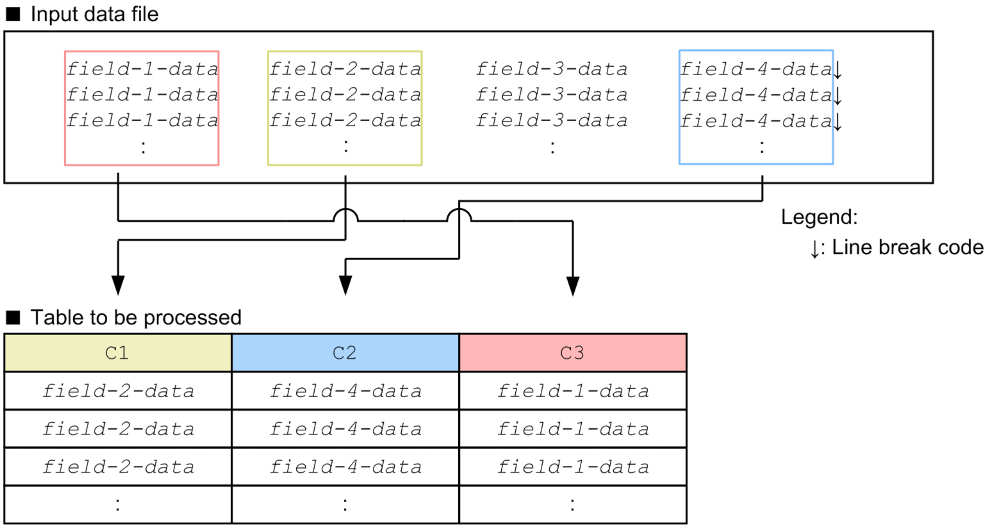
Note:

The `adbcolumninfo` operand is not specified for `C2`.

(d) When the number of field data items in the input data files is greater than the number of column definitions for the table to be processed

The following figure shows the information to be specified in the column structure information options when the number of field data items in the input data files is greater than the number of column definitions for the table to be processed.

Figure 17-14: When the number of field data items in the input data files is greater than the number of column definitions for the table to be processed


■ Input data file
field-1-data field-2-data field-3-data field-4-data↓
field-1-data field-2-data field-3-data field-4-data↓
field-1-data field-2-data field-3-data field-4-data↓
: : : :
Legend:
↓: Line break code
■ Table to be processed

C1	C2	C3
field-2-data	*field-4-data*	*field-1-data*
field-2-data	*field-4-data*	*field-1-data*
field-2-data	*field-4-data*	*field-1-data*
:	:	:

Information to be specified in the column structure information options

```
set adb_import_input_record_size = sum-of-the-lengths-in-bytes-of-field-1-data-thr
ough-field-4-data-and-of-line-break-codes

adbcolumninfo  -n C1  -p start-position-of-field-2-data,
                         length-of-field-2-data
adbcolumninfo  -n C2  -p start-position-of-field-4-data,
                         length-of-field-4-data
adbcolumninfo  -n C3  -p start-position-of-field-1-data,
                         length-of-field-1-data
```

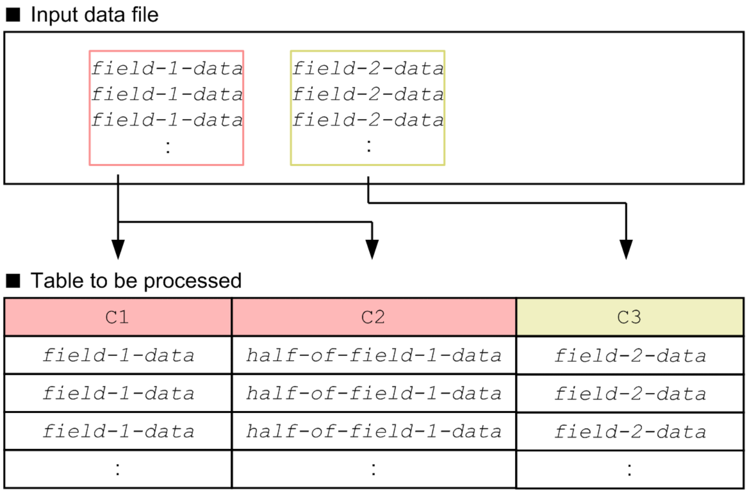
Note:

Field 3 data is not subject to data import.

(e) When specific field data in the input data files is to be stored in multiple columns of the table to be processed

The following figure shows the information to be specified in the column structure information options when specific field data in the input data files is to be stored in multiple columns of the table to be processed.

Figure 17-15: When specific field data in the input data files is to be stored in multiple columns of the table to be processed



C1	C2	C3
field-1-data	*half-of-field-1-data*	*field-2-data*
field-1-data	*half-of-field-1-data*	*field-2-data*
field-1-data	*half-of-field-1-data*	*field-2-data*
:	:	:

Information to be specified in the column structure information options

```
set adb_import_input_record_size = sum-of-the-lengths-of-field-1-data-and-field-2-
data

adbcolumninfo  -n C1  -p start-position-of-field-1-data,
                         length-of-field-1-data
adbcolumninfo  -n C2  -p start-position-of-field-1-data,
                         length-of-half-of-field-1-data
adbcolumninfo  -n C3  -p start-position-of-field-2-data,
                         length-of-field-2-data
```

17.3 Examples

This section explains by way of examples how to use the `adbimport` command to import data into tables.

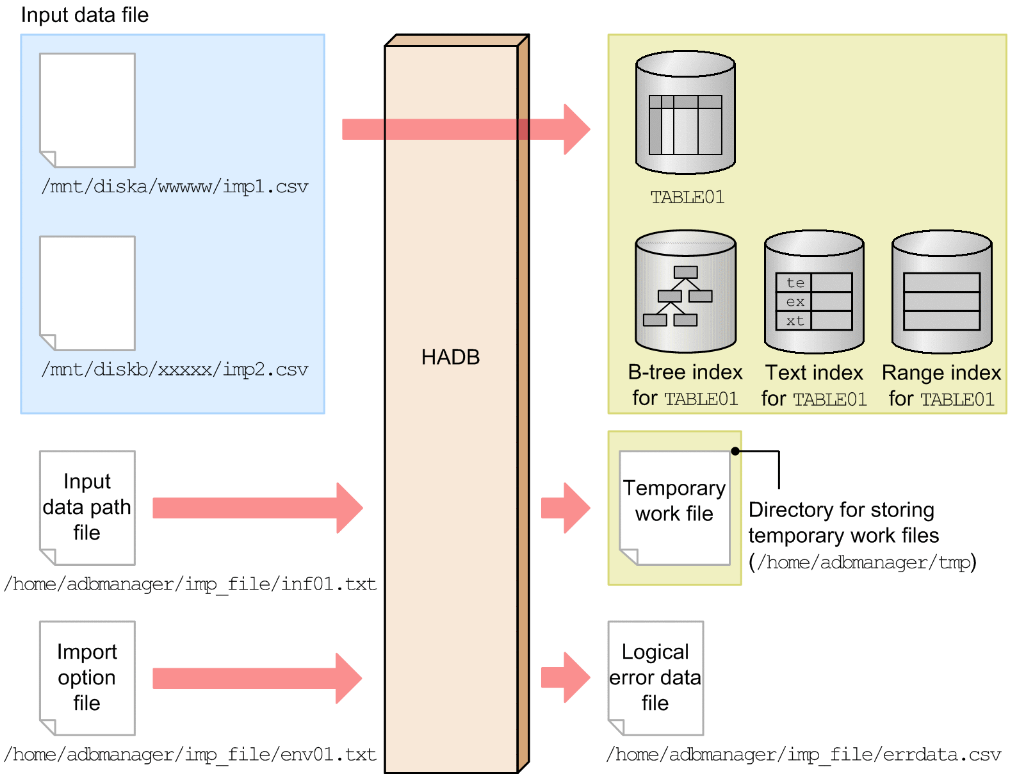
If a logical error occurs in input data as a result of executing the `adbimport` command, take corrective actions based on the explanation in 17.9.1 How to handle logical errors in input data.

If the `adbimport` command terminates abnormally, take appropriate action based on the explanation in 17.10 Handling abnormal termination of the adbimport command.

17.3.1 Importing table data in CSV format

This example imports table data under the following conditions:

- Table data is imported for the first time into table `TABLE01`.
- A B-tree index, text index, or range index is defined for `TABLE01`.
- There are two input data files (in CSV format).
- In the input data files, a single quotation mark (`'`) is used as the enclosing character and a comma (`,`) is used as the delimiter.
- Field data in the input data file satisfies the following two conditions:
  - The sort order of field data items matches the order in which the columns appear in the table subject to import processing.
  - The number of field data items matches the number of column definitions for the table to be processed.
- The input data might contain logical error data.
- Any logical error data that is detected is corrected before the data is imported.


Input data file
/mnt/diska/wwwww/imp1.csv
/mnt/diskb/xxxxx/imp2.csv
HADB
TABLE01
B-tree index for TABLE01
Text index for TABLE01
Range index for TABLE01
te
ex
xt
Input data path file
/home/adbmanager/imp_file/inf01.txt
Import option file
/home/adbmanager/imp_file/env01.txt
Temporary work file
Directory for storing temporary work files (/home/adbmanager/tmp)
Logical error data file
/home/adbmanager/imp_file/errdata.csv

This example uses the following procedure to execute the `adbimport` command.

Procedure

1. Create an input data path file.
2. Create an import option file.
3. Execute the `adbimport` command.

Using examples, the following subsections explain each of these steps.

(1) Creating an input data path file

This example creates an input data path file and specifies in it the paths of the two input data files.

Example of an input data path file

```
/mnt/diska/wwwww/imp1.csv
/mnt/diskb/xxxxx/imp2.csv
```

(2) Creating an import option file

This example creates an import option file that specifies import options.

Example of an import option file

```
set adb_import_errdata_file_name = "/home/adbmanager/imp_file/errdata.csv"   ...1
set adb_import_errdata_num = 500                                              ...2
set adb_import_rthd_num = 2                                                   ...3
set adb_import_sort_buff_size = 2
           :
```

Explanation:

1. Specifies a logical error data file to which logical error data is to be output.
2. Specifies the maximum number of logical error data items that can be output.
3. Specifies other performance-related import options.

(3) Executing the adbimport command

```
adbimport -u ADBUSER01                              ...1
          -p '#HelloHADB_01'                        ...2
          -k "'"                                    ...3
          -s ,                                      ...4
          -g 10                                     ...5
          -w /home/adbmanager/tmp                   ...6
          -z /home/adbmanager/imp_file/env01.txt    ...7
          TABLE01                                   ...8
          /home/adbmanager/imp_file/inf01.txt       ...9
```

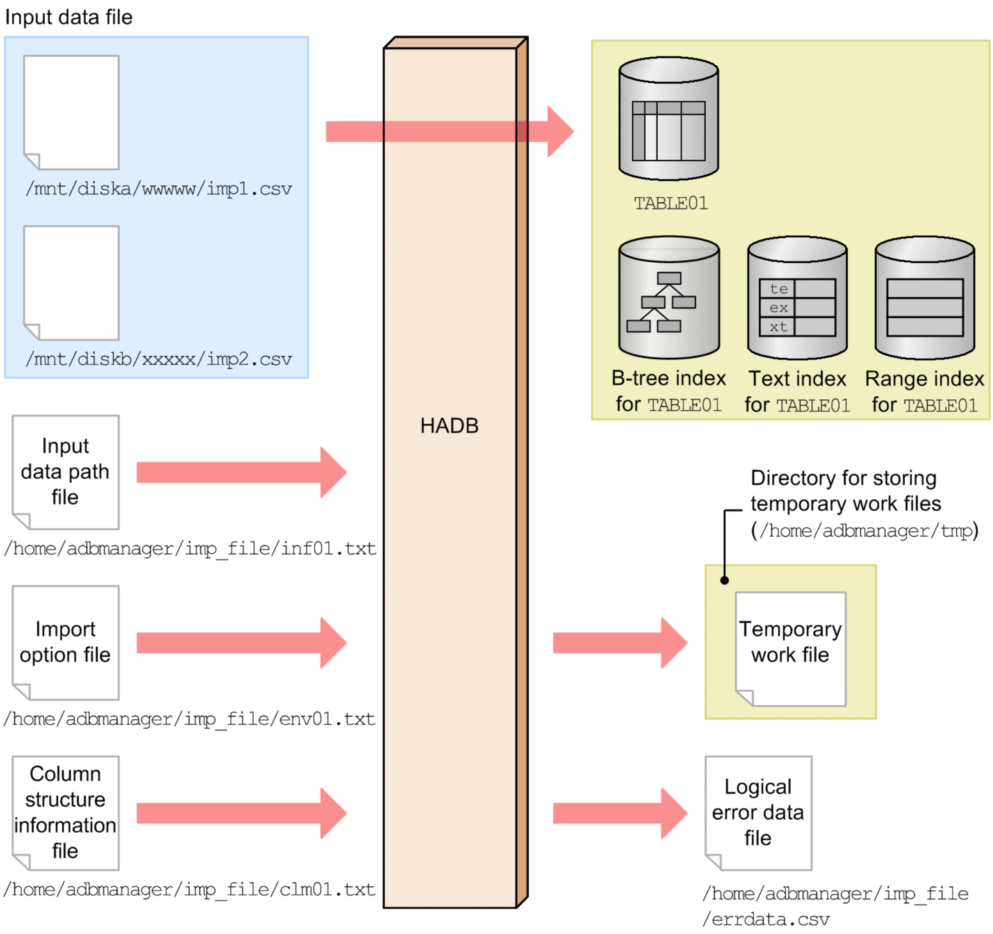
Explanation:

1. Specifies the authorization identifier of the HADB user who executes the `adbimport` command.
2. Specifies the password for the authorization identifier that is specified in the `-u` option.
3. Specifies the enclosing character (single quotation mark) used in the input data files. If the single quotation mark is used as the enclosing character, each single quotation mark must be enclosed in double quotation marks (").
4. Specifies the delimiter (comma) used in the input data files.
5. Specifies that a data import progress message (`KFAA80205-I`) is to be output after every 10,000 rows.
6. Specifies a directory for storing temporary work files that will be created during data import processing.
7. Specifies the absolute path name of the import option file created in (2) Creating an import option file.
8. Specifies the table that is to be the target of import processing (`TABLE01`).
9. Specifies the absolute path name of the input data path file created in (1) Creating an input data path file.

17.3.2 Importing table data in CSV format (when the column structure information file is specified)

This example imports table data under the following conditions:

- Table data is imported for the first time into table `TABLE01`.
- A B-tree index, text index, or range index is defined for `TABLE01`.
- There are two input data files (in CSV format).
- In the input data files, a single quotation mark (`'`) is used as the enclosing character and a comma (`,`) is used as the delimiter.
- The sort order and number of field data items in input data files do not match the order in which the columns appear and the number of column definitions in the table subject to import processing.
- The input data might contain logical error data.
- Any logical error data that is detected is corrected before the data is imported.


Input data file
/mnt/diska/wwwww/imp1.csv
/mnt/diskb/xxxxx/imp2.csv
Input data path file
/home/adbmanager/imp_file/inf01.txt
Import option file
/home/adbmanager/imp_file/env01.txt
Column structure information file
/home/adbmanager/imp_file/clm01.txt
HADB
TABLE01
te
ex
xt
B-tree index for TABLE01
Text index for TABLE01
Range index for TABLE01
Directory for storing temporary work files (/home/adbmanager/tmp)
Temporary work file
Logical error data file
/home/adbmanager/imp_file /errdata.csv

This example uses the following procedure for executing the `adbimport` command.

Procedure

1. Create an input data path file.
2. Create an import option file.
3. Create a column structure information file.
4. Execute the `adbimport` command.

Using examples, the following subsections explain each of these steps.

(1) Creating an input data path file

This example creates an input data path file and specifies in it the paths of the two input data files.

Example of an input data path file

```
/mnt/diska/wwwww/imp1.csv
/mnt/diskb/xxxxx/imp2.csv
```

(2) Creating an import option file

This example creates an import option file that specifies import options.

Example of an import option file

```
set adb_import_errdata_file_name = "/home/adbmanager/imp_file/errdata.csv"   ...1
set adb_import_errdata_num = 500                                             ...2
set adb_import_rthd_num = 2                                                  ...3
set adb_import_sort_buff_size = 2
           :
```

Explanation:

1. Specifies a logical error data file to which logical error data can be output.
2. Specifies the maximum number of logical error data items that can be output.
3. Specifies other performance-related import options.

(3) Creating a column structure information file

This example creates a column structure information file that specifies column structure information options.

Example of column structure information file

```
adbcolumninfo -n C1  -r 2
adbcolumninfo -n C2  -r 3
adbcolumninfo -n C3  -r 1
```

Explanation:

Specifies how to handle input data for each column (C1, C2, and C3) defined for the table to be processed.

For details about the specification patterns for column structure information options for each combination of contents of input data files and the column definition for the table to be processed, see (1) When the file format of the input data files is CSV in 17.2.5 Specification patterns for the column structure information options.

(4) Executing the adbimport command

```
adbimport -u ADBUSER01                                 ...1
          -p '#HelloHADB_01'                           ...2
          -k "'"                                       ...3
          -s ,                                         ...4
          -g 10                                        ...5
          -w /home/adbmanager/tmp                      ...6
          -z /home/adbmanager/imp_file/env01.txt       ...7
          -r /home/adbmanager/imp_file/clm01.txt       ...8
          TABLE01                                      ...9
          /home/adbmanager/imp_file/inf01.txt          ...10
```

Explanation:

1. Specifies the authorization identifier of the HADB user who executes the adbimport command.
2. Specifies the password for the authorization identifier that is specified in the -u option.
3. Specifies the enclosing character (single quotation mark) used in input data files. To specify a single quotation mark for the enclosing character, you must enclose the single quotation mark between double quotation marks (").
4. Specifies the delimiter (comma) used in input data files.
5. Specifies that a data import progress message (KFAA80205-I) is to be output after every 10,000 rows.
6. Specifies a directory for storing temporary work files that will be created during data import processing.
7. Specifies the absolute path name of the import option file created in (2) Creating an import option file.

8. Specifies the absolute path name of the column structure information file created in (3) Creating a column structure information file.

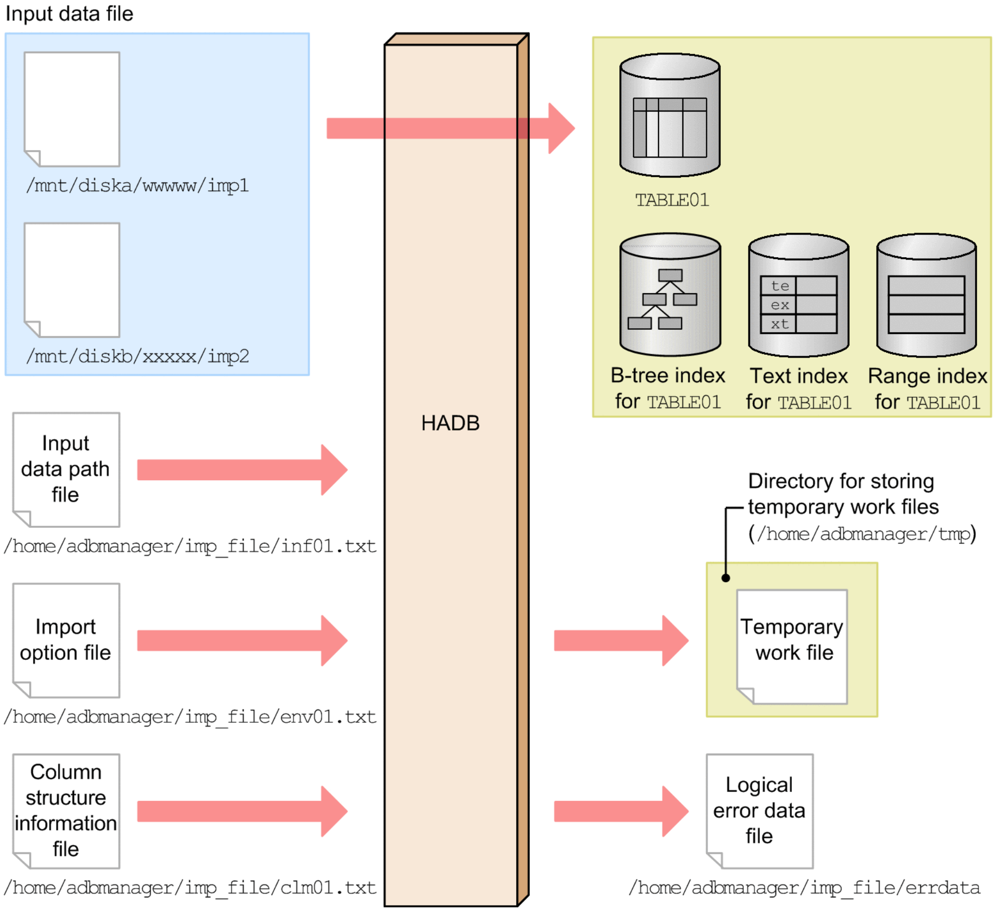
9. Specifies the table subject to import processing (TABLE01).

10. Specifies the absolute path name of the input data path file created in (1) Creating an input data path file.

17.3.3 Importing table data in fixed-length format

This example imports table data under the following conditions:

- Table data is imported for the first time into table TABLE01.
- A B-tree index, text index, or range index is defined for TABLE01.
- There are two input data files (in fixed-length format).
- The input data might contain logical error data.
- Any logical error data that is detected is corrected before the data is imported.


Input data file
/mnt/diska/wwwww/imp1
/mnt/diskb/xxxxx/imp2
Input data path file
/home/adbmanager/imp_file/inf01.txt
Import option file
/home/adbmanager/imp_file/env01.txt
Column structure information file
/home/adbmanager/imp_file/clm01.txt
HADB
TABLE01
te ex xt
B-tree index for TABLE01
Text index for TABLE01
Range index for TABLE01
Directory for storing temporary work files (/home/adbmanager/tmp)
Temporary work file
Logical error data file
/home/adbmanager/imp_file/errdata

This example uses the following procedure to execute the adbimport command.

Procedure

1. Create an input data path file.
2. Create an import option file.
3. Create a column structure information file.

4. Execute the adbimport command.

Using examples, the following subsections explain each of these steps.

(1) Creating the input data path file

This example creates an input data path file and specifies in it the paths of the two input data files.

Example of an input data path file

```
/mnt/diska/wwwww/imp1
/mnt/diskb/xxxxx/imp2
```

(2) Creating an import option file

This example creates an import option file that specifies import options.

Example of an import option file

```
set adb_import_errdata_file_name = "/home/adbmanager/imp_file/errdata"   ...1
set adb_import_errdata_num = 500                                         ...2
set adb_import_rthd_num = 2                                              ...3
set adb_import_sort_buff_size = 2
           :
```

Explanation:

1. Specifies a logical error data file to which logical error data is to be output.
2. Specifies the maximum number of logical error data items that can be output.
3. Specifies other performance-related import options.

(3) Creating a column structure information file

This example creates a column structure information file that specifies column structure information options.

Example of column structure information file

```
set adb_import_input_record_size = 38   ...1

adbcolumninfo -n C1  -p 1,10            ...2
adbcolumninfo -n C2  -p 11,8
adbcolumninfo -n C3  -p 19,20
```

Explanation:

1. Specifies the record length in the input data files.
2. Specifies how to handle input data for each column (C1, C2, and C3) defined for the table to be processed.

   This example assumes that columns C1 (CHAR(10)), C2 (INT), and C3 (CHAR(20)) are defined for TABLE01.

For details about the specification patterns for column structure information options for each combination of contents of input data files and the column definition for the table to be processed, see (2) When the file format of the input data files is fixed-length in 17.2.5 Specification patterns for the column structure information options.

(4) Executing the adbimport command

```
adbimport -u ADBUSER01                                ...1
          -p '#HelloHADB_01'                          ...2
          -g 10                                       ...3
          -w /home/adbmanager/tmp                     ...4
          -z /home/adbmanager/imp_file/env01.txt      ...5
          -f fix                                      ...6
          -r /home/adbmanager/imp_file/clm01.txt      ...7
          TABLE01                                     ...8
          /home/adbmanager/imp_file/inf01.txt         ...9
```

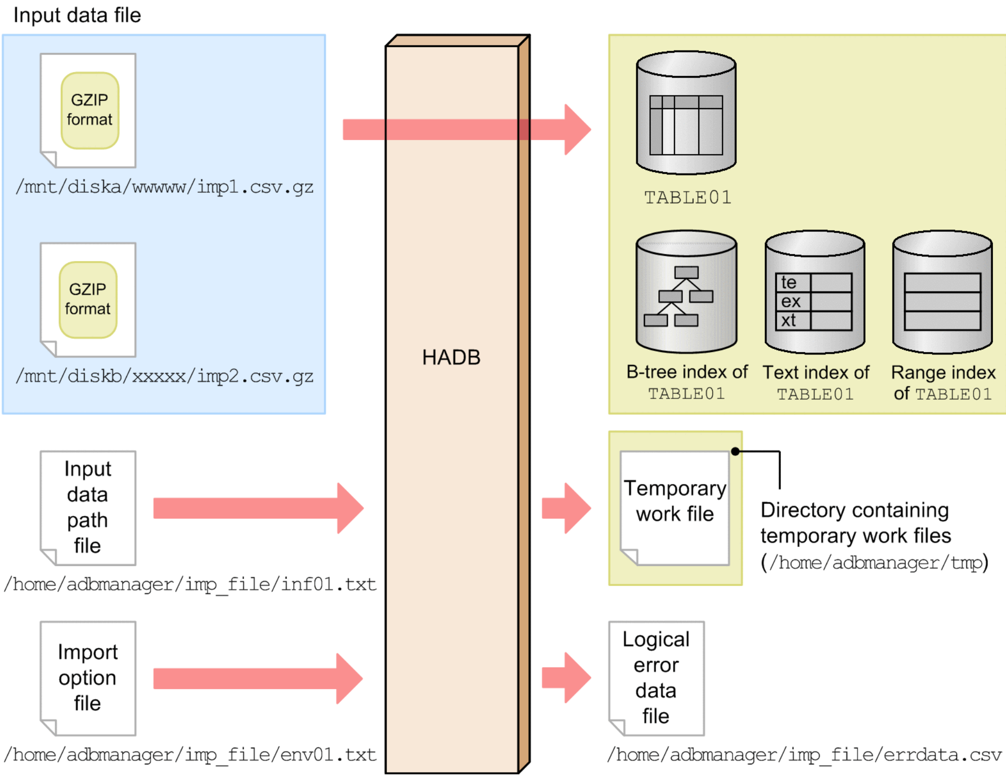
Explanation:

1. Specifies the authorization identifier of the HADB user who executes the `adbimport` command.
2. Specifies the password for the authorization identifier that is specified in the `-u` option.
3. Specifies that a data import progress message (`KFAA80205-I`) is to be output after every 10,000 rows.
4. Specifies a directory for storing temporary work files that will be created during data import processing.
5. Specifies the absolute path name of the import option file created in (2) Creating an import option file.
6. Specifies the fixed-length format for the input data files.
7. Specifies the absolute path name of the column structure information file created in (3) Creating a column structure information file.
8. Specifies the table that is to be the target of import processing (`TABLE01`).
9. Specifies the absolute path name of the input data path file created in (1) Creating the input data path file.

17.3.4 Background importing CSV-format table data compressed in GZIP format

In this example, table data is imported under the following conditions:

- Background importing table data to the table `TABLE01`
- B-tree, text, and range indexes are defined for table `TABLE01`.
- There are two input data files (CSV format).
- The input data files are compressed in GZIP format.
- Within the input data file, single quotation marks (`'`) are used as enclosing characters, and commas (`,`) are used as delimiters.
- Within the input data file, field data fulfills the following two conditions:
  - The field data is placed in the same order as the column structure of the target table to be imported.
  - The number of items of field data is the same as the number of column definitions in the target table to be imported.
- There might be logical errors in the input data.
- If any data contains a logical error, correct the data before performing background import.


Input data file
GZIP format
/mnt/diska/wwwww/imp1.csv.gz
GZIP format
/mnt/diskb/xxxxx/imp2.csv.gz
Input data path file
/home/adbmanager/imp_file/inf01.txt
Import option file
/home/adbmanager/imp_file/env01.txt
HADB
TABLE01
te ex xt
B-tree index of TABLE01
Text index of TABLE01
Range index of TABLE01
Temporary work file
Directory containing temporary work files (/home/adbmanager/tmp)
Logical error data file
/home/adbmanager/imp_file/errdata.csv

This example uses the following procedure to execute the `adbimport` command.

Procedure

1. Create an input data path file.
2. Create an import option file.
3. Execute the `adbimport` command.

Using examples, the following subsections explain each of these steps.

(1) Creating an input data path file

In this example, an input data path file is created, and paths are specified for all the input data files.

Example specification of an input data path file

```
/mnt/diska/wwwww/imp1.csv.gz
/mnt/diskb/xxxxx/imp2.csv.gz
```

(2) Creating an import option file

In this example, an import option file is created, and import options are specified.

Example specification of an import option file

```
set adb_import_errdata_file_name = "/home/adbmanager/imp_file/errdata.csv"   ...1
set adb_import_errdata_num = 500                                             ...2
set adb_import_read_size = 1024                                              ...3
set adb_import_decompress_buff_size = 1024
           :
```

Explanation:

1. The logical error data file that outputs the logical error data is specified.
2. The maximum number of output items of logical error data is specified.
3. Other import options relating to performance are specified.

(3) Executing the adbimport command

```
adbimport -u ADBUSER01                              ...1
          -p '#HelloHADB_01'                        ...2
          -k "'"                                    ...3
          -s ,                                      ...4
          -g 10                                     ...5
          -w /home/adbmanager/tmp                   ...6
          -z /home/adbmanager/imp_file/env01.txt    ...7
          -b                                        ...8
          TABLE01                                   ...9
          /home/adbmanager/imp_file/inf01.txt       ...10
```

Explanation:

1. Specifies the authorization identifier of the HADB user who executes the `adbimport` command.
2. Specifies the password for the authorization identifier that was specified in the `-u` option.
3. Specifies the enclosing characters (single quotation marks) to be used within input data files. If you specify single quotation marks as the enclosing characters, note that single quotation marks must be enclosed in double quotation marks (`"`).
4. Specifies the delimiters (commas) to be used within input data files.
5. Specifies to output progress messages (`KFAA80205-I`) for every 10,000 rows of data import processing.
6. Specifies the directory that stores temporary work files created during data import processing.
7. Specifies the absolute path name of the import option file created in (2) Creating an import option file.
8. Specifies to background import table data.
9. Specifies the target table to be imported (`TABLE01`).
10. Specifies the absolute path name of the input data path file created in (1) Creating an input data path file.

17.4 Return codes

The following table lists and describes the return codes of the `adbimport` command.

Table 17-13: Return codes of the adbimport command

Return code	Description
`0`	The `adbimport` command terminated normally.
`4`	The `adbimport` command terminated normally, but a warning message was issued. Check the displayed message or the message output to a message log file and take appropriate action. If one of the following warning messages was issued, a problem has occurred, such as a constraint violation or the existence of data that was not imported: • The `KFAA61205-W` message was issued. If this message (warning message indicating detection of duplicate key values) was issued, the table and a B-tree index violate the uniqueness constraint (a B-tree index that violates the uniqueness constraint is in the status called uniqueness constraint violation). Locate the duplicate key values and delete the corresponding rows. Then execute the `adbidxrebuild` command to rebuild the B-tree indexes. For details about how to handle such an error, see *Steps to take when the uniqueness constraint is violated (when the KFAA61205-W message is output)* in *Problems related to B-tree indexes* in *Troubleshooting* in the *HADB Setup and Operation Guide*. • A message indicating a logical error data was issued. Check the issued messages for the cause of the logical error data. Note that if you specified `1` for the `adb_import_errmsg_lv` import option, any messages indicating the causes of logical errors will not be issued. The normal data will have been imported. For details about how to handle logical error data, see 17.9.1 How to handle logical errors in input data.
`8`	One of the following errors occurred: • The specified `adbimport` command did not execute because it was invalid. • An error occurred during execution of the `adbimport` command. For the cause of the error and corrective action to take, see the displayed error message or the message that has been output to a message log file. For details, see 17.10 Handling abnormal termination of the adbimport command.

17.5 Format of input data files (CSV format specifications)

This section explains the format of input data files in CSV format.

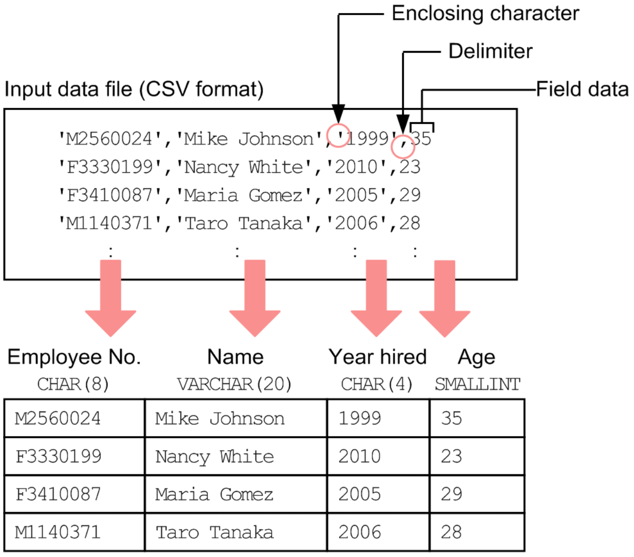
For details about the format of input data files (fixed-length format), see 17.6 Format of input data files (fixed-length format specifications).

If you want to use compressed input data files (CSV format), see 17.7 Compression format of input data files.

17.5.1 CSV format specifications

In CSV format, item data is separated by delimiters. The delimited item data is called *field data*. The following figure shows an example of the CSV format.

Figure 17-16: Example of the CSV format



```
'M2560024','Mike Johnson','1999',35
'F3330199','Nancy White','2010',23
'F3410087','Maria Gomez','2005',29
'M1140371','Taro Tanaka','2006',28
     :            :            :    :
```

Employee No. CHAR(8)	Name VARCHAR(20)	Year hired CHAR(4)	Age SMALLINT
M2560024	Mike Johnson	1999	35
F3330199	Nancy White	2010	23
F3410087	Maria Gomez	2005	29
M1140371	Taro Tanaka	2006	28

Explanation:

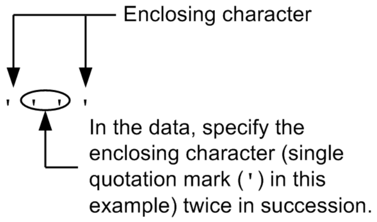
- Each row of data in an input data file corresponds to a row of data in the table.
- The field data items to be entered in the input data files need not be in the same order in which the columns appear in the table.
- Use the character encoding specified in the `ADBLANG` environment variable to create the input data file.
- If field data is enclosed in enclosing characters, specify the enclosing character in the `-k` option of the `adbimport` command.
- Demarcate the field data items with a delimiter. Specify this delimiter in the `-s` option of the `adbimport` command.
- Use the same format as for character string data to represent numeric values, such as age.

17.5.2 Coding rules for input data (CSV format)

- Specify in the files one row of table data per line. Specify a line-ending code at the end of each line. Use `X'0A'` (`LF`), `X'0D0A'`(`CRLF`), or `X'00'` as the line-ending code.

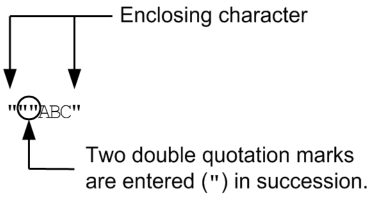
- If data is imported to a row store table, the maximum length of a row of input data is 512 megabytes (536,870,912 bytes). If a row exceeds 512 megabytes, a file input error will occur during execution of the `adbimport` command.
- No EOF control character can be used in an input data file.
- Use a delimiter to demarcate the field data.
- If you use enclosing characters, specify the enclosing character and then the delimiter. Any spaces between a delimiter and an enclosing character will be treated as data. Therefore, if there is a space between a delimiter and an enclosing character, the enclosing character might result in a specification error or might be stored as part of the data.
- To specify the enclosing character within the data, specify the enclosing character twice in succession in the data.

  Example: The single quotation mark (`'`) is used as the enclosing character and as a character in the data.


Enclosing character
' '' '
In the data, specify the enclosing character (single quotation mark (') in this example) twice in succession.

- If data begins with a character that is identical to the enclosing character, specify the enclosing character twice in succession in the data. Then, use the enclosing character to enclose the field data.

  Example: The double quotation mark (`"`) is used as the enclosing character and data is `"ABC`.



  If a single-byte space or tab character is inserted at the beginning of the data, there is no need to use the enclosing character to enclose the field data.

- Everything between enclosing characters is treated as data.
- If field data contains the delimiter character, use the enclosing character to enclose the entire field data. If you do not enclose the entire character string in the field with the enclosing character, the delimiter character in the data will be treated as a delimiter, resulting in an error such as a column count mismatch in the input data.

  The following shows examples where the enclosing character is the double-quotation mark (`"`) and the delimiter is the comma (`,`).

  Example 1: The following example treats *1*, *foo,bar*, and *3* as three columns of field data:

```
1,"foo,bar",3
```

  Example 2: The following example treats *1*, *foo*, *bar*, and *3* as four columns of field data:

```
1,foo,bar,3
```

17.5.3 Specification format of field data (CSV format)

The following table describes the specification format of field data.

Table 17-14: Specification format of field data

No.	Data type		Specification format of field data			
			Format	Specification example	Notes	Empty string specification example#1
1	Numeric data	INTEGER	[{+|−}]*a...a* + and −: Signs *a...a*: Numeric value (*a* is 0 to 9)	• 100 • -123 • 000 • 0657	• A maximum of 20 characters combining signs and numeric values can be specified. • Regardless of the format or the restriction on the number of characters, one or more single-byte spaces or tab characters can be inserted before or after each character. #2	• ...,*,... • ...,"*",... • ...,,... • ...,"",... • ...,"""",... If none is specified in the -k option, examples that use the enclosing character cannot be specified.
2		SMALLINT	[{+|−}]*a...a* + and −: Signs *a...a*: Numeric value (*a* is 0 to 9)	• 100 • -0123 • 0 • +0657	• A maximum of 11 characters combining signs and numeric values can be specified. • Regardless of the format or the restriction on the number of characters, one or more single-byte spaces or tab characters can be inserted before or after each character. #2	Same as No. 1
3		DECIMAL	[{+|−}] {*a...a*[.[*b...b*]]|.*b...b*} + and −: Signs *a...a*: Integer part (*a* is 0 to 9) *b...b*: Decimal part (*b* is 0 to 9)#3	• 100 • -123.00 • Δ.00 • 012. • -1.56 • +.560	• A maximum of 38 characters combining the integer part and the decimal part can be specified (if the precision and scaling are the same in the column at the storage destination, a maximum of 39 characters can be specified only when the integer part (0) is not omitted). • Regardless of the format or the restriction on the number of	Same as No. 1

No.	Data type		Specification format of field data			
			Format	Specification example	Notes	Empty string specification example[#1]
					characters, one or more single-byte spaces or tab characters can be inserted before or after each character. #2	

4		`DOUBLE PRECISION`	`[{+|-}]`
			`{`*a...a*`[.[`*b...b*`]]|.`*b...b*`}[{`*E*`|`*e*`}[[{+|-}]`*c...c*`]]`
			`+` and `-`: Signs
			a...a: Integer part of the mantissa (*a* is `0` to `9`)
			b...b: Decimal part of the mantissa (*b* is `0` to `9`)
			c...c: Exponent part (*c* is `0` to `9`)[#4]
			E and *e*: Floating point numeric literal (literal identifying the exponent in a floating-point number)
				• `100`
				• `-Δ123`
				• `0.Δ`
				• `-1.5600`
				• `.56`
				• `-02.4e+9`
				• `000e`
				• `2.4E+009`
					• A maximum of 509 characters can be specified.[#5]
					• Regardless of the format or the restriction on the number of characters, one or more single-byte spaces or tab characters can be inserted before or after each character. #2
						Same as No. 1

5	Character string data	`CHARACTER`	*a...a* *a...a*: Data containing at least one character	• `abcdefΔΔ` • `ABCDEF` • `Δ`	• A number of characters up to the definition length of the column at the storage definition can be specified. • Except for the beginning of data, single-byte spaces that continue to the end of data can be omitted.[#6]	• `...,,...` • `...,"",...` Single-byte spaces and tab characters cannot be inserted. If `none` is specified in the `-k` option, examples that use the enclosing character cannot be specified.
6		`VARCHAR`	*a...a* *a...a*: Data containing at least one character	• `abcdefΔΔ` • `ABCDEF` • `ΔΔ`	• A number of characters up to the definition length of the column at the storage definition can be specified.	• `...,,...` The enclosing character cannot be specified. Furthermore, single-byte spaces and tab characters cannot be inserted. **■ When data with a length of zero is specified** • `...,"",...` If `none` is specified in the `-k` option, data with a length of zero

No.	Data type		Specification format of field data			
			Format	Specification example	Notes	Empty string specification example#1
						cannot be specified.
7	Datetime data	`DATE`	`{`*YYYY*`-`*MM*`-`*DD* `|`*YYYY*`/`*MM*`/`*DD*`}` *YYYY*: Year (`0001` to `9999`) *MM*: Month (`01` to `12`) *DD*: Date (`01` to the last day of the month)	- `2013-06-10` - `2013/06/10`	- Regardless of the format, one or more single-byte spaces or tab characters can be inserted before or after each character. #2	Same as No. 1
8		`TIME`	*hh*`:`*mm*`:`*ss*`[.[`*nn...n*`]]` *hh*: Hour (`00` to `23`) *mm*: Minute (`00` to `59`) *ss*: Second (`00` to `59`) *nn...n*: (*n* is `0` to `9`)	- `11:03:58` - `11:03:58.` - `11:03:58Δ.1234`	- A maximum of 12 characters can be specified for the fraction of a second (*nn...n*).#7 - Regardless of the format or the restriction on the number of characters, one or more single-byte spaces or tab characters can be inserted before or after each character. #2	Same as No. 1
9		`TIMESTAMP`	`{`*YYYY*`-`*MM*`-`*DD* `|`*YYYY*`/`*MM*`/`*DD*`}` Δ*hh*`:`*mm*`:`*ss*`[.[`*nn...n*`]]` *YYYY*: Year (`0001` to `9999`) *MM*: Month (`01` to `12`) *DD*: Date (`01` to the last day of the month) *hh*: Hour (`00` to `23`) *mm*: Minute (`00` to `59`) *ss*: Second (`00` to `59`) *nn...n*: (*n* is `0` to `9`)	- `2013-06-10Δ11:03:58` - `2013-06-10Δ11:03:58Δ.1234`	- A maximum of 12 characters can be specified for the fraction of a second (*nn...n*).#7 - Regardless of the format or the restriction on the number of characters, one or more single-byte spaces or tab characters can be inserted before or after each character. #2	Same as No. 1
10	Binary data	`BINARY`	For hexadecimal strings *a...a* *a*: `0` to `9`, `A` to `F`, or `a` to `f`	- `12340000` - `90ΔAB` - `90abΔCDEF`	- A number of characters up to twice the definition length of the column at the storage definition and that is a multiple of 2 can be specified.#8 - Except for the beginning of data, `00` that continues	Same as No. 1

No.	Data type		Specification format of field data			
			Format	Specification example	Notes	Empty string specification example#1
					to the end of data can be omitted.#9 • Regardless of the format or the restriction on the number of characters, one or more single-byte spaces or tab characters can be inserted before or after each character. #2	
11			For binary strings *a*...*a* *a*: `0` or `1`	• `01010101` • `0101Δ0101`	• A number of characters up to eight times the definition length of the column at the storage definition and that is a multiple of 8 can be specified.#8 • Except for the beginning of data, `00000000` that continues to the end of data can be omitted.#9 • Regardless of the format or the restriction on the number of characters, one or more single-byte spaces or tab characters can be inserted before or after each character. #2	Same as No. 1
12		`VARBINARY`	For hexadecimal strings *a*...*a* *a*: `0` to `9`, `A` to `F`, or `a` to `f`	• `12340000` • `90ΔAB` • `90abΔCDEF`	• A number of characters up to twice the definition length of the column at the storage definition and that is a multiple of 2 can be specified.#8 • Regardless of the format or the restriction on the number of characters, one or more single-byte spaces or tab characters can be	• `...,*,...` • `...,"*",...` • `...,,...` If `none` is specified in the `-k` option, examples that use the enclosing character cannot be specified. **■ When data with a length of zero is specified** • `...,"",...`

No.	Data type		Specification format of field data			
			Format	Specification example	Notes	Empty string specification example#1
					inserted before or after each character. #2	• ...,"""",... If `none` is specified in the `-k` option, data with a length of zero cannot be specified.
13			For binary strings *a*...*a* *a*: `0` or `1`	• `01010101` • `0101Δ0101`	• A number of characters up to eight times the definition length of the column at the storage definition and that is a multiple of 8 can be specified.#8 • Regardless of the format or the restriction on the number of characters, one or more single-byte spaces or tab characters can be inserted before or after each character. #2	Same as No. 12

Legend:

Δ: One or more single-byte spaces or tab characters

`,`: Delimiter

`"`: Enclosing character

#1

If you specify an empty string when the `DEFAULT` clause is specified for a column at the storage location, the default value for the column is stored. When the `DEFAULT` clause is not specified for a column at the storage location, the null value is stored. If you want to store the null value in a column for which the `DEFAULT` clause has been specified, set the value of `adb_import_null_string` to `NULL`.

For details about the actual values to be stored in the target columns, see 17.8 Value that is stored for empty-string input data.

#2

If one or more single-byte spaces (`0x20`) or tab characters (`0x09`) are entered before or after a character, those single-byte spaces or tab characters are deleted.

Example: Δ`1` Δ`23` ΔΔ `4` ΔΔΔ -> `1234`

If all the data is deleted, an empty string is assumed.

#3

Any low-order digits that do not fit the scaling of the storage destination are discarded.

#4

If the exponent is omitted, +0 is assumed to be the exponent.

#5

Digit cancellation might occur depending on the specified value.

#6

If the input data is shorter than the defined length, trailing spaces are added.

#7

If the fraction of a second (*nn...n*) is shorter than the fractional seconds precision for the table's data type, trailing zeros are added.

If the fraction of a second (*nn...n*) exceeds the fractional seconds precision for the table's data type, the input data is truncated.

#8

A logical error results if the number of characters in the hexadecimal string is not a multiple of 2.

A logical error results if the number of characters in the binary string is not a multiple of 8.

#9

If the input data is shorter than the definition length, `0x00` is stored in the remaining portion.

17.6 Format of input data files (fixed-length format specifications)

This section explains the format of input data files in fixed-length format.

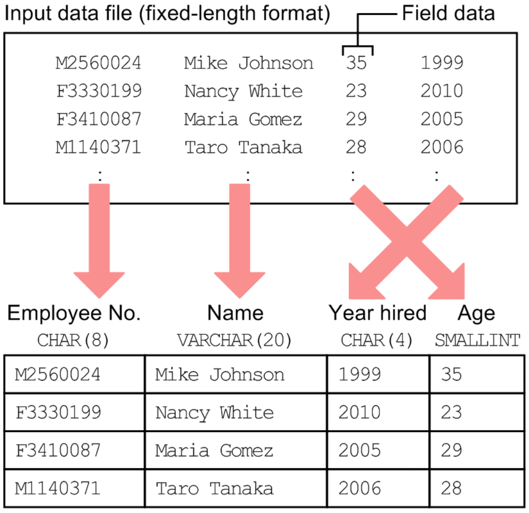
For details about the format of input data files (CSV format), see 17.5 Format of input data files (CSV format specifications).

If you want to use compressed input data files (fixed-length format), see 17.7 Compression format of input data files.

17.6.1 Fixed-length format specifications

In the fixed-length format, record lengths are fixed. The following figure shows an example of the fixed-length format.

Figure 17-17: Example of the fixed-length format



```
M2560024    Mike Johnson   35     1999
F3330199    Nancy White    23     2010
F3410087    Maria Gomez    29     2005
M1140371    Taro Tanaka    28     2006
   :             :          :       :
```

Employee No. CHAR(8)	Name VARCHAR(20)	Year hired CHAR(4)	Age SMALLINT
M2560024	Mike Johnson	1999	35
F3330199	Nancy White	2010	23
F3410087	Maria Gomez	2005	29
M1140371	Taro Tanaka	2006	28

Explanation:

- The length of a row of input data is fixed.
- The field data items in the input data files need not be in the same order that the columns appear in the table.

17.6.2 Coding rules for input data (fixed-length format)

- The length of a row of input data must be fixed.
- A line break code can be entered at the end of a row.
- Create input data files using the character encoding specified in the ADBLANG environment variable.
- Use uniform start positions for all column values. The following table explains how to make the start position uniform for each data type.

Table 17-15: How to make the start positions uniform

No.	Data type		Method for making start positions uniform	Handling of spaces and tabs
1	Numeric data		Use one of the following methods to set the start position (methods 2 and 3 can both be used at the same time): 1. Fill data with leading or trailing spaces (`x20`). 2. Fill integer data with leading zeros. 3. Fill decimal data with trailing zeros.	Before data is stored, any unneeded spaces (`0x20`) and tabs (`0x09`) are deleted.#1
2	Character string data	Fixed length	--	--
3		Variable length	Fill data with trailing spaces (`x20`) to make start positions uniform.	Before data is stored, any trailing spaces (`0x20`) and tabs (`0x09`) are deleted.#2
4	Datetime data		Fill data with leading or trailing spaces (`x20`) to make start positions uniform.	Before data is stored, any unneeded spaces (`0x20`) and tabs (`0x09`) are deleted.#1
5	Binary data	Fixed length	Fill data with leading or trailing spaces (`x20`) to make start positions uniform.	Before data is stored, any unneeded spaces (`0x20`) and tabs (`0x09`) are deleted.#1
6		Variable length	Fill data with leading or trailing spaces (`x20`) to make start positions uniform.	Before data is stored, any unneeded spaces (`0x20`) and tabs (`0x09`) are deleted.#2

Legend:

--: Not applicable.

#1

If all the data is deleted, the input data becomes invalid.

#2

If all the data is deleted, it is treated as data with a length of zero.

17.6.3 Specification format of field data (fixed-length format)

Specify data all in character format.

For details about the specification values for each data type, see 17.5.3 Specification format of field data (CSV format).

The following are notes about the fixed-length format that differ from the CSV format:

- The method for specifying empty strings and default values for columns differs from that for the CSV format.

  To specify an empty string or a default value for a column, specify data to be treated as the empty string or the default value for the column in the `-c` option of the `adbcolumninfo` operand in the column structure information options. When input data that matches the specified value is detected, the empty string or the default value is stored in the corresponding column. Note that comparison is based on the input data before unneeded spaces and tabs have been removed.

Important

If the NOT NULL constraint is defined for a column and the corresponding data in the input data contains a specification that is treated as an empty string, a logical error occurs during execution of the adbimport command.

- Enclosing characters are treated as data.

  If enclosing characters are not needed as data, specify the part without the enclosing characters for the start position and length in the -p option of the adbcolumninfo operand.

  If enclosing characters are included as data, they need be specified in succession, unlike in the CSV format.

- Variable-length data might be treated as data with a length of zero depending on the specification method. For details, see Table 17-15: How to make the start positions uniform in 17.6.2 Coding rules for input data (fixed-length format).

17.7 Compression format of input data files

This section explains the compression format of input data files.

The `adbimport` command can even import input data files (CSV or fixed-length format) that are in the following compression format.

▪ Compression format of input data files

- GZIP format

The following GZIP-format compressed files can be imported by using the `adbimport` command:

- Input data files (CSV or fixed-length format) compressed in GZIP format by using the OS `gzip` command
- Output data files (CSV format) compressed in GZIP format by using the `adbexport` command with the `--compress GZIP` option specified

  For details about the compression format of output data files, see 14.6 Compression format of output data files.

Compressed input data files are imported by using the same method as for uncompressed input data files. Data can be imported by executing the `adbimport` command in which the absolute path to the storage location of the compressed input data file is specified for the input data path file.

Important

Only GZIP can be used as the compression format of input data files. Input data files compressed in other formats cannot be imported.

When importing input data files compressed in formats other than GZIP format, decompress the compressed input data files before importing them.

Note

For details about the format of input data files (CSV format), see 17.5 Format of input data files (CSV format specifications).

For details about the format of input data files (fixed-length format), see 17.6 Format of input data files (fixed-length format specifications).

17.8 Value that is stored for empty-string input data

The table below shows the actual value that is stored in the target column when input data is the empty string.

If input data is not the empty string, the value of that input data is always stored regardless of the specification made during table definition and the value of the `adb_import_null_string` import option.

Table 17-16: Value that is stored for empty-string input data

No.	Specification made in the table definition		Value of the adb_import_null_string import option	Actual value that is stored
	NOT NULL constraint	DEFAULT clause		
1	Not specified.	Not specified.	`DEFAULT`, or the option was omitted.	Null value
2			`NULL`	Null value
3		Specified.	`DEFAULT`, or the option was omitted.	Default value of the column#
4			`NULL`	Null value
5	`NOT NULL`	Not specified.	`DEFAULT`, or the option was omitted.	--
6			`NULL`	--
7		Specified.	`DEFAULT`, or the option was omitted.	Default value of the column#
8			`NULL`	--

Legend:

--: No value is stored because a logical error results.

#

For details about the default values of columns, see the following topics:

• *Specification format and rules for the CREATE TABLE statement* in *CREATE TABLE (define a table)* in *Definition SQL* in the manual *HADB SQL Reference*

• *Setting a default value for a column (DEFAULT clause)* in *Designing a table* in *Designing a Database* in the *HADB Setup and Operation Guide*

17.9 Handling logical errors in input data

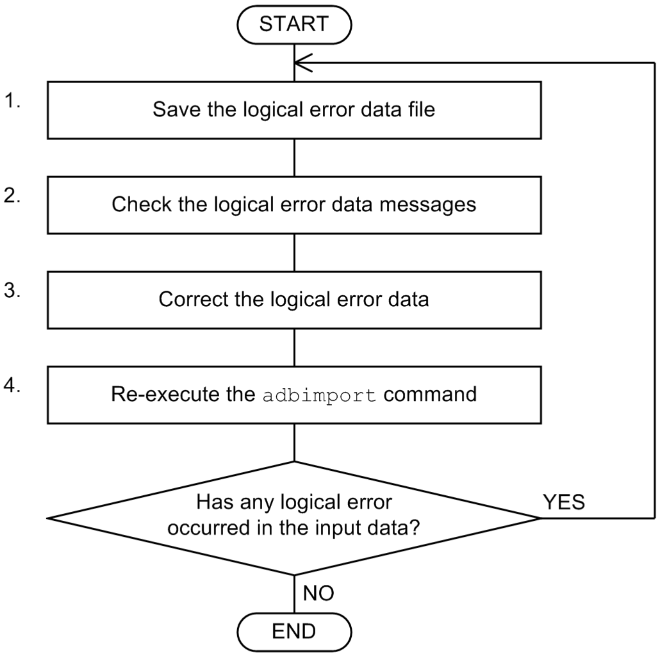
This section explains how to handle logical errors in input data.

17.9.1 How to handle logical errors in input data

Figure 17-18: Procedure for handling logical errors in input data shows the procedure for handling logical errors in input data. This procedure assumes that logical error data is output to a logical error data file.

If you are using input data files compressed in GZIP format, input data means decompressed data.

Figure 17-18: Procedure for handling logical errors in input data


START
1.
Save the logical error data file
2.
Check the logical error data messages
3.
Correct the logical error data
4.
Re-execute the adbimport command
Has any logical error occurred in the input data?
YES
NO
END

The following subsections explain each step.

(1) Saving the logical error data file

Save the logical error data file.

(2) Checking the logical error data messages

The KFAA50217-E through KFAA50224-E messages and the KFAA52100-E message concern logical error data. Check if any of these messages have been issued.

(3) Correcting the logical error data

The output pattern for the logical error data file differs depending on the combination of the following three factors:

- Type of data storage format for the table to be imported
- File format of input data files

- Whether the column structure information file (the `-r` option of the `adbimport` command) was specified during import processing

The following describes output patterns for the logical error data file:

- When row data containing logical errors is output without any processing

  When one of the following conditions is satisfied, the row data containing logical errors is output to the logical error data file without any processing:

  - When data is imported to a row store table, the file format of the input data files is CSV and the `-r` option is not specified for the `adbimport` command.
  - When data is imported to a row store table, the file format of the input data files is fixed-length.
  - Data is imported to a column store table.

- When, out of the row data containing logical errors, the only data extracted is field data destined for storage in the table to be processed

  When the following conditions are satisfied, only field data that was destined for storage in the table to be processed is extracted to the logical error data file:

  - Data is imported to a row store table.
  - The file format of the input data files is CSV.
  - A column structure information file was specified in the `-r` option of the `adbimport` command.

The following describes the handling method for each of these cases.

(a) When row data containing logical errors is output without any processing

Open the logical error data file that was saved in (1) Saving the logical error data file, and then correct the logical error data.

The order of the rows containing the logical error data corresponds to the order in which the messages checked in (2) Checking the logical error data messages were output. Check the messages and correct the data directly in the logical error data file.

Example of output message

```
KFAA50217-E  The input data length is invalid.
             (input file = 1, row = 1, column = 2)                ...1

KFAA50222-E  The input data contains an invalid empty string.
             (input file = 1, row = 2, column = 3)                ...2

    :
```

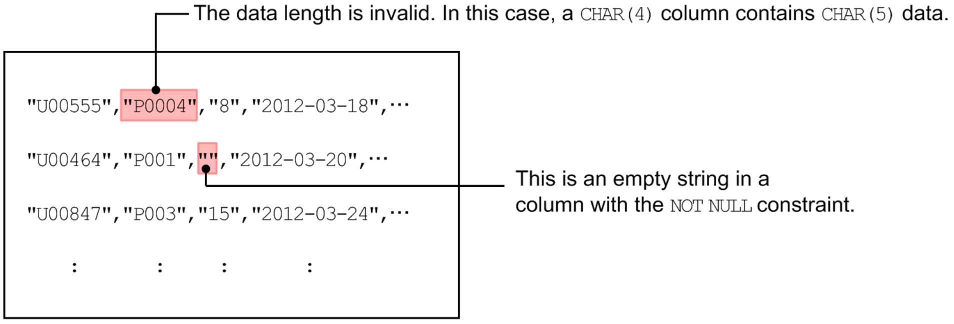
Explanation:

1. The `KFAA50217-E` message is issued when a data length is invalid.

   `input file = 1 ,row = 1, column = 2` indicates the location in the input data file of the data resulting in the logical error. In this example, the logical error occurred in the data located in `row` 1, `column` 2 of the first input data file (`input file`) specified in the input data path file.

2. The `KFAA50222-E` message is issued when data in a column with the `NOT NULL` constraint contains the empty string.

Content of the logical error data file (when the row data is output without any processing)



The data length is invalid. In this case, a CHAR(4) column contains CHAR(5) data.

```
"U00555","P0004","8","2012-03-18",…
"U00464","P001","","2012-03-20",…
"U00847","P003","15","2012-03-24",…
    :        :     :         :
```

This is an empty string in a column with the NOT NULL constraint.

(b) When only data that was destined for storage in the table to be processed is extracted

The handling method differs depending on whether the logical error data is field data that was destined for storage in the table to be processed.

- If field data that was destined for storage in the table to be processed caused a logical error:

  The field data that caused an error is output to the logical error data file.

  In this case, open the logical error data file that was saved as explained in (1) Saving the logical error data file, and correct the logical error data.

  The order of the rows containing the logical error data corresponds to the order in which the messages checked in (2) Checking the logical error data messages were output. Check the messages and correct the data directly in the logical error data file.

- If field data that was not destined for storage in the table to be processed caused a logical error:

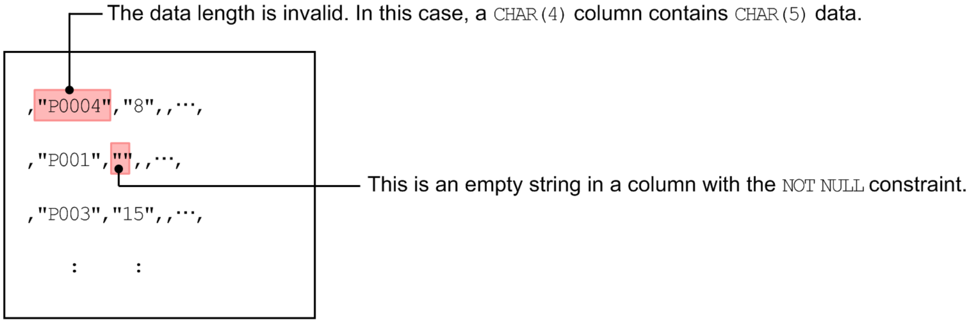
  The field data that caused an error is not output to the logical error data file.

  In this case, see the message you checked in (2) Checking the logical error data messages, and identify the cause of the error in the input data file.

  After the corresponding row data in the input data file is checked, if the content that was output to the logical error data file shows no problem, there is no need to correct that file. If the content that was output to the logical error data file shows a problem, extract the applicable row data from the input data file and correct it.

The message that is output is the same as when row data containing logical errors is output without any processing. For a message example, see (a) When row data containing logical errors is output without any processing.

Content of the logical error data file (when only data that was destined for storage is extracted)



The data length is invalid. In this case, a CHAR(4) column contains CHAR(5) data.

```
,"P0004","8",,…,
,"P001","",,…,
,"P003","15",,…,
    :      :
```

This is an empty string in a column with the NOT NULL constraint.

Note: If field data not destined to be stored in the table to be processed contains a logical error, the applicable field data is not output to the logical error data file. In this case, check the input data file.

Note

When only data that was destined for storage is extracted to the logical error data file, delimiters are extracted to the logical error data file so that the number of input data files matches the number of field data items. (Therefore, when you re-execute the `adbimport` command, you can use the corrected logical error data file as the input data file without changing the options.)

The (*largest field data number that did not become target for storage in table to be processed* + 1)th and subsequent delimiters are not extracted.

(4) Re-executing the adbimport command

Re-execute the `adbimport` command to import the corrected data. Specify the file containing the corrected data (logical error data file) as the input file and then re-execute the `adbimport` command.

If there was no need to correct the logical error data file, use the logical error data file that is output directly as the input data file. If you extracted row data from the input data file and corrected it, use the file in which you stored the extracted and corrected row data as the input data file.

Important

Make sure that you execute the `adbimport` command in addition mode (`-d` option omitted). If you execute the `adbimport` command in creation mode, the correct data that has already been imported will be deleted.

As a general rule, do not change any options other than the `-d` option. Also, do not change the import options.

17.10 Handling abnormal termination of the adbimport command

This subsection describes the actions to take if the `adbimport` command terminates abnormally.

17.10.1 First thing to check

Check if the following messages have been issued:

- `KFAA50225-E`
- `KFAA96606-E`

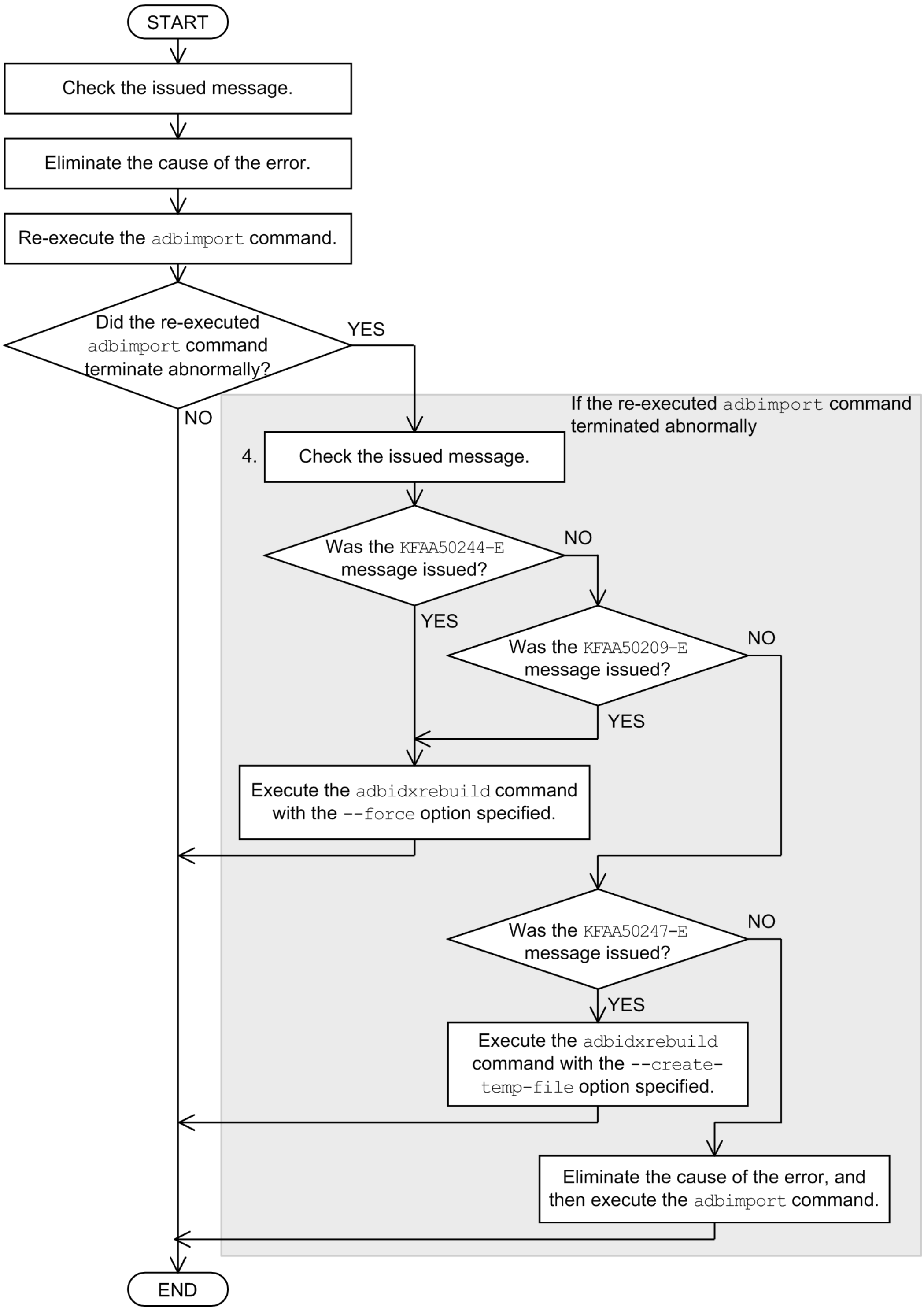
If either of these messages has been issued, there is an error in the specified command options or import options. Check and correct the command options and import options and then re-execute the `adbimport` command.

If neither of these messages was issued, go to 17.10.2 Identifying the cause of the error and taking corrective action.

17.10.2 Identifying the cause of the error and taking corrective action

The following figure shows the procedural flow to follow if the `adbimport` command terminates abnormally.

Figure 17-19: Procedural flow to follow if the adbimport command terminates abnormally


START
1. Check the issued message.
2. Eliminate the cause of the error.
3. Re-execute the adbimport command.
Did the re-executed adbimport command terminate abnormally?
YES
NO
If the re-executed adbimport command terminated abnormally
4. Check the issued message.
Was the KFAA50244-E message issued?
NO
YES
Was the KFAA50209-E message issued?
NO
YES
Execute the adbidxrebuild command with the --force option specified.
Was the KFAA50247-E message issued?
NO
YES
Execute the adbidxrebuild command with the --create-temp-file option specified.
Eliminate the cause of the error, and then execute the adbimport command.
END

Procedure

1. Check the messages that have been issued and identify the cause of the error.
2. Take appropriate action to eliminate the cause of the error.
3. Re-execute the `adbimport` command.
4. If the re-executed `adbimport` command terminates abnormally, check the message that was issued and execute the `adbidxrebuild` or `adbimport` command.

   For details, see 17.10.7 If an error occurs during re-execution of the adbimport command.

If a B-tree index has been defined for a table subject to import processing (including for tables in which a primary key is defined), or if a text index has been defined for a table subject to import processing, see 17.10.3 When a B-tree index or a text index has been defined for a table subject to import processing. If logical error data is set to be output, see 17.10.4 When logical error data is set to be output.

■ When the multi-node function is used

If the `adbimport` command is canceled and the table to be processed becomes non-updatable, release the base table from updatable status by re-executing the canceled `adbimport` command.

If the master node was switched due to abnormal termination of the HADB server on the master node, execute the `adbidxrebuild` command on the new master node to release the base table from non-updatable status. At this time, execute the `adbidxrebuild` command with the `--create-temp-file` option specified.

For details about how to switch the master node to another node and how to start the HADB server without using the multi-node function, see the topic *Problems related to startup or termination of the HADB servers in the multi-node configuration* in *Troubleshooting (when the multi-node function is being used)* in *Operations When Using the Multi-Node Function* in the *HADB Setup and Operation Guide*.

For details about how to execute the `adbidxrebuild` command with the `--create-temp-file` option specified, see (1) Executing the adbidxrebuild command with the --create-temp-file option specified in 17.10.7 If an error occurs during re-execution of the adbimport command.

17.10.3 When a B-tree index or a text index has been defined for a table subject to import processing

If a B-tree index has been defined for a table subject to import processing (including for tables in which a primary key is defined), or if a text index has been defined for a table subject to import processing, check whether the `KFAA80203-I` message has been issued before you execute the `adbimport` command. The `KFAA80203-I` message is issued when table data import processing is completed. If this message has been issued, it means that the time required for re-executing data import processing can be reduced because the re-execute facility that starts from the processing to create B-tree indexes and text indexes is being applied.

If this message has not been issued, you must perform data import processing from the beginning (starting with importing the table data).

The re-execute facility that starts from the processing to create B-tree indexes and text indexes is applied automatically when the `adbimport` command is re-executed by the same HADB user who executed the `adbimport` command that terminated abnormally. No special operation is required.

If you want to change command options and import options when re-executing the `adbimport` command, see 17.10.6 Changing command options and import options when re-executing commands.

Important

- If an access error occurred on a command status file (file under the `$DBDIR/ADBSYS/ADBUTL` directory) and the `KFAA50244-E` message has been issued, execute the `adbidxrebuild` command with the `--force` option specified.

  Also, when an index is defined for the table to be processed while the `adbimport` command is stopped, and the `KFAA50209-E` message has been issued, execute the `adbidxrebuild` command with the `--force` option specified. If the `adbimport` command has not been canceled, execute the `adbidxrebuild` command without the `--force` option specified.

For details about how to execute the `adbidxrebuild` command with the `--force` option specified, see (2) Executing the adbidxrebuild command with the --force option specified in 17.10.7 If an error occurs during re-execution of the adbimport command.

- If an access error occurs on a temporary work file (the `KFAA50247-E` message has been issued), execute the `adbidxrebuild` command with the `--create-temp-file` option specified.

  For details about how to execute the `adbidxrebuild` command with the `--create-temp-file` option specified, see (1) Executing the adbidxrebuild command with the --create-temp-file option specified in 17.10.7 If an error occurs during re-execution of the adbimport command.

17.10.4 When logical error data is set to be output

If the `KFAA80203-I` message explained in 17.10.3 When a B-tree index or a text index has been defined for a table subject to import processing has been issued, the table data import processing has been completed, and logical error data has been output to the logical error data file. After the `adbimport` command has been re-executed, see 17.9.1 How to handle logical errors in input data and handle the logical error.

If the `KFAA80203-I` message has not been issued, the table data import processing has not been completed. This also means that there might be logical error data that has not been output to the logical error data file. Correct the portion of the logical error data on the basis of the logical error data that has been output, or re-execute the `adbimport` command and correct all of the logical error data on the basis of the logical error data that has been output so far.

17.10.5 Whether the re-execute facility that starts from the processing to create B-tree indexes and text indexes can be applied

Depending on the `adbimport` command's cancellation timing, the re-execute facility that starts from the processing to create B-tree indexes and text indexes might or might not be applied when the `adbimport` command is re-executed.

The table below describes whether the re-execute facility that starts from the processing to create B-tree indexes and text indexes is applied, depending on the `adbimport` command's cancellation timing.

Note that the database recovery method is the same regardless of the cancellation timing.

Table 17-17: Whether the re-execute facility that starts from the processing to create B-tree indexes and text indexes is applied, depending on the adbimport command's cancellation timing

-d option	Timing of the cancellation[#1]		
	While creating table data	During sort processing	During creation of B-tree and text indexes[#2]
Specified (creation mode)	• **Whether the re-execute facility is applied** Not applied.[#3] • **Table status** The table is in one of the following statuses: • There is no data.	• **Whether the re-execute facility is applied** Applied.[#4] • **Table status** The table is non-updatable. • **Index status** Import of range indexes is complete. Import of B-tree and text indexes is incomplete.	

-d option	Timing of the cancellation#1		
	While creating table data	During sort processing	During creation of B-tree and text indexes#2
	• Status before the `adbimport` command was executed. - **Index status** A B-tree index, text index, or range index is in one of the following statuses: • There is no data. • Status before the `adbimport` command was executed.		
Omitted (addition mode)	- **Whether the re-execute facility is applied** Not applied.#3 - **Table status** The table is in the status that it was in before the `adbimport` command was executed. - **Index status** A B-tree index, text index, or range index is in the status that it was in before the `adbimport` command was executed.	- **Whether the re-execute facility is applied** Applied.#4 - **Table status** The table is non-updatable. - **Index status** Import of range indexes is complete. B-tree and text indexes are in the status that they were in before the `adbimport` command was executed.	

Legend:

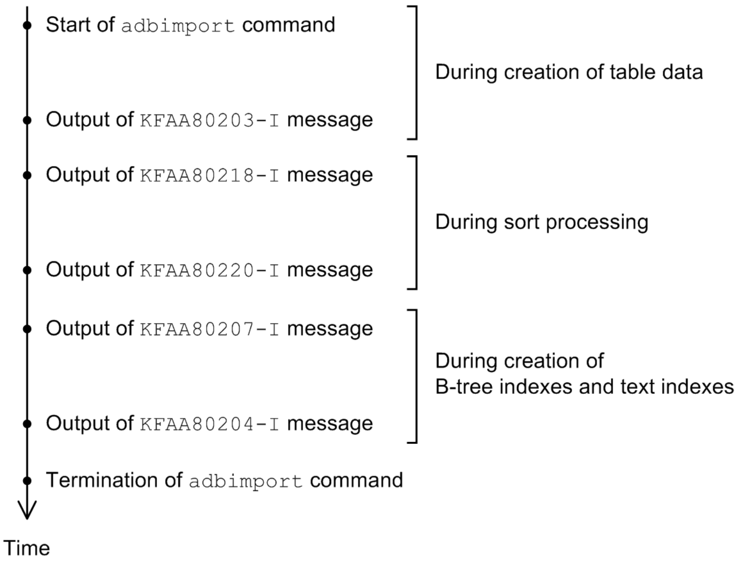
Re-execute facility: Re-execute facility that starts from the processing to create B-tree indexes and text indexes

Note:

B-tree indexes include the B-tree index for the primary key.

#1

You can determine when the command was canceled by the messages that are issued, as shown in the following:



Start of `adbimport` command

Output of `KFAA80203-I` message

During creation of table data

Output of `KFAA80218-I` message

Output of `KFAA80220-I` message

During sort processing

Output of `KFAA80207-I` message

Output of `KFAA80204-I` message

During creation of B-tree indexes and text indexes

Termination of `adbimport` command

#2

If the KFAA80204-I message (whose return code is 0 or 4) has been issued, the table and indexes have been imported. In this case, there is no need to re-execute the adbimport command.

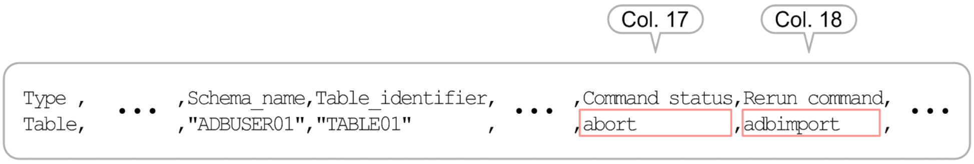
If the KFAA80204-I message is not output, execute the following command, and then check the second line in the output results (the column titles occupy the first line in the output results):

```
adbdbstatus -c table -n name-of-table-for-which-adbimport-command-was-canceled
```

Specify *name-of-table-for-which-adbimport-command-was-canceled* in the format *schema-name*.*table-identifier*.

■ **When the value of the Command_status column (column 17) is abort and the value of the Rerun_command column (column 18) is adbimport**

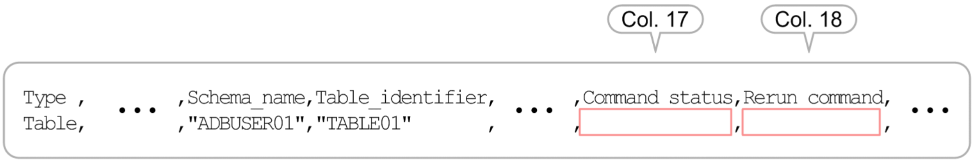
Import processing was not completed. Re-execute the adbimport command.



```
Type ,   ...  ,Schema_name,Table_identifier,  ...  ,Command_status,Rerun_command,  ...
Table,        ,"ADBUSER01","TABLE01"       ,       ,abort         ,adbimport    ,
```

■ **When no values are displayed in the Command_status column (column 17) and the Rerun_command column (column 18)**

Import processing has been completed. There is no need to re-execute the adbimport command.



```
Type ,   ...  ,Schema_name,Table_identifier,  ...  ,Command_status,Rerun_command,  ...
Table,        ,"ADBUSER01","TABLE01"       ,       ,              ,             ,
```

#3

When the adbimport command is re-executed, the data import processing is performed from the beginning.

#4

When the adbimport command is re-executed, the data import processing restarts from the point of B-tree and text index creation processing. However, if all the B-tree indexes and text indexes defined for the table to be processed are deleted before the adbimport command is re-executed, only command termination processing is performed.

17.10.6 Changing command options and import options when re-executing commands

This subsection explains the effects of changing command options and import options when the adbimport command is re-executed.

(1) Changing command options

Whether a changed command option takes effect depends on when the adbimport command is canceled. The following table describes how to resume processing when changing command options.

Table 17-18: Resuming processing when changing command options

No.	Command option to be changed	When canceled#		
		During creation of a data part of a table	During sort processing	During creation of B-tree and text indexes
1	`-u` option	N	N	N
2	`-p` option	N	N	N
3	`-d` option	Y	--	--
4	`-k` option	Y	--	--
5	`-s` option	Y	--	--
6	`-g` option	Y	--	--
7	`-w` option	Y	--	--
8	`-z` option	Y	Y	Y
9	`-e` option	Y	--	--
10	`--force` option	If you specify `--force`, the processing is forcibly re-executed. Therefore, this option does not apply.		
11	`-b` option	Y	--	--
12	`--status wait` option	Y	--	--
13	`-m` option	Y	Y	Y
14	`-f` option	Y	--	--
15	`-r` option	Y	--	--
16	*table-to-be-processed*	N	N	N
17	*input-data-path-file-name*	Y	--	--

Legend:

Y: Changes take effect.

N: Changes will not cause re-execution.

--: Changes are ignored (the processing is executed using the specification before the change).

#

For details about the timing of cancellation, see Table 17-17: Whether the re-execute facility that starts from the processing to create B-tree indexes and text indexes is applied, depending on the adbimport command's cancellation timing.

(2) Changing import options

Whether a changed import option takes effect depends on when the `adbimport` command is canceled. The following table describes how to resume processing when changing import options.

Table 17-19: Resuming processing when changing import options

No.	Import option to be changed	When canceled#		
		During creation of a data part of a table	During sort processing	During creation of B-tree and text indexes
1	`adb_import_errdata_file_name`	Y	--	--
2	`adb_import_errdata_num`	Y	--	--
3	`adb_import_errmsg_lv`	Y	--	--
4	`adb_import_rthd_num`	Y	Y	Y
5	`adb_import_read_size`	Y	--	--
6	`adb_import_buff_blk_num`	Y	--	--
7	`adb_import_sort_buff_size`	Y	Y	--
8	`adb_import_dividx_rd_buff_size`	Y	Y	Y
9	`adb_import_dividx_wt_buff_size`	Y	Y	Y
10	`adb_import_txt_buff_size`	Y	Y	Y
11	`adb_import_decompress_buff_size`	Y	--	--
12	`adb_import_null_string`	Y	--	--

Legend:

Y: Changes take effect.

--: Changes are ignored (the processing is executed using the specification before the change).

#

For details about the timing of cancellation, see Table 17-17: Whether the re-execute facility that starts from the processing to create B-tree indexes and text indexes is applied, depending on the adbimport command's cancellation timing.

17.10.7 If an error occurs during re-execution of the adbimport command

If an error occurs during re-execution of the `adbimport` command, identify the cause of the error and take the necessary action according to the following procedure.

Procedure

1. Check whether the `KFAA50244-E` message has been issued.

   If the `KFAA50244-E` message has been issued, an access error occurred on a command status file (file under the `$DBDIR/ADBSYS/ADBUTL` directory). Execute the `adbidxrebuild` command with the `--force` option specified. The `adbidxrebuild` command can inherit index creation processing by the `adbimport` command.

   For details, see (2) Executing the adbidxrebuild command with the --force option specified.

   If the `KFAA50244-E` message has not been issued, proceed to either of the following steps.

   - If an index was defined for the table to be processed while the `adbimport` command was stopped:

     Proceed to step 2.

   - If an index was not defined for the table to be processed while the `adbimport` command was stopped:

Proceed to step 3.

2. Check whether the KFAA50209-E message has been issued.

   If the KFAA50209-E message has been issued, the adbimport command cannot be re-executed because an index in unfinished status exists. Execute the adbidxrebuild command with the --force option specified. The adbidxrebuild command can inherit index creation processing by the adbimport command.

   For details, see (2) Executing the adbidxrebuild command with the --force option specified.

   If the KFAA50209-E message has not been issued, proceed to step 3.

3. Check whether the KFAA50247-E message has been issued.

   If the KFAA50247-E message has been issued, an access error occurred on a temporary work file. Execute the adbidxrebuild command with the --create-temp-file option specified. The adbidxrebuild command can inherit index creation processing by the adbimport command.

   For details, see (1) Executing the adbidxrebuild command with the --create-temp-file option specified.

   If the KFAA50247-E message has not been issued, proceed to step 4.

4. Check whether a message other than KFAA50209-E, KFAA50244-E, or KFAA50247-E has been issued.

   Check the message that was issued and identify the cause of the error. After removing the cause of the error, execute the adbimport command.

For an overview of the process during which the adbidxrebuild command inherits index creation processing, see (2) Inheriting index creation processing from the adbimport command in 16.1.3 Optional functions of the adbidxrebuild command.

(1) Executing the adbidxrebuild command with the --create-temp-file option specified

If the KFAA50247-E message has been issued, an access error occurred on a temporary work file. Execute the adbidxrebuild command with the --create-temp-file option specified.

Important

- Do not delete the command status file (file under the $DBDIR/ADBSYS/ADBUTL directory) for the adbimport command until the executed adbidxrebuild command terminates normally. If there is no command status file when the adbidxrebuild command is executed, an error might occur in the adbidxrebuild command. (The KFAA50244-E message is issued.) In this case, execute the adbidxrebuild command with the --force option specified instead of the --create-temp-file option.
- It is assumed that the --create-temp-file option of the adbidxrebuild will be specified when there is no temporary work file necessary for re-execution (index record file or sort result file). If the adbidxrebuild command is executed when a temporary work file exists, the existing file will be re-built. However, if you do not have write or read privileges for the applicable file, an access error might occur during execution of the adbidxrebuild command. (The KFAA30959-E, KFAA40204-E, KFAA40205-E, KFAA40214-E, KFAA41205-E, or KFAA41206-I message is issued. 13 (EACCES) is output for the error number.) In such a case, first delete the applicable file using the OS's rm command or a similar method, and then re-execute the adbidxrebuild command.
- When the --create-temp-file option is specified for the adbidxrebuild command, the adbidxrebuild command is executed, inheriting the content of the options that were specified for the adbimport command. If the -b, -m, or --status wait option was specified, the specified content is also inherited.

These options cannot be specified for the `adbidxrebuild` command since they are `adbimport` command options. These options are only inherited when the `--create-temp-file` option is specified.

- After cancelation of the `adbidxrebuild` command with the `--create-temp-file` option specified, if the `adbidxrebuild` command is executed, the command resumes index rebuild processing from the point of table retrieval in the chunk for which processing was canceled, regardless of whether the `--create-temp-file` option is specified.

The following table shows the information to specify for the `adbidxrebuild` command when it is executed with the `--create-temp-file` option specified.

Table 17-20: Information to specify for the adbidxrebuild command (when the --create-temp-file option is specified)

No.	adbidxrebuild command specification format	Whether options can be changed from the canceled adbimport command	Remarks
1	`adbidxrebuild`	Specify `adbidxrebuild`.	--
2	`-u` *authorization-identifier*	N	--
3	`[-p` *password*`]`	N	--
4	`[-g` *output-interval-for-index-rebuild-processing-progress-message*`]`	O	--
5	`[-w {`*temporary-work-file-storage-directory-name* `|` *name-of-directory-path-file-specifying-temporary-work-file-storage-locations*`}]`	Y	If the disk that stores temporary work files runs out of free space, specify another disk with a larger capacity as the storage destination. For details about the actions to take if the disk that stores temporary work files runs out of free space, see *Steps to take in the event of a shortage of disk space for storing temporary work files during command execution* in *Command-related problems* in *Troubleshooting* in the *HADB Setup and Operation Guide*.
6	`[-z` *index-rebuild-option-file-name*`]`	Y	For details about the values to be specified for index rebuild options, see Table 17-21: Index rebuild options that correspond to import options (when the --create-temp-file option is specified).
7	`[-i` *index-identifier-file-name*`]`	There is no need to specify an option. (An option will be ignored even if it is specified.)	Applies to the indexes defined for the table to be processed.
8	`[{--create-temp-file|--force}]`	Specify `--create-temp-file`.	--
9	*table-to-be-processed*	N	--

Legend:

Y: The specification value can be changed from the option of the canceled `adbimport` command.

O: Option specification is optional.

N: The specification value cannot be changed from the option of the canceled `adbimport` command.

--: Not applicable

If there are index rebuild options that correspond to the import options you specified when executing the canceled adbimport command, specify values that are close to the values specified for the import options. If you specify for the index rebuild options values that are different from the values specified for the import options, the adbidxrebuild command is executed with the values specified for the index rebuild options.

The following table shows index rebuild options that correspond to import options.

Table 17-21: Index rebuild options that correspond to import options (when the --create-temp-file option is specified)

No.	Import option	Corresponding index rebuild option
1	adb_import_errdata_file_name	N
2	adb_import_errdata_num	N
3	adb_import_errmsg_lv	N
4	adb_import_rthd_num	adb_idxrebuild_rthd_num#
5	adb_import_read_size	N
6	--	adb_idxrebuild_scan_buff_size
7	adb_import_buff_blk_num	adb_idxrebuild_buff_blk_num
8	adb_import_sort_buff_size	adb_idxrebuild_sort_buff_size
9	adb_import_dividx_rd_buff_size	adb_idxrebuild_dvix_rdbuff_size
10	adb_import_dividx_wt_buff_size	adb_idxrebuild_dvix_wtbuff_size
11	adb_import_txt_buff_size	adb_idxrebuild_txt_buff_size
12	adb_import_decompress_buff_size	N
13	adb_import_null_string	N

Legend:

N: There is no corresponding index rebuild option.

--: There is no applicable import option. Specify a value based on the explanation in the specification guidelines for index rebuild options. For details about the specification guidelines for index rebuild options, see (2) Explanation of specification format in 16.2.2 Format of index rebuild options.

#

If a value smaller than 3 is specified for import option adb_import_rthd_num, specify 3 for index rebuild option adb_idxrebuild_rthd_num.

(2) Executing the adbidxrebuild command with the --force option specified

In the following cases, you need to execute the adbidxrebuild command with the --force option specified.

- If an index is defined for the table to be processed while the adbimport command is stopped, and the KFAA50209-E message is issued during re-execution of the adbimport command:

  The adbimport command cannot be re-executed because an index in unfinished status exists.

- If the KFAA50244-E message is issued during re-execution of the adbimport command:

  An access error occurred on a command status file (file under the $DBDIR/ADBSYS/ADBUTL directory).

In these cases, take the necessary action according to the following procedure.

Procedure

1. Delete the temporary work files.

   Delete the files under the following directories (temporary work files) using the OS's `rm` command or a similar method. If there are any files for which you do not have write or read privileges, you might not be able to execute the `adbidxrebuild` command.

   - If the `-w` option was specified during execution of the previous command that was canceled:

     The temporary work file storage directories specified in the `-w` option

   - If the `-w` option was omitted during execution of the previous command that was canceled:

     The `$DBDIR/ADBWORK` directory

2. Execute the `adbidxrebuild` command with the `--force` option specified.

   Execute the `adbidxrebuild` command with the `--force` option specified.

   If the previously canceled command was the `adbimport` command with background import processing applied (the `-b` option specified), proceed to step 3. In all other cases, skip step 3.

3. Execute the `adbimport` command with the `-b` option specified.

   After cancellation of the `adbimport` command with background import processing applied (the `-b` option specified), executing the `adbidxrebuild` command with the `--force` option specified deletes the storage information of table data that was being processed by the `adbimport` command. Therefore, after the `adbidxrebuild` command completes, re-execute the canceled `adbimport` command with the `-b` option specified.

> **Important**
>
> After the `adbidxrebuild` command with the `--force` option specified is canceled, if you re-execute the `adbidxrebuild` command, index rebuild processing is forcibly re-executed from the beginning even if you do not specify the `--force` option. Therefore, even if you specify the `--create-temp-file` option, it is ignored. When index rebuild processing is forcibly re-executed from the beginning, the `KAFF80215-I` message is issued.

The following table shows the information to specify for the `adbidxrebuild` command when you execute it with the `--force` option specified.

Table 17-22: Information to specify for the adbidxrebuild command (when the --force option is specified)

No.	adbidxrebuild command specification format	Whether options can be changed from the canceled adbimport command	Remarks
1	`adbidxrebuild`	Specify `adbidxrebuild`.	--
2	`-u` *authorization-identifier*	Y	If the executing user of the `adbimport` command was deleted or if the `IMPORT TABLE` privilege was revoked, and as a result the same executing user can no longer execute the `adbimport` command, take the following action: Execute the `adbidxrebuild` command by specifying an authorization identifier that has the `REBUILD INDEX` privilege.

No.	adbidxrebuild command specification format	Whether options can be changed from the canceled adbimport command	Remarks
3	`[-p` *password*`]`	Y	If the authorization identifier was changed, specify the password for the authorization identifier.
4	`[-g` *output-interval-for-index-rebuild-processing-progress-message*`]`	O	--
5	`[-w {`*temporary-work-file-storage-directory-name*`|`*name-of-directory-path-file-specifying-temporary-work-file-storage-locations*`}]`	Y	If the disk that stores temporary work files runs out of free space, specify another disk with a larger capacity as the storage destination. For details about the actions to take if the disk that stores temporary work files runs out of free space, see *Steps to take in the event of a shortage of disk space for storing temporary work files during command execution* in *Command-related problems* in *Troubleshooting* in the *HADB Setup and Operation Guide*.
6	`[-z` *index-rebuild-option-file-name*`]`	Y	For details about the values to be specified for index rebuild options, see Table 17-23: Index rebuild options that correspond to import options (when the --force option is specified).
7	`[-i` *index-identifier-file-name*`]`	There is no need to specify an option. (An option will be ignored even if it is specified.)	Applies to the indexes defined for the table to be processed.
8	`[{--create-temp-file|--force}]`	Specify `--force`.	--
9	*table-to-be-processed*	N	--

Legend:

Y: The specification value can be changed from the option of the canceled `adbimport` command.

O: Option specification is optional.

N: The specification value cannot be changed from the option of the canceled `adbimport` command.

--: Not applicable

If there are index rebuild options that correspond to the import options specified when executing the canceled `adbimport` command, specify values that are close to the values specified for the import options. If you specify for the index rebuild options values that are different from the values specified for the import options, the `adbidxrebuild` command is executed with the values specified for the index rebuild options.

The following table shows index rebuild options that correspond to import options.

Table 17-23: Index rebuild options that correspond to import options (when the --force option is specified)

No.	Import option	Corresponding index rebuild option
1	`adb_import_errdata_file_name`	N
2	`adb_import_errdata_num`	N
3	`adb_import_errmsg_lv`	N
4	`adb_import_rthd_num`	`adb_idxrebuild_rthd_num`#
5	`adb_import_read_size`	N

No.	Import option	Corresponding index rebuild option
6	--	adb_idxrebuild_scan_buff_size
7	adb_import_buff_blk_num	adb_idxrebuild_buff_blk_num
8	adb_import_sort_buff_size	adb_idxrebuild_sort_buff_size
9	adb_import_dividx_rd_buff_size	adb_idxrebuild_dvix_rdbuff_size
10	adb_import_dividx_wt_buff_size	adb_idxrebuild_dvix_wtbuff_size
11	adb_import_txt_buff_size	adb_idxrebuild_txt_buff_size
12	adb_import_decompress_buff_size	N
13	adb_import_null_string	N

Legend:

N: There is no corresponding index rebuild option.

--: There is no applicable import option. Specify a value based on the explanation in the specification guidelines for index rebuild options. For details about the specification guidelines for index rebuild options, see (2) Explanation of specification format in 16.2.2 Format of index rebuild options.

#

If a value smaller than 3 is specified for import option `adb_import_rthd_num`, specify `3` for index rebuild option `adb_idxrebuild_rthd_num`.

17.11 Notes

1. If the adbimport command is executed while another command or application program is running, HADB might not be able to allocate the required number of processing real threads. If this happens, HADB waits until the required number of processing real threads can be allocated before executing the adbimport command. If allocation of processing real threads is placed in wait status, see the topic *Points to consider about the number of processing real threads to be used during command execution* in *Points to consider when executing commands concurrently* in *Preparing Resources* in the *HADB Setup and Operation Guide* and take appropriate action.

2. Do not perform read or write operations on the input data files (by using other applications) while the adbimport command is being executed.

3. Use X'0A' (LF) or X'0D0A' (CRLF) for the line break code in files specified in the options of the adbimport command and in import options.

4. Data import processing continues even if there is a duplicated key value in a unique index (the B-tree index for the primary key or a B-tree index for which UNIQUE was specified in the CREATE INDEX statement). The KFAA61205-W warning message indicating detection of duplicate key values is issued. In such a case, the table and the B-tree index subject to data import processing violate the uniqueness constraint. Locate the duplicate key values and delete the corresponding rows (a B-tree index that violates the uniqueness constraint is in the status called uniqueness constraint violation). Then re-execute the adbidxrebuild command and rebuild the corresponding B-tree index.

5. Use the adbcancel command to cancel execution of the adbimport command. Alternatively, use a method such as the adbstop --force command or the OS's kill command to forcibly terminate the server process. For details about how to use the adbcancel command to suspend an executing command, see 1.6 When suspending (forcibly terminating) an executing command.

6. Data import processing is performed by server processes. Therefore, using the **Ctrl**+**C** keys to forcibly terminate the adbimport command process will not interrupt data import processing that is being performed by server processes (such processing continues). When the data import processing is completed, the command outputs the KFAA81001-I message to the message log file.

7. When data is imported to a column store table, information about the column-data compression type and the data storage rate is output in the following messages:

   - KFAA82000-I: The column-data compression type for each column is output. This message is output for each thread that performs data import processing.
   - KFAA82002-I: The data storage rate for each column is output.
   - KFAA82003-I: The data storage rate for the entire table is output.

8. Work files (temporary work files) are created during data import processing. These temporary work files are deleted automatically when the adbimport command terminates normally (the return code is 0 or 4). If the adbimport command does not terminate normally, these work files might not be deleted.

   Because some of the temporary work files will be needed when the re-execute facility that starts from the processing to create B-tree indexes and text indexes is used, do not delete temporary work files before you use the re-execute facility. If you do delete them, the re-execute facility will not function correctly.

9. Because temporary work files might cause a shortage of disk space, we recommend that you delete temporary work files from time to time in the following cases:

   - There will be no need to re-execute the adbimport command.
   - There is no need to re-execute the adbidxrebuild command.

   Check the following directories periodically for any remaining temporary work files and delete any that are found:

   - $DBDIR/ADBSYS/ADBUTL

- Temporary work file storage directories specified in the `-w` option (or in `$DBDIR/ADBWORK` when the `-w` option was omitted)

10. The following table shows how the `adbidxrebuild` command inherits index creation processing from the `adbimport` command.

Table 17-24: How the adbidxrebuild command inherits index creation processing from the adbimport command

No.	Canceled command[#1]	Command to be executed following cancelation			Whether processing is inherited
		Command name	Option specification		
			--create-temp-file option	--force option	
1	`adbimport`	`adbidxrebuild`	Specified	Not specified	Y[#2, #3]
2			Not specified	Specified	Y[#2, #4, #5]
3			Not specified	Not specified	N

Legend:

Y: Index creation processing can be inherited.

N: Index creation processing cannot be inherited. (The `KFAA50201-E` message is issued.)

#1

To identify the canceled command, output the table summary information by executing the `adbdbstatus` command. The name of the command that was canceled is output to the `Rerun_command` column in the output result. For details about the `adbdbstatus` command, see 13. adbdbstatus (Analyze the Database Status).

If the `adbidxrebuild` command is executed and then canceled after cancelation of the `adbimport` command, `adbidxrebuild` is displayed in the `Rerun_command` column. If the `--create-temp-file` option was specified for the canceled `adbidxrebuild` command, re-execute the `adbidxrebuild` command with the `--create-temp-file` option specified. In this case, the information about the cancelation of the `adbimport` command is saved in a command status file (file under the `$DBDIR/ADBSYS/ADBUTL` directory), and therefore indexes are rebuilt for the chunk that was being processed when the `adbimport` command was canceled.

#2

Executing the `adbidxrebuild` command rebuilds all indexes that were defined for the table to be processed. The `-i` option of the `adbidxrebuild` command is ignored even if it is specified.

#3

If the `KFAA50247-E` message is output during re-execution of the `adbimport` command (no temporary work file exists), specify the `--create-temp-file` option, and then execute the `adbidxrebuild` command. In this way, index creation processing can be inherited.

#4

After an index is defined in the target table to be processed while the `adbimport` command is canceled, the `KFAA50209-E` message might be output (if there are indexes in unfinished status) if the `adbimport` command is re-executed. In this case, specify the `--force` option to execute the `adbidxrebuild` command. In this way, index creation processing can be inherited.

To identify the indexes in unfinished status, output the index summary information by executing the `adbdbstatus` command, and check the `Unfinished` column in the output result. If an index is in unfinished status, `unfinished` is output to the `Unfinished` column in the output result. For details about the `adbdbstatus` command, see 13. adbdbstatus (Analyze the Database Status).

If the adbimport command has not been canceled, execute the adbidxrebuild command without the --force option specified.

#5

If the KFAA50244-E message is issued while the adbimport command is stopped (no command status file exists), execute the adbidxrebuild command with the --force option specified. In this way, index creation processing can be inherited.

11. When you forcibly execute the adbimport command with the --force option specified, you must also specify the -d option (creation mode) at the same time. Therefore, all data stored in the table is deleted during forcible execution, and input data is stored after the table has been initialized. For details about how to forcibly execute the adbimport command, see --force in (3) Explanation of options in 17.2.1 Explanation of the specification format and options.
12. If the adbimport command is executed on a newly defined table, data is imported in the creation mode, even if the -d option is omitted. Therefore, if the adbimport command terminates abnormally while data is being imported into the newly defined table, see the handling methods indicated for *Specified (creation mode)* in the -d option column in Table 17-17: Whether the re-execute facility that starts from the processing to create B-tree indexes and text indexes is applied, depending on the adbimport command's cancellation timing.

    In the case of a table in which data was stored and then the number of data items was set to zero by an SQL statement, if the -d option is omitted, the adbimport command imports data in addition mode.
13. Make sure that there is no duplication of absolute paths for the directories and files specified in the adbimport command's options or import options. This applies to the following directories and files:
    - Temporary work file storage directories (specified in the -w option or specified in the directory path file specified in the -w option)
    - Directory path files specifying the storage locations of temporary work files (specified in the -w option)
    - Import option file (specified in the -z option)
    - Column structure information file (specified in the -r option)
    - Input data path file (specified in the command option)
    - Input data files (specified in the input data path file)
    - Logical error data file (specified in the adb_import_errdata_file_name import option)

    Make sure that the absolute paths of these directories and files are unique from those of any file names used by any programs other than the adbimport command. If a duplicate file name is used, data in the file might be lost. Also make sure that symbolic links and relative paths are specified correctly.
14. Note the following when you execute a retrieval SQL statement and the adbimport command concurrently by using background import:
    - Because the table that is to be retrieved is also subject to import processing, the performance of retrieval SQL statements might deteriorate due to conflicts between input and output.
    - If the sum of the number of threads used by the adbimport command and the number of the threads used by the retrieval SQL statements is greater than the value of the adb_sys_rthd_num operand specified in the server definition, processing will not start until the preceding processing has been completed (the subsequent processing will be placed in wait status).
15. Do not delete a user who is executing the adbimport command or revoke such a user's IMPORT TABLE privilege while data is being imported. If the executing user is deleted or the user's privileges are revoked by mistake and the adbimport command terminates abnormally, it will not be possible to use the re-execute facility that starts from the processing to create B-tree indexes and text indexes.

If the executing user is deleted or the user's privileges are revoked by mistake, take either of the following corrective actions:

- As the same user, make the `adbimport` command executable and re-execute this command. For details about the corrective action to take, see the topic *Managing HADB users* or *Managing access privileges* in *Unscheduled Operations* in the *HADB Setup and Operation Guide*.
- As a different user who has the `REBUILD INDEX` privilege for the table to be processed, execute the `adbidxrebuild` command with the `--force` option specified. For details about how to execute the `adbidxrebuild` command with the `--force` option specified after cancellation of the `adbimport` command, see (2) Executing the adbidxrebuild command with the --force option specified in 17.10.7 If an error occurs during re-execution of the adbimport command.

18 adbinfoget (Collect Troubleshooting Information)

This chapter explains the functionality provided by the `adbinfoget` command and how to use the command.

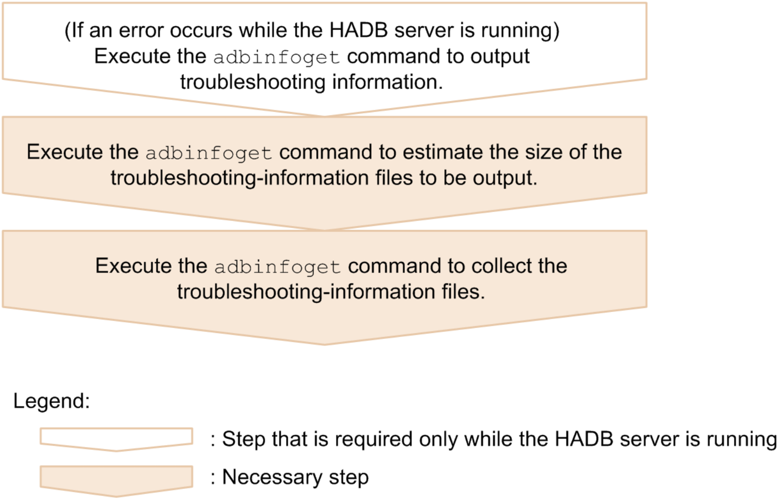
18.1 Description

The adbinfoget command collects troubleshooting-information files that can be used to determine the cause of an error. The command also outputs troubleshooting information and estimates the size of the troubleshooting-information files that will be output.

> **Important**
>
> When the multi-node function is being used, the adbinfoget command outputs troubleshooting information and estimates the size of and collects troubleshooting-information files on only the node where the command is executed. This means that when a failure occurs while the multi-node function is being used, you must execute the adbinfoget command on all nodes to collect troubleshooting-information files.

Execute this command to collect troubleshooting-information files whenever *Contact the customer support center* is given as the action to take in response to a message output during HADB server operation. The following figure shows the procedure for using the adbinfoget command to collect troubleshooting-information files.

Figure 18-1: Using the adbinfoget command to collect troubleshooting-information files


(If an error occurs while the HADB server is running)
Execute the adbinfoget command to output troubleshooting information.
Execute the adbinfoget command to estimate the size of the troubleshooting-information files to be output.
Execute the adbinfoget command to collect the troubleshooting-information files.
Legend:
: Step that is required only while the HADB server is running
: Necessary step

- **Outputting troubleshooting information (while the HABD server is running)**

  The following troubleshooting information is output if the HADB server terminates abnormally:

  - Core files
  - HADB dumps

  The preceding troubleshooting information is not output if an error occurs while the HADB server is running. Examples of such errors include an application program or command that does not terminate and an HADB server that has become unresponsive. Therefore, if you collect a troubleshooting-information file immediately after an error occurred, you cannot collect necessary information.

  If an error occurs while the HADB server is running, you must first output the preceding troubleshooting information. Then, perform the following procedure.

Note

Core files are output as troubleshooting information (detailed version). Output of core files might require a long time. Therefore, if you know that the detailed version of troubleshooting information will not be collected, only output HADB dumps.

- **Estimating the size of the troubleshooting-information files to be output**

  The `adbinfoget` command estimates the size of the troubleshooting-information files that will be output. A considerable amount of troubleshooting-information might be output, because it includes information about shared memory. Therefore, check its potential size before you collect the information. The required troubleshooting-information files will not be collected if there is insufficient disk space. Additionally, the performance of the HADB server might be adversely affected if a shortage of disk space occurs.

  The command provides the maximum amount of disk space that might be required for collecting the troubleshooting-information files. This maximum amount of space might not end up corresponding to the size of the troubleshooting-information files that are actually output.

- **Collecting the troubleshooting-information files**

  There are the following four types of troubleshooting-information files. *YYYYMMDDhhmmss* in each file name indicates the time when the `adbinfoget` command was executed.

  - Troubleshooting-information file (mail version) (`adbinfo`*YYYYMMDDhhmmss*`_mail.tar.gz`)

    This file stores the minimum amount of information required for troubleshooting.

  - Troubleshooting-information file (light version) (`adbinfo`*YYYYMMDDhhmmss*`_light.tar.gz`)

    This file stores troubleshooting information, but excludes some information, such as core files and shared memory dump files.

  - Troubleshooting-information file (detailed version) (`adbinfo`*YYYYMMDDhhmmss*`_detail.tar.gz`)

    This file stores all information that can be collected by the `adbinfoget` command.

  - Troubleshooting-information file (root version) (`adbinfo`*YYYYMMDDhhmmss*`_root.tar.gz`)

    This file stores the additional information that can be collected by the `adbinfoget` command when it is executed with the `-r` option by a superuser.

  Typically, you will collect all four types of troubleshooting-information files, the mail, light, detailed, and root versions. If collection of troubleshooting-information files is expected to take a long time because of the estimated output size of the detailed version, you might want to collect only the mail, light, and root versions. Always collect the root version of the troubleshooting-information files.

  If the `adbinfoget` command is executed after the HADB server has terminated abnormally, the command collects troubleshooting information for the HADB server at the point when the failure occurred and collects OS status information as of the time of the command's execution. If the HADB server is running or has been terminated, the command collects the troubleshooting information for the HADB server and the OS status information as of the time of the command's execution.

18.2 Specification format for the adbinfoget command

This section explains the specification format and options for the `adbinfoget` command.

The only OS users who can execute the `adbinfoget` command are HADB administrators.

18.2.1 Explanation of the specification format and options

(1) Specification format

```
adbinfoget {-g | -m | -o troubleshooting-information-files-output-directory}
           [-f core-file-name] [-l] [-r]
```

(2) Purpose of executing the adbinfoget command and combination of options

The following table shows the options to be specified depending on the purpose of executing the `adbinfoget` command.

For details about each option, see (3) Explanation of options.

Table 18-1: Options to be specified in the adbinfoget command

No.	Purpose of executing the adbinfoget command		Option to be specified
1	Outputting troubleshooting information while the HABD server is running	Output information for troubleshooting-information file (detailed version) (and output core files and HADB dumps).	`-g`
2		Output information for the mail, light, and root versions of troubleshooting-information files (and output HADB dumps only).	`-g` and `-l`
3	Estimating the output size of the troubleshooting-information files	Estimate the output size of the light, detailed and root versions of the troubleshooting-information files.#1	`-m` and `-r`
4		Estimate the output size of the light, detailed, and root versions of the troubleshooting-information files (when the `-f` option is specified for collecting the troubleshooting-information files).	`-m`, `-f`, and `-r`
5	Collecting troubleshooting-information files	Collect the mail, light, and detailed versions of the troubleshooting-information files (when the core files to be collected are not specified#2).	`-o` and `-r`
6		Collect the mail, light, detailed, and root versions of the troubleshooting-information files (and collect the core file specified in the `-f` option).	`-o`, `-f`, and `-r`
7		Collect only the mail, light, and root versions of the troubleshooting-information files.	`-o`, `-l`, and `-r`

#1

If the `adb_core_path` operand in the server definition is not specified, a value that includes the size of the latest core file located under the `$DBDIR/SPOOL` directory is estimated.

If the adb_core_path operand in the server definition is specified, a value that includes the size of the latest error information (core file) located under the directory specified in the adb_core_path operand in the server definition is estimated.

#2

If the adb_core_path operand in the server definition is not specified, the latest core file located under the $DBDIR/SPOOL directory is collected.

If the adb_core_path operand in the server definition is specified, the latest core file located under the directory specified in the adb_core_path operand in the server definition is collected.

(3) Explanation of options

- -g

Specify this option when outputting troubleshooting information shown in the following table while the HABD server is running. The information to be output is determined depending on whether the -l option is specified with this option.

Table 18-2: Troubleshooting information to be output

No.	Troubleshooting information to be output	If the -l option is not specified	If the -l option is specified	Output directory
1	Core files	Yes	No	$DBDIR/SPOOL directory, or the directory specified in the adb_core_path operand in the server definition
2	HADB dumps	Yes	Yes	$ADBDIR/spool directory

Legend:

Yes: Information is output.

No: Information is not output.

HADB server operation is paused while information is being output.

Note

You can specify this option only while the HADB server is running.

- -m

Specify this option when estimating the output size of a troubleshooting-information file. If you specify this option, the troubleshooting-information file is not collected.

Instead, the command estimates the sizes of the light and detailed versions of the troubleshooting-information files that will be output.

The mail version of the troubleshooting-information files is generated by extracting part of the light version of the troubleshooting-information files. Consequently, the estimated size of the mail version of the troubleshooting-information files is included in the estimated size of the light version of the troubleshooting-information files. Therefore, even when this option is specified, the command does not estimate the size of the mail version of the troubleshooting-information files alone.

- -o *troubleshooting-information-files-output-directory*

~<path name>((2 to 118 bytes))

Specifies the absolute path name of the directory to which the troubleshooting-information files are to be output.

Write and execution privileges must have been granted in advance to the HADB administrator for the directory specified in this option.

- `-f` *core-file-name*

  ~<path name>((2 to 510 bytes))

  Normally, you will omit this option.

  Specifies the absolute path of a core file in the following cases:

  1. The `adbstart` was executed after the `adbinit` command was executed, but the HADB server's start processing failed.
  2. The `adbstart` command was executed after the HADB server was replaced, but the HADB server's start processing failed.
  3. You wish to collect as troubleshooting information a core file containing specific error information from among all the error information output when the HADB server terminated abnormally
  4. The `kernel.core_pattern` kernel parameter is specified, but no error information (core file) was output under the `$DBDIR/SPOOL` directory or the directory specified in the server definition's `adb_core_path` operand.

  For cases 1 and 2 above, specify the absolute path of the following error information (core file), as appropriate:

  - If the `adb_core_path` operand is omitted in the server definition

    Error information (core file) under the `$DBDIR/SPOOL` directory

  - If the `adb_core_path` operand is specified in the server definition

    Error information (core file) under the directory specified in the `adb_core_path` operand in the server definition

  For case 4, specify the absolute path of the error information (core file) under the directory specified in the `kernel.core_pattern` kernel parameter.

  If this option is omitted, one of the following core files containing error information is collected as troubleshooting information:

  - If the `adb_core_path` operand is omitted in the server definition

    The core file under the `$DBDIR/SPOOL` directory that contains the most recent error information

  - If the `adb_core_path` operand is specified in the server definition

    The core file under the directory specified in the `adb_core_path` operand in the server definition that contains the most recent error information

- `-l`

  Specify this option in the following cases:

  - When outputting troubleshooting information (with the `-g` option specified together)

    If core files need not be output, specify this option. If you specify this option together with the `-g` option, a core file is not output (only HADB dumps are output).

    Because a core file is very large, the mail, light, and root versions of a troubleshooting-information file do not include a core file. Therefore, if you know that the detailed version of troubleshooting information will not be collected, specify this option together with the `-g` option. For details about troubleshooting information that is output when the `-g` option is specified, see Table 18-2: Troubleshooting information to be output.

    If you omit this option and specify the `-g` option, core files and HADB dumps are output.

  - When collecting troubleshooting-information files (without the `-g` option specified)

    Specifies that only the mail and light versions of the troubleshooting-information files are to be collected. You specify this option when the detailed version of the troubleshooting-information files is too large to collect.

    When this option is omitted, the command collects the mail, light, and detailed versions of the troubleshooting-information files.

Because a core file is very large, the mail, light, and root versions do not include a core file. Therefore, if the -l option is specified together with the -f option, the core file specified in the -f option will not be collected.

- -r

  Specify this option in any of the following cases:

  - You need to estimate the size of the root version of troubleshooting information that will be output, in addition to the light and detailed versions of troubleshooting information that will be output.
  - You need to acquire the root version of the troubleshooting-information files in addition to the mail, light, and detailed versions of troubleshooting information.
  - You need to collect the root version of the troubleshooting-information files in addition to the mail and light versions of troubleshooting information.

  When you execute the adbinfoget command with this option specified, the OS's su command is executed by the adbinfoget command. It then collects the root version of the troubleshooting-information files by executing OS commands that require root privileges, and by referencing system files that require root privileges. Therefore, when you execute the adbinfoget command with this option specified, enter a superuser password when requested.

  If execution of the OS's su command fails, the output size of all troubleshooting-information files (the detailed, light, mail, and root versions) is not estimated. In addition, none of the troubleshooting-information files is collected.

18.3 List of troubleshooting information items that are stored

The following table lists the troubleshooting information items that are stored in the mail version, the light version, the detailed version and the root version troubleshooting-information files.

Table 18-3: Troubleshooting information that is stored

No.	Troubleshooting information		Mail version	Light version	Detailed version	root version
	Classification	Description				
1	Information that is collected by executing HADB commands	HADB server status	Y	Y	Y	N
2		Connection status	Y	Y	Y	N
3		Thread status	Y	Y	Y	N
4		Pseudo-thread information	Y	Y	Y	N
5		Error module trace information	Y	Y	Y	N
6		Memory usage status	Y	Y	Y	N
7		Shared memory information	Y	Y	Y	N
8		Global buffer information	Y	Y	Y	N
9		Local work table buffer information	Y	Y	Y	N
10		HADB server's version information	Y	Y	Y	N
11		Locked resource allocation request status	Y	Y	Y	N
12		Node information	Y	Y	Y	N
13		Client group and command group information	Y	Y	Y	N
14		Shared memory dumps	N	N	Y	N
15		Buffer trace	N	N	Y	N
16		HADB dumps	P	Y	Y	N
17		`adbinfoget` command's logs	Y	Y	Y	Y
18		Status of the updated-row columnizing facility	Y	Y	Y	N
19	Definition and output files that are referenced by the HADB server	All directories and files under `$ADBDIR/conf`	Y	Y	Y	N
20		Files under `$ADBDIR/spool`	P	P	Y	N
21		Files under the `$DBDIR` directory or core files under the directory specified in the `adb_core_path` server definition operand	P	P	Y	N
22		Statistics log file	P	P	Y	N
23		All directories and files under `$ADBCLTDIR`#`/conf`	Y	Y	Y	N
24		Files under `$ADBCLTDIR`#`/spool`	P	P	Y	N
25		SQL trace file	P	P	Y	N
26		Access path search information log file	N	Y	Y	N
27		List information of audit trail files	Y	Y	Y	N

No.	Troubleshooting information		Mail version	Light version	Detailed version	root version
	Classification	Description				
		- List of files under the directory specified for the `adb_audit_log_path` operand				
28		HADB library information	Y	Y	Y	N
29		Synonym dictionary information - List of files under the directory specified for the `adb_syndict_storage_path` operand - List of files under the directory specified for the `adb_syndict_node_storage_path` operand	Y	Y	Y	N
30	Information that is collected by executing OS commands	CPU usage rate	Y	Y	Y	N
31		Operational status of CPU by process	Y	Y	Y	N
32		Process information	Y	Y	Y	N
33		Memory status	Y	Y	Y	N
34		Shared memory status	Y	Y	Y	N
35		Disk information	Y	Y	Y	N
36		Disk status	Y	Y	Y	N
37		Network status information	Y	Y	Y	N
38		Network information	Y	Y	Y	N
39		Kernel parameters	Y	Y	Y	N
40		System information	Y	Y	Y	N
41		Package information	Y	Y	Y	N
42		Environment variable information	Y	Y	Y	N
43		Host name	Y	Y	Y	N
44		Backtrace information	Y	Y	Y	N
45		Kernel logs	Y	Y	Y	N
46		System resource limitation information	Y	Y	Y	N
47		`libc.so` version information	Y	Y	Y	N
48		Communication information	Y	Y	Y	N
49		User information	Y	Y	Y	N
50		Device configuration information	Y	Y	Y	N
51		LVM configuration information	N	N	N	Y
52		VG information	N	N	N	Y
53	Definition and output files that are referenced by the OS	Kernel information	Y	Y	Y	N
54		Processor information	Y	Y	Y	N
55		Memory information	Y	Y	Y	N
56		Disk information	Y	Y	Y	N

No.	Troubleshooting information		Mail version	Light version	Detailed version	root version
	Classification	Description				
57		Map information	Y	Y	Y	N
58		`ulimit` information	Y	Y	Y	N
59		OS information	Y	Y	Y	N
60		syslog	Y	Y	Y	Y

Legend:

Y: Stored.

P: Part of the information is stored.

N: Not stored.

#

The directory path specified in the `ADBCLTDIR` environment variable for the server machine. If the `ADBCLTDIR` environment variable has not been specified, no information is stored.

18.4 Examples

This section explains how to use the `adbinfoget` command.

18.4.1 Collecting the mail, light, detailed and root versions of the troubleshooting-information files

To collect the mail, light, detailed and root versions of the troubleshooting-information files:

1. Estimate the size of the troubleshooting-information files to be output

```
adbinfoget -m -r

KFAA91401-I adbinfoget processing started.
KFAA91404-I The troubleshooting information (detail) output by the adbinfoget comm
and is 38737721219 bytes.
KFAA91404-I The troubleshooting information (light, mail) output by the adbinfoget
 command is 9053448 bytes.
Password:#
KFAA91404-I The troubleshooting information (root) output by the adbinfoget comman
d is 7995597 bytes.
KFAA91402-I adbinfoget processing ended. (return code = 0)
```

The underscored area shows the size of the following troubleshooting-information files that will be output:

- The detailed version of the troubleshooting-information files
- The mail and light versions of the troubleshooting-information files
- The root version of the troubleshooting-information files

#

When `Password:` is displayed, enter a superuser password.

The `Password:` display varies depending on the locale.

2. Collect the troubleshooting-information files

If there is enough disk space available based on the size estimate obtained in step 1, you can collect the troubleshooting-information files.

```
adbinfoget -o /home/adbmanager -r

KFAA91401-I adbinfoget processing started.
KFAA91406-I Creation of a light-version troubleshooting information file started.
KFAA91407-I Creation of a light-version troubleshooting information file is comple
te.
KFAA91408-I Creation of a detail-version troubleshooting information file started.
KFAA91409-I Creation of a detail-version troubleshooting information file is compl
ete.
Password:#
KFAA91410-I Creation of a root-version troubleshooting information file started.
KFAA91411-I Creation of a root-version troubleshooting information file is complet
e.
KFAA91402-I adbinfoget processing ended. (return code = 0)
```

#

When `Password:` is displayed, enter a superuser password.

The `Password:` display varies depending on the locale.

18.4.2 Collecting the mail, light and root versions of the troubleshooting-information files

To collect the mail, light and root versions of the troubleshooting-information files:

1. Estimate the size of the troubleshooting-information files to be output

```
adbinfoget -m -r

KFAA91401-I adbinfoget processing started.
KFAA91404-I The troubleshooting information (detail) output by the adbinfoget comm
and is 38737721219 bytes.
KFAA91404-I The troubleshooting information (light, mail) output by the adbinfoget
 command is 9053448 bytes.
Password:#
KFAA91404-I The troubleshooting information (root) output by the adbinfoget comman
d is 7995597 bytes.
KFAA91402-I adbinfoget processing ended. (return code = 0)
```

The underscored area shows the size of the following troubleshooting-information files that will be output:

- The mail and light versions of the troubleshooting-information files
- The root version of the troubleshooting-information files

#

When `Password:` is displayed, enter a superuser password.

The `Password:` display varies depending on the locale.

2. Collect the troubleshooting-information files

If there is enough disk space available based on the size estimate obtained in step 1, you can collect the troubleshooting-information files.

```
adbinfoget -o /home/adbmanager -l -r

KFAA91401-I adbinfoget processing started.
KFAA91406-I Creation of a light-version troubleshooting information file started.
KFAA91407-I Creation of a light-version troubleshooting information file is comple
te.
Password:#
KFAA91410-I Creation of a root-version troubleshooting information file started.
KFAA91411-I Creation of a root-version troubleshooting information file is complet
e.
KFAA91402-I adbinfoget processing ended. (return code = 0)
```

#

When `Password:` is displayed, enter a superuser password.

The `Password:` display differs depending on the locale.

18.5 Return codes

The following table lists and describes the return codes of the `adbinfoget` command.

Table 18-4: Return codes of the adbinfoget command

Return code	Description
`0`	The `adbinfoget` command terminated normally.
`4`	An error occurred during execution of the `adbinfoget` command, but troubleshooting-information files were collected. For the cause of the error, see the displayed message or the message output to a message log file.
`8`	An error occurred during execution of the `adbinfoget` command. Check the displayed message or the message output to a message log file and take appropriate action.

18.6 Notes

1. You must have specified the `ADBDIR` environment variable before you can execute the `adbinfoget` command.
2. The `adbinfoget` command can be executed regardless of the status of the HADB server. However, to execute the command with the `-g` option specified, make sure that the HADB server is running.
3. If the `adbinfoget` command terminates abnormally, a work directory (`adbinfo`*XXXXXXXXXX*, where *XXXXXXX* is the date and time when the `adbinfoget` command was executed) might remain under the troubleshooting-information files output directory. If this occurs, use the OS's `rm` command to delete the work directory.
4. OS commands are executed as an extension of the `adbinfoget` command. If execution of such an OS command fails, an error message is output to the standard error output. In such a case, check the error message, eliminate the cause of the error, and then re-execute the `adbinfoget` command.
5. HADB commands (such as `adbls` commands) are executed as an extension of the `adbinfoget` command. If execution of such an HADB command fails, an error message is output to the standard error output or to a message log file. In such a case, check the error message, eliminate the cause of the error, and then re-execute the `adbinfoget` command.
6. Do not execute the `adbinfoget` command while the `adbinit` command is executing. If the `adbinfoget` command is executed at such a time, any command that is executed as an extension of the `adbinit` or `adbinfoget` command might terminate abnormally, or invalid troubleshooting information might be collected.
7. Depending on the information specified in the server definition, it might take a very long time to collect the detailed version of the troubleshooting-information files. To cancel collection of troubleshooting-information files that is underway, terminate the `adbinfoget` command's process by, for example, pressing the **Ctrl**+**C** keys or using the OS's `kill` command. You can then execute the `adbinfoget` command with the `-l` option specified to collect only the mail and light versions.

19 adbinfosweep (Delete Troubleshooting Information)

This chapter explains the functionality provided by the `adbinfosweep` command and how to use the command.

19.1 Description

The `adbinfosweep` command deletes troubleshooting information output by the HADB server.

When the multi-node function is being used, only the troubleshooting information output by the HADB server on the node where the `adbinfosweep` command is executed is deleted.

The following troubleshooting information can be deleted:

- Core files containing error information (`$DBDIR/SPOOL/core.*` or *path-specified-in-the-adb_core_path-operand-in-the-server-definition*`/core.*`)
- Server definition storage file (`$ADBDIR/spool/.defrslt`)
- HADB dump files (`$ADBDIR/spool/adbdump`*YYYYMMDDhhmmss*`.`*server-process-process-id*#1 and `$ADBDIR/spool/adbdumperror`*YYYYMMDDhhmmssSSSSSS_TTTTTTTTTTTTTTTTTTTT*`.`*server-process-process-id*#2)

#1

YYYYMMDDhhmmss in a file name indicates the time the file was created.

#2

YYYYMMDDhhmmssSSSSSS in a file name indicates the time the file was created.

TTTTTTTTTTTTTTTTTTTT in the file name indicates the thread ID of the real thread resulting in an internal conflict error.

19.2 Specification format for the adbinfosweep command

This section explains the specification format for the `adbinfosweep` command.

```
adbinfosweep
```

The only OS users who can execute the `adbinfosweep` command are HADB administrators.

19.3 Return codes

The following table lists and describes the return codes of the `adbinfosweep` command.

Table 19-1: Return codes of the adbinfosweep command

Return code	Description
`0`	The `adbinfosweep` command terminated normally.
`8`	An error occurred during execution of the `adbinfosweep` command. Check the displayed message or the message output to a message log file and take appropriate action.

19.4 Notes

1. You must have specified the ADBDIR environment variable before you can execute the adbinfosweep command.
2. Execute the adbinfosweep command after the HADB server has terminated.
3. OS commands are executed as an extension of the adbinfosweep command. If execution of such an OS command fails, an error message is output to the standard error output. In such a case, check the error message, eliminate the cause of the error, and then re-execute the adbinfosweep command.
4. HADB commands are executed as an extension of the adbinfosweep command. If execution of such an HADB command fails, an error message is output to the standard error output or to a message log file. In such a case, check the error message, eliminate the cause of the error, and then re-execute the adbinfosweep command.
5. Do not execute the adbinfosweep command while the adbinit command is executing. If the adbinfosweep command is executed in such a case, any command that is executed as an extension of the adbinit or adbinfosweep command might terminate abnormally.
6. Do not execute the adbinfosweep command while the HADB server is starting or terminating. If the adbinfosweep command is executed at such a time, the HADB server or the adbinfosweep command might terminate abnormally.
7. Only one adbinfosweep command can be executed at a time. If multiple adbinfosweep commands are executed concurrently, one or more of the commands might terminate abnormally.

20 adbinit (Initialize the Database)

This chapter explains the functionality provided by the `adbinit` command and how to use the command.

20.1 Description

You execute the adbinit command when you initialize HADB. The DB areas shown in the following table are created when the adbinit command is executed.

Important

- If the adbinit command is executed when DB areas already exist in the DB directory, all the DB areas (including DB areas other than data DB areas) are deleted. Then, the DB areas shown in the following table are created.
- If you want to add or change a specific DB area, use the adbmodarea command rather than the adbinit command. For details about the adbmodarea command, see 34. adbmodarea (Add and Change DB Areas).

Table 20-1: DB areas created when the adbinit command is executed

No.	DB area that is created	Description
1	Master directory DB area	Created automatically when the adbinit command is executed.
2	Dictionary DB area	
3	Work table DB area	
4	System-table DB area	
5	Data DB areas	The adbinitdbarea operand of the initialization options is used to define the data DB areas that are to be created.

The DB areas listed above can be created as regular files under the DB directory or as block special files. We recommend that you allocate block special files. You use initialization options of the adbinit command to specify which files are to be created.

20.2 Specification format for the adbinit command

This section explains the specification format for the `adbinit` command and the format of the initialization options.

The only OS users who can execute the `adbinit` command are HADB administrators.

20.2.1 Explanation of the specification format and options

(1) Specification format

```
adbinit
   -u authorization-identifier
  [-p password]
  initialization-option-file-name
  DB-directory-name
```

(2) Explanation of options

- `-u` *authorization-identifier*

  ~<character string>((1 to 100 bytes))

  Specifies an authorization identifier for the first HADB user that is to be created when the HADB server is initialized. The `DBA` and `CONNECT` privileges are granted automatically to this first HADB user.

  For this option, specify an authorization identifier in the range from 1 to 100 bytes. Note that the byte count (1 to 100 bytes) does not include the double quotation marks used to enclose the authorization identifier.

  The following rules apply to specifying this information:

  - The characters permitted for the authorization identifier include single-byte uppercase and lowercase letters, single-byte numeric characters, backslash (`\`), hash mark (`#`), and at mark (`@`).
  - If the character string used as the authorization identifier includes a lowercase letter, enclose the entire authorization identifier in backslashes and double quotation marks (`\"`). Alternatively, enclose the entire authorization identifier in double quotation marks (`"`) and then enclose it in single quotation marks (`'`).

    Example 1: `-u \"ADBuser01\"`

    Example 2: `-u '"ADBuser01"'`

    If the authorization identifier is not specified as shown above, the character string used as the authorization identifier will be treated as being in all uppercase letters. For example, `-u ADBuser01` will be treated as `-u ADBUSER01`.
  - If the character string used as the authorization identifier contains a backslash (`\`), specify the escape character (`\`) immediately before the backslash. Alternatively, enclose the entire authorization identifier in single quotation marks (`'`).

    Example 1: `-u ADBUSER\\01`

    Example 2: `-u 'ADBUSER\01'`

    In this example, `ADBUSER\01` is specified as the authorization identifier.

  For details about the rules for specifying authorization identifiers, see *Authorization identifier specification rules* in the *HADB Setup and Operation Guide*.

Note

After initialization, use of the authorization identifier and password specified in the `adbinit` command will be required to connect to the HADB server.

- `-p` *password*

  ~<character string>((1 to 255 bytes))

  Specifies a password for the first HADB user that is to be created when the HADB server is initialized.

  The following rules apply to specifying this information:

  - The characters permitted for the password include single-byte uppercase and lowercase letters, single-byte numeric characters, and the following single-byte symbols:

    ```
    \ @ ` ! " # $ % & ' ( ) * : + ; [ ] { } , = < > | - . ^ ~ / ? _
    ```

  - Unlike the authorization identifier specified in the `-u` option, the password is case sensitive.
  - If the password contains a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), specify the escape character (`\`) immediately before that character. Alternatively, enclose the entire password in single quotation marks (`'`).

    Example 1: `-p Password\|01`

    Example 2: `-p 'Password|01'`

    In this example, `Password|01` is specified as the password.

  For details about the rules for specifying passwords, see *Password specification rules* in the *HADB Setup and Operation Guide*.

  If the `adbinit` command is executed with this option omitted, a message prompting the user to enter a password will be displayed. In such a case, enter the password from the standard input.

Note

You can change the password entered here with the `ALTER USER` statement. For details about how to change passwords, see *Changing an HADB user's password* in the *HADB Setup and Operation Guide*.

Important

How you specify a character string containing a lowercase letter differs between the `-u` and `-p` options.

The following shows specification examples:

Example 1: Specifying `ADBUSER01` as the authorization identifier and `PASSWORD01` as the password

```
adbinit -u ADBUSER01 -p PASSWORD01
```

Example 2: Specifying `ADBuser01` as the authorization identifier and `Password01` as the password

```
adbinit -u \"ADBuser01\" -p Password01
```

Example 3: Specifying `ADBuser\01` as the authorization identifier and `Password\01` as the password

```
adbinit -u \"ADBuser\\01\" -p Password\\01
```

- *initialization-option-file-name*

  ~<OS path name>((1 to 1,023 bytes))

Specifies the absolute path name or relative path name of the file that contains the initialization options. For details about the initialization options, see 20.2.2 Format of initialization options.

- *DB-directory-name*

  ~<path name>((2 to510 bytes))

  Specifies the absolute path name of the DB directory. Specify the DB directory name specified in the `adb_db_path` operand in the server definition.

  When you execute the `adbinit` command, the command creates the file entities of or symbolic links to the DB area files under the DB directory specified here.

  The following notes apply:

  - Only one DB directory can be created.
  - If an existing DB directory is specified, files in the DB directory that are to be overwritten are first deleted and then re-created.
  - Do not create the DB directory under the server directory. The root directory cannot be used as the DB directory.

20.2.2 Format of initialization options

(1) Specification format

```
[set adb_init_dbarea_initialize = {Y|N}]
[set adb_init_multi_max = maximum-number-of-threads-used-to-create-DB-area-files]
[set adb_init_wrk_page_size = page-size-of-work-table-DB-area]
[set adb_init_wrk_blk_path = block-special-file-name-for-allocating-work-table-DB-are
a-file]
[set adb_init_mst_blk_path = block-special-file-name-for-allocating-master-directory-
DB-area-file]
[set adb_init_dic_blk_path = block-special-file-name-for-allocating-dictionary-DB-are
a-file]
[set adb_init_stbl_blk_path = block-special-file-name-for-allocating-system-table-DB-
area-file]

{{adbinitdbarea
     -n data-DB-area-name
    [-p {4|8|16|32}]
    [-v block-special-file-name[,block-special-file-name]...]
    [-q]
    [-i initial-allocation-size[,number-of-files-to-be-created]]
    [-f data-DB-area-file-storage-directory-name]
    [-g output-interval-for-progress-messages-for-DB-area-file-creation-processing]
 }}
```

The specification rules for the foregoing initialization options are the same as for the server definitions. For details about the server definition specification files, see the topic *Syntax rules for the server definition* in the *HADB Setup and Operation Guide*.

(2) Explanation of specification format

(a) Set format

- adb_init_dbarea_initialize = {Y|N}

  Specifies whether the DB areas are to be initialized. When N is specified in this operand, block special files must be allocated to all DB area files other than in the work table DB area.

  Y:

  Initializes all DB areas.

  If specification of this operand is omitted, Y is assumed.

  N:

  Creates only symbolic links without initializing the DB areas. Note that only the work table DB area is initialized.

  When N is specified in this operand, the following option values are ignored when DB area files are created:

  - **adbinit command options**
    - -u option
    - -p option
  - **Initialization options (operands in the set format)**

    adb_init_multi_max operand
  - **Initialization options (adbinitdbarea operand's options)**
    - -q option
    - -g option
    - -p option
    - Initial allocation size in the -i option

  Multi-node function

  - If you will be creating a DB directory for a slave node (executing the adbinit command on a slave node), specify N in this operand.
  - For details about how to create DB directories when the multi-node function is being used, see 20.3.2 Creating DB directories (when using the multi-node function).

- adb_init_multi_max = *maximum-number-of-threads-used-to-create-DB-area-files*

  ~<integer>((1 to 65,535))<<40>>

  Specifies the maximum number of threads to be used to create data DB area files.

  Specify this operand in the following cases:

  - **An error related to the number of threads or file open operations, or a memory shortage has occurred during execution of the adbinit command**

    If many data DB area files are created, the numbers of threads and descriptors might exceed the system limit, resulting during execution of the adbinit command in an error related to the number of threads or file open operations, or a memory shortage (output of the KFAA96211-E message). In such a case, you might be able to prevent such errors by executing the adbinit command with a small value specified in this operand.

    If 1 is specified in this operand, one data DB area file is created at a time.

  - **40 or more block special files are used**

    Performance might improve if you increase the maximum number of threads by specifying a larger value in this operand.

When data DB areas are created, one thread is generated for each data DB area and one thread is created for each data DB area file. For example, if you create DB area A (two DB area files) and DB area B (three DB area files), a total of seven threads are generated.

- adb_init_wrk_page_size = *page-size-of-work-table-DB-area*

  ~<integer>((32 to 32,768))<<256>> (kilobytes)

  Specifies the page size (in kilobytes) for the work table DB area. Specify the page size that you estimated when you designed the work table DB area. For details about estimating the page size of the work table DB area, see the topic *Designing a work table DB area* in the *HADB Setup and Operation Guide*.

  If you specify a value that is not divisible by 32, the specified value is rounded up to the next multiple of 32 kilobytes.

  The work table DB area is created according to the page size specified in this operand. Specifying the adb_dbarea_wrk_page_size operand in the server definition allows you to change the page size of the work table DB area when the HADB server starts. For details about the adb_dbarea_wrk_page_size operand, see *Operands related to system configuration (set format)* in the *HADB Setup and Operation Guide*.

- adb_init_wrk_blk_path = *block-special-file-name-for-allocating-work-table-DB-area-file*

  ~<path name>((2 to 255 bytes))

  When a block special file is to be allocated for a work table DB area file, this option specifies an absolute path name for the block special file, expressed as a file entity or a symbolic link. If this operand is omitted, work table DB area files are created under the DB directory.

  Make sure that the file name specified in this operand differs from the file names of the block special files specified in other operands.

  Important

  An improvement in performance can be expected when block special files are allocated instead of regular files.

  However, note that when block special files are allocated, any existing data in the block special files is deleted. If you are not sure whether existing data ought to be deleted, do not allocate block special files.

  A block special file mounted on the OS cannot be allocated. A block special file to be allocated must be unmounted.

- adb_init_mst_blk_path = *block-special-file-name-for-allocating-master-directory-DB-area-file*

  ~<path name>((2 to 255 bytes))

  When a block special file is to be allocated for the master directory DB area file, this option specifies an absolute path name for the block special file, expressed as a file entity or a symbolic link. If this operand is omitted, the master directory DB area file is created under the DB directory.

  Make sure that the file name specified in this operand differs from the file names of the block special files specified in other operands.

  For details about specifying block special files, see the notes for the adb_init_wrk_blk_path operand.

- adb_init_dic_blk_path = *block-special-file-name-for-allocating-dictionary-DB-area-file*

  ~<path name>((2 to 255 bytes))

  When a block special file is to be allocated for the dictionary DB area file, this option specifies an absolute path name for the block special file, expressed as a file entity or a symbolic link. If this operand is omitted, the dictionary DB area file is created under the DB directory.

  Make sure that the file name specified in this operand differs from the file names of the block special files specified in other operands.

  For details about specifying block special files, see the notes for the adb_init_wrk_blk_path operand.

- `adb_init_stbl_blk_path` = *block-special-file-name-for-allocating-system-table-DB-area-file*

  ~<path name>((2 to 255 bytes))

  When a block special file is to be allocated for the system-table DB area file, this option specifies an absolute path name for the block special file, expressed as a file entity or a symbolic link. If this operand is omitted, the system-table DB area file is created under the DB directory.

  For details about specifying block special files, see the notes for the `adb_init_wrk_blk_path` operand.

(b) Command format

- `adbinitdbarea`

  Specifies the definition of a data DB area. You specify this operand for each data DB area that is to be created.

  The options to be specified depend on the type of DB area files that are to be allocated for the data DB area. The following table provides guidelines for the options to be specified.

Table 20-2: Guidelines for the options to be specified depending on the type of files that are to be allocated as data DB area files

No.	Option name	Type of files allocated as DB area files		Description of option
		Block special file	Regular file	
1	`-n`	R	R	Specifies the name for the data DB area.
2	`-p`	R	R	Specifies the page size for the data DB area.
3	`-v`	R	N	Specifies names for the block special files that are to be allocated as the data DB area files.
4	`-q`	R	N	Specifies that the data DB area files are to be created in quick mode.
5	`-i`	R	R	Specifies the initial allocation size for each data DB area file and the number of data DB area files to be created.
6	`-f`	O	O	Specifies that symbolic links to the data DB area files or block special files are to be stored in a subdirectory of the DB directory.
7	`-g`	O	O	Specifies the interval for output of progress messages for data DB area file creation processing.

Legend:

R: Required/Recommended. Specification of this option is required or is recommended.

O: Decide whether to specify this option after reading its description.

N: Not specified.

`-n` *data-DB-area-name*

~<character string>((1 to 30 bytes))

Specifies a name for the data DB area that is to be created. The name specified in this option is used as the name of the data DB area files.

The following notes apply:

- The characters permitted for a DB area name are alphanumeric characters, underscore (_), and hyphen (-). A DB area name must begin with an alphabetic character.
- If a DB area name to be used is the same as a reserved word for SQL statements, enclose the DB area name in a backslash and double quotation mark (`\"`). For details about the reserved words for SQL statements, see *Reserved words* in *SQL Basics* in the manual *HADB SQL Reference*.

- If a specified DB area name contains a lowercase letter, enclose the entire DB area name in a backslash and double quotation mark (`\"`). If it is not enclosed in a backslash and double quotation mark, the character string used as the DB area name will be treated as being in all uppercase letters.

  Example: `\"DBarea01\"`

- This DB area name must differ from the name of any other DB area or directory created under the DB directory. For details about the names of other DB areas, see Table 20-5: Attributes of DB areas that are created in 20.5.1 Attributes of DB areas that are created. For details about the names of directories that are created under the DB directory, see Table 20-6: Directories that are created under the DB directory in 20.5.2 Directories that are created.

`-p {4|8|16|32}`

Specifies the page size (in kilobytes) for the data DB area. Specify the page size you estimated when you designed the data DB area. For details about how to estimate the page size for the data DB area, see *Points to consider when determining the page size in data DB areas* in the *HADB Setup and Operation Guide*.

`-v` *block-special-file-name*`[,`*block-special-file-name*`]...`

~<path name>((2 to 255 bytes))

Specify this option when you allocate block special files as data DB area files. This option specifies the absolute path names of the block special files, expressed as file entities or symbolic links.

The following notes apply:

- Each block special file name must differ from any block special file name specified in any other operand.
- For guidelines for the number of block special files (number of logical volumes that make up one DB area) and size to be specified in this option, see the following: *Points to consider when setting up an LVM* in the *HADB Setup and Operation Guide*.
- Specify as many block special file names as there are DB area files to be created (as specified in the `-i` option). If fewer block special file names are specified than the number of DB area files to be created, the `adbinit` command results in an error.
- For details about specifying block special files, see the notes for the `adb_init_wrk_blk_path` operand.

When you specify this option, the command creates symbolic links to the block special files under the DB directory.

`-q`

Specifies that the data DB area files are to be created in quick mode. Specifying this option enables you to reduce the time required to create the data DB area files.

The `-q` option is applied if the files to be initialized are block special files.

> **Note**
>
> When the data DB area files are created in quick mode, only each file's management page (containing information such as the file size) is created. The pages for storing actual data are not created. This is why it takes less time to create the data DB area files.

`-i` *initial-allocation-size*`[,`*number-of-files-to-be-created*`]`

Specifies the initial allocation size for each data DB area file and the number of data DB area files to be created.

- *initial-allocation-size*

  ~<alphabetic and numeric characters>((0K to 127P))<<0K>>

  Specifies the initial allocation size for each data DB area file, using one of the following letters to indicate the units: `K` for kilobytes, `M` for megabytes, `G` for gigabytes, `T` for terabytes, or `P` for petabytes. The size of a data DB area file is determined according to the specified initial allocation size. However, depending on the location of pages during initial allocation, the initial allocation size specified in this option might not be allocated for data DB area files.

The guidelines for the initial allocation size depend on the type of the data DB area file. Therefore, do not create different types of data DB area files at the same time. The following table shows the guidelines for the initial allocation size according to the types of data DB area file.

Table 20-3: Types of data DB area file and guidelines for the initial allocation size

No.	Type of data DB area file		Guidelines for the value of initial allocation size
1	Regular file		The value of *initial-allocation-size* multiplied by the *number-of-files-to-be-created* must be less than the value of *free-space-in-file-system-in-which-to-allocate-data-DB-area-files*. You can specify any value that meets this condition. When you determine the specific value, consider the disk space consumption and the time required for executing the `adbinit` command.
2	Block special file	Physical volume (normal block special file to which physical blocks are statically allocated)	If all physical volumes to be allocated have the same size, specify the size of one volume for *initial-allocation-size*. We do not recommend that physical volumes of different sizes are allocated at the same time. If you need to allocate physical volumes of different sizes at the same time, specify the size of the smallest volume.
3		Virtual volume (block special file with storage virtualization technology called thin provisioning# applied)	To use a virtual volume of less than 16 terabytes, specify `0K`. If a virtual volume might use 16 or more terabytes, specify the size you are planning to use. However, we do not recommend allocation of virtual volumes if the use of 16 or more terabytes is expected. If a size of 16 or more terabytes is specified, a large number of physical blocks are required from the beginning, which lacks the advantage of using lower cost virtual volumes.

#: For details about thin provisioning, see the manuals of storage products.

If the data being stored exceeds the initial allocation size of the data DB area file, the HADB server automatically expands the required area (that is, automatic extension of DB area is performed). The maximum value of the area that can be expanded depends on the initial allocation size. For details, see *DB area automatic extension* in the *HADB Setup and Operation Guide*.

Note

The size of the data DB area file might not be allocated as specified in the `-i` option. The following describes the reason.

A data DB area file consists of the following pages:

- Directory pages
- Segments (pages for storing table data and indexes)

When the size of the data DB area file is specified in the `-i` option, pages are located so that the file fits in the specified size. However, if the size specified in the `-i` option is reached in the middle of a directory page or segment, the number of pages is adjusted as follows:

- Consecutive directory pages are always created at the beginning of the data DB area file. Therefore, if a value smaller than the minimum size of the data DB area file (approximately 5.4 megabytes) is specified, the value is rounded up.
- Pages that make up segments are managed in units of a segment (that is, in units of the number of pages that make up a segment). Therefore, the number of pages are adjusted so

that the specified file size is not exceeded in the middle of a segment. For example, if the file size is exceeded in the middle of the eighth segment, the eighth segment is not created (only seven segments are created).

For details about directory pages and segments in data DB area files, see *Existence of directory page groups and locations of directory pages* in *Page types* in *Pages* in *DB area structure (segments and pages)* in the *HADB Setup and Operation Guide*.

- *number-of-files-to-be-created*
  ~<integer>((1 to 1,024))<<1>> (count)
  Specifies the number of data DB area files to be created.
  If the `-v` option is specified, make sure that *number-of-files-to-be-created* is the same as the number of block special files specified in the `-v` option.

The initial allocation size, depending on the `-i` option value, is explained in the following by way of an example.

Example:

```
adbinitdbarea -n DBAREA01 -i 2G,3
```

In this example, three data DB area files are to be created in the data DB area named `DBAREA01`, and two gigabytes of space are to be allocated as the initial allocation size for each data DB area file. The total initial allocation size for `DBAREA01` will be six gigabytes.

If the initial allocation size determined by the `-i` option value exceeds the maximum permissible size, the `adbinit` command will result in an error and the `KFAA96228-E` message will be issued. If this happens, change the `-i` option value according to the information provided in the message.

Important

If a large block special file exceeding the maximum value of automatic extension of the DB area file is allocated, an unusable area is generated in the block special file. As a result, the warning message `KFAA96244-W` is output when the `adbinit` command is executed. If you want to use areas in the block special file almost to the capacity limit, take the corrective action described for the `KFAA96244-W` message.

The maximum value of automatic extension of a DB area file is determined according to the specified initial allocation size. A multiple of 16 terabytes rounded up from the specified initial allocation size will be the maximum value of automatic extension of one DB area file. For example, if the specified initial allocation size is 20 terabytes, the maximum value of automatic extension of the DB area file is 32 terabytes.

Note that there is no way to make an unusable area usable.

To allocate small-capacity and large-capacity block special files to a single DB area, initialize the block special files by specifying a different initial allocation size for each of these files. The following shows an example.

Example: The following two block special files are allocated to DB area `AREA1`:

- `/dev/mapper/vg_10TB_1` (size: 10 terabytes)
- `/dev/mapper/vg_30TB_1` (size: 30 terabytes)

Procedure

1. Initialize the database by using the `adbinit` command. At this time, initialize the `/dev/mapper/vg_10TB_1` file with the initialization option specified as follows.

   ```
   adbinitdbarea -n AREA1 ... -i 10T -v /dev/mapper/vg_10TB_1
   ```

2. Start the HADB server by using the `adbstart` command.

3. Expand DB area `AREA1` by using the `adbmodarea` command. At this time, add the `/dev/mapper/vg_30TB_1` file with the DB area addition and modification option specified as follows.

   ```
   adbexpandarea -n AREA1 ... -i 30T -v /dev/mapper/vg_30TB_1
   ```

When the two block special files are concurrently initialized in step 1 above, a value exceeding 10 terabytes cannot be specified for the initial allocation size. Therefore, the maximum value of automatic extension of the DB area file is 16 terabytes. In this case, the maximum available space in the 30-terabyte block special file is only 16 terabytes, which is the maximum value of automatic extension of the DB area file.

`-f` *data-DB-area-file-storage-directory-name*

~<path name>((1 to 70 bytes))

Specifies the subdirectory that stores the data DB area files or a symbolic link to the block special files, expressed as a path name relative to the DB directory. A data DB area file with the DB area name as its name or a symbolic link to a block special file is stored directly under this subdirectory.

If you omit this option, the data DB area file names or the symbolic links to the block special files are stored directly under the DB directory.

The following shows specification examples.

Example:

```
adbinitdbarea -n DBAREA01 -f DB01/
```

This example creates a subdirectory named `DB01` directly under the DB directory and then creates a data DB area file with file name `DBAREA01` or symbolic link to the block special file under that subdirectory.

The following notes apply when you specify this option:

- There is no need to specify a forward slash (`/`) at the beginning of a relative path name.
- Specify a path name that begins with a single-byte alphabetic character, ends with a forward slash (`/`), and consists of only single-byte alphanumeric characters, the underscore (_), and the hyphen (`-`). If the path name does not end with a forward slash, a forward slash is added automatically at the end of the path name. The permitted maximum length of a path name is 69 bytes.
- If the name of the DB area file storage directory includes a lowercase letter, enclose the entire storage directory name in a backslash and double quotation mark (`\"`). If it is not enclosed in a backslash and double quotation mark, the character string used as the storage directory name will be treated as being in all uppercase letters.

  Example: \"Db02/\"
- No path under a directory that is created under the DB directory can be specified. For details about the directories that are created under the DB directory, see Table 20-6: Directories that are created under the DB directory in 20.5.2 Directories that are created.
- If the specified path contains an intermediate directory with the same name as the DB area name, directory or file creation might fail. For this reason, you must make sure that the directory in which each DB area file is created does not contain any intermediate directory with the same name as the DB area name.

`-g` *output-interval-for-progress-messages-for-DB-area-file-creation-processing*

~<integer>((0 to 100))<<0>> (%)

Specify this option when you wish to issue messages that report the progress of data DB area file creation processing.

This option specifies as a percentage the message output interval. For example, if 30 is specified, the command issues the `KFAA96232-I` message reporting the progress when completion of DB area creation processing reaches 30%, 60%, and 90%.

If you specify `0` or nothing in this option, no progress messages are output.

(3) Model initialization option

A model initialization option is provided (`$ADBDIR/sample/conf/adbinit.opt`). Use this model when you create initialization options.

If you use the sample application program, you need data DB areas (`ADBUTBL01` and `ADBUIDX01`). This model codes definitions for the data DB areas (specification of `adbinitdbarea` operands). Therefore, if you use the model, make sure that you do not delete the definitions of the data DB areas.

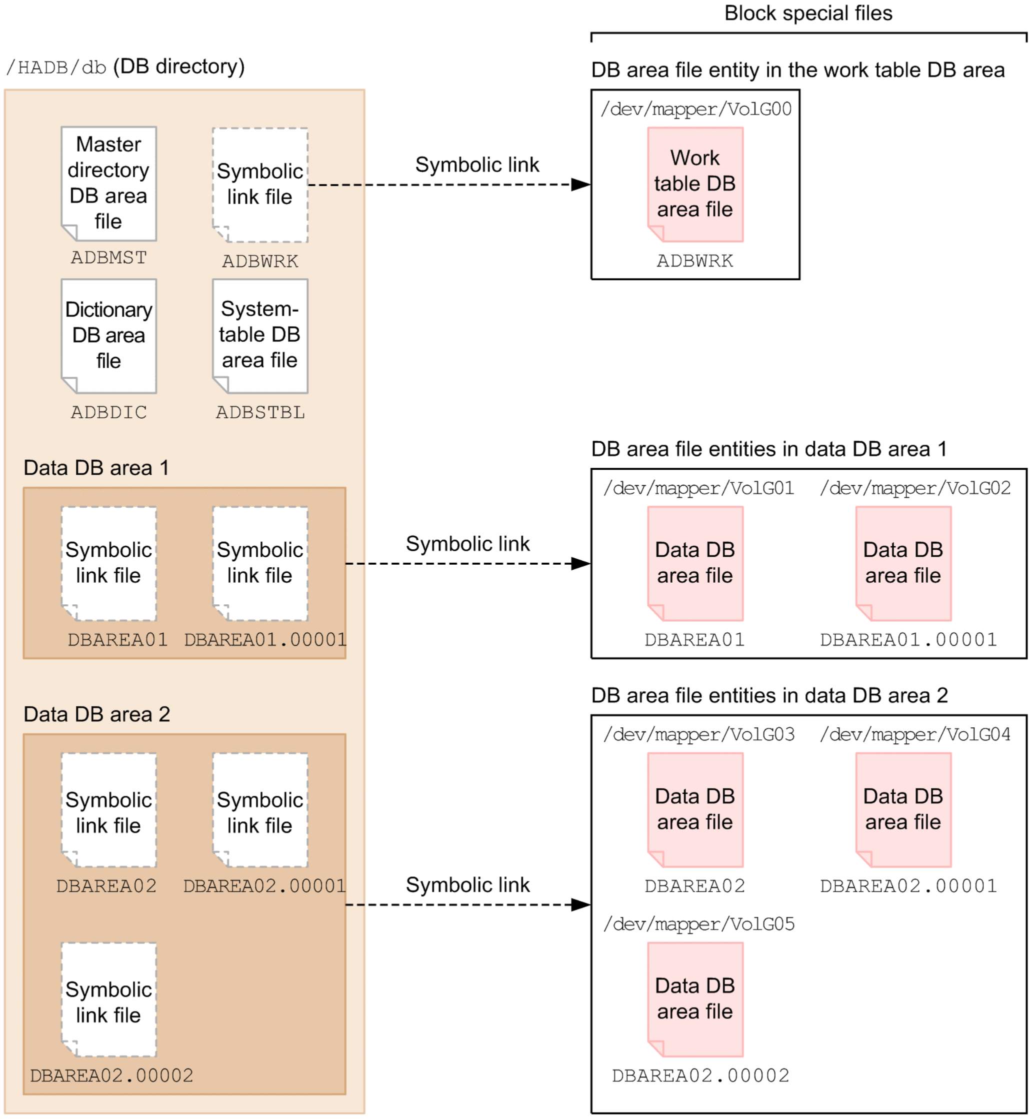
For details about the sample application program, see *Sample Application Program* in the *HADB Application Development Guide*.

20.3 Examples

This section explains by way of examples how to create a DB area by using the `adbinit` command.

20.3.1 Creating DB area files using block special files

The example explained in this section creates a DB area by executing the `adbinit` command. The command allocates block special files as data DB area files and work table DB area files, as shown in the following:



Explanation:

- The DB directory is `/HADB/db`.
- A block special file (`/dev/mapper/VolG00`) is allocated as a work table DB area file.
- Block special files (`/dev/mapper/VolG01` and `/dev/mapper/VolG02`) are allocated as data DB area files for data DB area 1.

  The following defines the DB area to be created:

  DB area name: `DBAREA01`

Page size of DB area: 4 kilobytes

Number of DB area files: 2

Initial allocation size for each DB area file: 500 megabytes

- Block special files (/dev/mapper/VolG03 through /dev/mapper/VolG05) are allocated as data DB area files for data DB area 2.

  The following defines the DB area to be created:

  DB area name: DBAREA02

  Page size of DB area: 4 kilobytes

  Number of DB area files: 3

  Initial allocation size for each DB area file: 1 gigabyte

This example assumes the existence of a symbolic link destination directory and block special files and that the required access permissions have been granted.

The example uses the following procedure to execute the adbinit command.

Procedure

1. Create an initialization option file.
2. Execute the adbinit command.

Using an example, the following subsections explain each of these steps.

(1) Creating an initialization option file

This example creates an initialization option file and specifies initialization options.

Example of an initialization option file

```
set adb_init_wrk_page_size = 32                                            ...1
set adb_init_wrk_blk_path = /dev/mapper/VolG00                             ...2

adbinitdbarea -n DBAREA01 \                                                ...3
              -p 4 \
              -v /dev/mapper/VolG01,/dev/mapper/VolG02 \
              -q \
              -i 500M,2

adbinitdbarea -n DBAREA02 \                                                ...4
              -p 4 \
              -v /dev/mapper/VolG03,/dev/mapper/VolG04,/dev/mapper/VolG05 \
              -q \
              -i 1G,3
```

Explanation:

1. Specifies the page size for the work table DB area.
2. Specifies the block special file (/dev/mapper/VolG00) that is to be allocated as a work table DB area file.
3. Specifies the definition of data DB area 1.

   -n: Specifies the data DB area name (DBAREA01).

   -p: Specifies the page size (4 kilobytes) of the data DB area.

   -v: Specifies the block special files (/dev/mapper/VolG01 and /dev/mapper/VolG02) that are to be allocated as data DB area files.

-q: Specifies that the data DB area files are to be created in quick mode.

-i: Specifies the initial allocation size (500 megabytes) of each data DB area file and the number of files (2) to be created.

4. Specifies the definition of data DB area 2.

   -n: Specifies the data DB area name (DBAREA02).

   -p: Specifies the page size (4 kilobytes) of the data DB area.

   -v: Specifies the block special files (/dev/mapper/VolG03 through /dev/mapper/VolG05) that are to be allocated as data DB area files.

   -q: Specifies that the data DB area files are to be created in quick mode.

   -i: Specifies the initial allocation size (1 gigabyte) of each data DB area file and the number of files (3) to be created.

(2) Executing the adbinit command

```
adbinit -u ADBUSER01                                  ...1
        -p '#HelloHADB_01'                            ...2
        /HADB/server/conf/adbinit.opt                 ...3
        /HADB/db                                      ...4
```

Explanation:

1. Specifies the authorization identifier of the first HADB user that is to be created during initialization.
2. Specifies a password for the authorization identifier specified in the -u option.
3. Specifies the name of the initialization option file created in (1) Creating an initialization option file.
4. Specifies the DB directory name.

20.3.2 Creating DB directories (when using the multi-node function)

When the multi-node function is being used, some of the DB areas are shared among all HADB servers in the multi-node configuration. For details about the multi-node function, see the topic *Multi-node function* in *Architecture* in the *HADB Setup and Operation Guide*. For details about the DB areas that are shared among all HADB servers in a multi-node configuration, see the topic *DB area file configuration* in *DB areas (areas for storing tables and indexes)* in *Architecture* in the *HADB Setup and Operation Guide*.

To share DB areas, the different systems must be able to uniquely identify a device allocated to DB area files. For this purpose, WWN is used. WWN is a fixed identifier that does not depend on the system and that is allocated to each device on the HBA.

If you will be using the multi-node function, when you create the DB directory, allocate to each DB area to be shared a block special file that includes WWN (path beginning with /dev/disk/by-id/wwn-).

When the multi-node function is being used, execute the adbinit command using the following procedure to create the DB directory.

Procedure

1. Create an initialization option file for the master node.
2. Execute the adbinit command on the master node.
3. Create an initialization option file for each slave node.

4. Execute the `adbinit` command on each slave node.

Perform steps 3 and 4 for each slave node on each HADB server.

Note that you can also specify block special files for work table DB areas in the server definition. For details, see the description of the `adb_blk_path_wrk` operand in *Operands related to system configuration (set format)* in *Detailed descriptions of the server definition operands* in *Designing the Server Definition* in the *HADB Setup and Operation Guide*.

Using an example, the following subsections explain each of these steps.

(1) Creating an initialization option file for the master node

Create an initialization option file and specify the initialization options as follows:

- Specify `Y` in the `adb_init_dbarea_initialize` operand because each DB area for the master node must be initialized.
- Allocate a block special file that includes a WWN identifier to each and every DB area that will be shared among all HADB servers in the multi-node configuration.
- If you want to allocate a block special file to a DB area (work table DB area) that will not be shared among the HADB servers in the multi-node configuration, allocate a block special file specific to the master node.

Example of an initialization option file for master node

```
set adb_init_dbarea_initialize = Y
set adb_init_wrk_blk_path = /dev/mapper/adb-ADBWRK_lv
set adb_init_mst_blk_path = /dev/disk/by-id/wwn-0x600605b003a2e280172ab596258093bc
set adb_init_dic_blk_path = /dev/disk/by-id/wwn-0x600605b003a2e280172ab5962580e259
set adb_init_stbl_blk_path = /dev/disk/by-id/wwn-0x600605b003a2e280172ab5962581311
b
adbinitdbarea -n DBAREA1 -i 1G,2 -v \
  /dev/disk/by-id/wwn-0x60060e80102172300511166300000000,\
  /dev/disk/by-id/wwn-0x60060e80102172300511166300000001
```

(2) Executing the adbinit command on the master node

On the master node, execute the `adbinit` command specifying the name of the initialization option file created in (1) Creating an initialization option file for the master node.

For details, see (2) Executing the adbinit command in 20.3.1 Creating DB area files using block special files.

(3) Creating an initialization option file for each slave node

Copy the initialization option file created in (1) Creating an initialization option file for the master node to each slave node, and then change the initialization option settings as follows:

- Specify `N` in the `adb_init_dbarea_initialize` operand.
- If a block special file specified in the initialization option file for the master node satisfies either of the following conditions, change the block special file name or delete the corresponding specification:
  - A block special file is specified for a DB area that will not be shared among the HADB servers in the multi-node configuration.
  - There is no block special file that is specific to the slave node and has the same name as a block special file specified for a DB area that will not be shared among the HADB servers in the multi-node configuration.

- Do not change the settings for the other operands.

Example of an initialization option file for a slave node

```
set adb_init_dbarea_initialize = N
set adb_init_mst_blk_path = /dev/disk/by-id/wwn-0x600605b003a2e280172ab596258093bc
   :
```

(4) Executing the adbinit command on each slave node

On the slave node, execute the `adbinit` command specifying the name of the initialization option file created in (3) Creating an initialization option file for each slave node.

For details, see (2) Executing the adbinit command in 20.3.1 Creating DB area files using block special files.

20.4 Return codes

The following table lists and explains the return codes of the `adbinit` command.

Table 20-4: Return codes of the adbinit command

Return code	Description
0	The `adbinit` command terminated normally.
4	The `adbinit` command terminated normally, but a warning message has been issued. Check the displayed message or the warning message output to a message log file.
8	An error occurred while the `adbinit` command was executing. For the cause of the error, see the displayed message or the message that has been output to a message log file.

20.5 DB areas and directories that are created

This section explains the attributes of the DB areas and the directories that are created when the adbinit command is executed.

20.5.1 Attributes of DB areas that are created

The following table describes the attributes of the DB areas that are created by executing the adbinit command.

Table 20-5: Attributes of DB areas that are created

No.	Type of DB area	DB area name	DB area file name	DB area ID	Page size (kilobytes)	Segment size (pages)
1	Master directory DB area	ADBMST	ADBMST	1	4	256
2	Dictionary DB area	ADBDIC	ADBDIC	2	4	8
3	Work table DB area	ADBWRK	ADBWRK	3	Specified in the adb_init_wrk_page_size operand	1
4	System-table DB area	ADBSTBL	ADBSTBL	4	4	16
5	Data DB area	Specified in the -n option#1	Specified in the -n option#1, #2	11 to 1,024	Specified in the -p option#1	4,096 kilobytes ÷ *page size*#3

Notes

- A maximum of 1,014 data DB areas can be created.
- One each of the DB areas other than the data DB areas is created automatically.

#1

Option of the adbinitdbarea operand

#2

Files with the same name as the DB area name are created. If a subdirectory is specified in the -f option of the adbinitdbarea operand, the DB area files are created directly under that subdirectory.

#3

The segment size is determined automatically based on the page size.

Page size (kilobytes)	Segment size (pages)
4	1,024
8	512
16	256
32	128

20.5.2 Directories that are created

When you execute the adbinit command, the directories described in the following table are created under the DB directory.

Table 20-6: Directories that are created under the DB directory

No.	Directory name	Description
1	ADBSYS	Stores files used by HADB. Such files as system log files are stored under this directory.
2	ADBWORK	Stores files used by HADB. Temporary files that are created during execution of the command are stored under this directory.
3	SPOOL	Stores core files containing the error information used by HADB.

Note

In addition to the directories listed in this table, other directories to be used by HADB might also be created.

20.6 Notes

1. Data DB area files cannot be created in a file system that does not support direct I/O.
2. If you want to add DB areas or you want to add DB area files to existing DB areas after you have completed initialization of a database by executing the `adbinit` command, you must use one of the following methods to modify the DB area structure:
    - Execute the `adbmodarea` command to add DB areas and DB area files.
    - Re-executing the `adbinit` command to re-initialize the database.
3. If you use block special files, specify block special files that have been created by LVM or device names that begin with `/dev/disk/` according to HADB's path naming rules.

    When you will be using the multi-node function, specify a device name beginning with `/dev/disk/by-id/wwn-` for a DB area file that will be shared among nodes. For details, see 20.3.2 Creating DB directories (when using the multi-node function).
4. If a disk is added but is not recognized by the OS, restart the OS to let it recognize the disk. When the disk is recognized by the OS, block special files can be used.
5. When you will be using the multi-node function, use the OS command to initialize the file system for the system directory before you execute the `adbinit` command. For details, see the topic *Creating a database* in *Building a system that uses the multi-node function* in *Operations When Using the Multi-Node Function* in the *HADB Setup and Operation Guide*.

21 adbinstall (Install HADB Server or Client)

This chapter explains the functionality provided by the `adbinstall` command and how to use the command.

21.1 Description

The `adbinstall` command installs an HADB server and an HADB client (Linux version).

21.2 Specification format for the adbinstall command

This section explains the specification format and options for the `adbinstall` command.

The following are the OS users that can execute the `adbinstall` command:

- **For installing an HADB server**
  HADB administrator
- **For installing an HADB client (for Linux)**
  OS user managing HADB clients

21.2.1 Explanation of the specification format and options

(1) Specification format

```
/install/adbinstall {-s server-directory-path
                    |-c client-directory-path}
```

For the underlined part, specify the absolute path of the storage where the `adbinstall` command is located.

(2) Explanation of options

The following options specify whether an HADB server or an HADB client (for Linux) is to be installed:

- `-s` *server-directory-path*

  ~<path name>((2 to 118 bytes))

  Specifies the absolute path of a server directory for installing an HADB server.

  Make sure that the HADB administrator can write to the directory specified in this option. If the HADB administrator cannot write to the specified directory, the `KFAA91553-E` message will be output and the command will result in an error.

- `-c` *client-directory-path*

  ~<path name>((2 to 118 bytes))

  Specifies the absolute path of a client directory for installing an HADB client (for Linux).

  In this option, always specify a directory to which the OS user managing HADB clients can write. If the HADB administrator cannot write to the specified directory, the `KFAA91553-E` message will be output and the command will result in an error.

21.3 Return codes

The following table lists and describes the return codes of the `adbinstall` command.

Table 21-1: Return codes of the adbinstall command

Return code	Description
0	The `adbinstall` command terminated normally.
4	The `adbinstall` command issued a warning message and then terminated. Check the displayed message, and then take the appropriate action.
8	An error occurred during execution of the `adbinstall` command. Check the displayed message, and then take the appropriate action.

21.4 Notes

1. The `adbinstall` command is for Linux systems only. This command cannot be used in a Windows system.
2. Store the installation data (`tar.gz` file) in the directory that stores the `adbinstall` command. If the `adbinstall` command and the installation data are not stored in the same directory, it will not be possible to install the HADB server or HADB client.
3. When you install the HADB server, grant the HADB administrator write privileges to the following directories:
    - The directory that stores the `adbinstall` command and the installation data (`tar.gz` file)
    - The directory specified in the `-s` option of the `adbinstall` command

    If you execute the `adbinstall` command by specifying a directory not writable by the HADB administrator, an error occurs (the `KFAA91553-E` message is output). In such cases, grant write privileges to the preceding directories, and then re-execute the `adbinstall` command.
4. When you install an HADB client, grant the OS user who manages the HADB client write privileges to the following directories:
    - The directory that stores the `adbinstall` command and the installation data (`tar.gz` file)
    - The directory specified in the `-c` option of the `adbinstall` command

    If you execute the `adbinstall` command by specifying a directory not writable by the OS user who manages the HADB client, an error occurs (the `KFAA91553-E` message is output). In such cases, grant write privileges to the preceding directories, and then re-execute the `adbinstall` command.
5. If you execute the `adbinstall` command as root, not as the OS user specified as the HADB administrator, a warning message (`KFAA91558-W` message) is output. Normally, the OS user specified as the HADB administrator executes the `adbinstall` command. Therefore, if the `KFAA91558-W` message is output, make sure that executing the `adbinstall` command by root does not cause any problems.

    If a program exists, enter `n` or `N` in response to the input request in the `KFAA91559-Q` message that is output after the `KFAA91558-W` message. Then, execute the `adbinstall` command as the OS user specified as the HADB administrator.

    If you execute the `adbinstall` command as a superuser other than root, the `KFAA91558-W` message is not output.

    > **Note**
    >
    > Here, root indicates a user for which 0 is displayed by executing the OS's `id -u` command. This includes a case where a value of 0 is displayed by executing the OS's `id -u` command after another OS user was changed to root by using the OS's `su` command.

6. If the server directory specified in the `-s` option or the client directory specified in the `-c` option does not exist, the directory will be created automatically when the `adbinstall` command is executed.
7. OS commands are executed as an extension of the `adbinstall` command. If execution of such an OS command fails, an error message will be output to the standard error output. In such a case, check the error message, eliminate the cause of the error, and then re-execute the `adbinstall` command.
8. If you are reinstalling an HADB server or client (including installation of a revised edition or upgrading), make a backup copy of the server directory or client directory before you execute the `adbinstall` command.
9. If the server directory specified in the `-s` option or the client directory specified in the `-c` option already exists, the `KFAA91554-Q` message (asking whether the specified directory is to be overwritten) is displayed. If the directory

is to be overwritten, enter `Y`. If anything other than `Y` is entered, installation processing is canceled and the `adbinstall` command terminates.

22 adbls -d cltgrp (Display Information on Client Groups and Command Groups)

This chapter explains the functionality provided by the `adbls -d cltgrp` command and how to use the command.

22.1 Description

The `adbls -d cltgrp` command displays information on client groups and command groups.

The `adbls -d cltgrp` command displays the following information:

- Information on client groups and command groups
- Information on HADB clients and commands that do not belong to any groups

For details about the client-group facility, see the topic *Client-group facility* in *Architecture* in the *HADB Setup and Operation Guide*.

When the multi-node function is being used, the command displays information on only the nodes on which the command was executed.

22.2 Specification format for the adbls -d cltgrp command

This section explains the specification format and options for the `adbls -d cltgrp` command.

The following OS users can execute the `adbls -d cltgrp` command:

- HADB administrator
- OS users who belong to the HADB administrators group

22.2.1 Explanation of the specification format and options

(1) Specification format

```
adbls -d cltgrp [-t] [-c]
```

(2) Explanation of options

- `-t`

  Suppresses display of the title line. For example, if you need to regularly redirect information to a file, you can specify this option to prevent duplicate title lines.

- `-c`

  Displays comma delimited values. For example, if you want to use spreadsheet software to read an output file to which data has been redirected, you can specify this option to output information delimited by commas.

  If this option is specified, the following types of information are enclosed in double quotation marks (`"`).

  - Character string in the *title of an output item*
  - *Character string* information in an output item

  Numeric value information in an output item is not enclosed in double quotation marks (`"`). Double quotation marks are not used if an output item is empty.

22.3 Examples

This example executes the `adbls -d cltgrp` command to display information on client groups and command groups.

```
adbls -d cltgrp

CLIENT_GROUP   MAX_CON   GUARANTEE_CON   CON   MAX_RTHD   GUARANTEE_RTHD   RTHD   WRNMSG_CON   RSTWRN_CON
GROUP01        2         0               1     4          0                2      90           80
GROUP02        1         1               0     3          3                0      80           60
-----          7         5               0     7          3                0      85           80
```

Information on client groups and command groups is displayed in the following order:

1. Information on client groups and command groups
   This information is output on as many lines as the number of times that the `adbcltgrp` operand in the server definition is specified.
2. Information on HADB clients and commands that do not belong to any groups
   This information is displayed all in one row.

Explanation of the output items

The content of the output items varies depending on the type of information that is output to each row.

■ Information on client groups and command groups

`CLIENT_GROUP`

For a client group, a client group name (the value specified in the `-g` option of the `adbcltgrp` operand in the server definition) is displayed.

For a command group, `command` is displayed.

`MAX_CON`

Displays the maximum number of concurrent connections for the group (the value specified in the `-m` option of the `adbcltgrp` operand in the server definition)#.

`GUARANTEE_CON`

Displays the guaranteed minimum number of concurrent connections for the group (the value specified in the `-u` option of the `adbcltgrp` operand in the server definition)#.

`CON`

Displays the number of HADB clients and commands belonging to groups that are connected to the HADB server.

`MAX_RTHD`

Displays the maximum number of processing real threads usable by a group (the value specified in the `-r` option of the `adbcltgrp` operand in the server definition)#.

`GUARANTEE_RTHD`

Displays the guaranteed minimum number of processing real threads usable by a group (the value specified in the `-e` option of the `adbcltgrp` operand in the server definition)#.

`RTHD`

Displays the number of processing real threads being used by HADB clients and commands that belong to a group.

WRNMSG_CON:

Displays the output trigger of a warning message regarding the maximum number of concurrent connections (the value specified in the -w option of the adbcltgrp operand in the server definition) that is applied to this client group.

RSTWRN_CON:

Displays the trigger to reset the warning message output status regarding the maximum number of concurrent connections (the value specified in the -w option of the adbcltgrp operand in the server definition)# that is applied to this client group.

#

If the value specified for each option of the adbcltgrp operand in the server definition is different from the value that is actually applied, the value that is actually applied is displayed.

■ Information on HADB clients and commands that do not belong to any group

CLIENT_GROUP

Displays -----.

MAX_CON

Displays the freely usable number of concurrent connections#.

GUARANTEE_CON

Displays the number of connections usable by HADB clients and commands that do not belong to any group#.

CON

Displays the total number of connections to the HADB server being used by HADB clients and commands that do not belong to any group.

MAX_RTHD

Displays the freely usable number of processing real threads#.

GUARANTEE_RTHD

Displays the number of processing real threads usable by HADB clients and commands that do not belong to any group#.

RTHD

Displays the total number of processing real threads being used by HADB clients and commands that do not belong to any group.

WRNMSG_CON:

Displays the output trigger# of a warning message regarding the maximum number of concurrent connections that are applied to HADB clients and commands that do not belong to any group.

RSTWRN_CON:

Displays the trigger to reset the warning message output status regarding the maximum number of concurrent connections that are applied to HADB clients and commands that do not belong to any group.

#

For details about each item, see *Points to consider when using the client-group facility* in *Preparing Resources* in the *HADB Setup and Operation Guide*.

22.4 Return codes

The following table lists and describes the return codes of the `adbls` command. These return codes are common to all `adbls` commands.

Table 22-1: Return codes of the adbls command

Return code	Description
`0`	The `adbls` command terminated normally.
`4`	The `adbls` command issued a warning message and then terminated. The execution result of the `adbls` command is displayed. Check the displayed message or the message output to a message log file and take appropriate action.
`8`	An error occurred during execution of the `adbls` command. Check the displayed message or the message output to a message log file and take appropriate action.

22.5 Notes

1. The `adbls` command references the status of the HADB server without locking it so that the performance of the HADB server is not affected. For this reason, an inconsistent value might be displayed depending on when the command is executed. In such a case, re-execute the `adbls` command.
2. If you execute an `adbls` command (except for the `adbls -d srv` or `adbls -d ver` command) after the HADB server has terminated abnormally or forcibly, the information immediately before the abnormal or forced termination is displayed.

   If the multi-node function is used, executing the `adbls` command on the node of the HADB server that terminated abnormally or forcibly displays the status of the HADB server and the node immediately before the abnormal or forced termination.
3. If you execute an `adbls` command (except for the `adbls -d srv` or `adbls -d ver` command) while the HADB server is engaged in start or termination processing, invalid information might be displayed.
4. The `adbls` command (except for the `adbls -d srv` or `adbls -d ver` command) cannot be executed after the HADB server has terminated normally.

23

adbls -d cnct (Display the Connection Status)

This chapter explains the functionality provided by the `adbls -d cnct` command and how to use the command.

23.1 Description

The `adbls -d cnct` command displays the HADB server's connection status.

If the multi-node function is used, information on all connections to the HADB servers in the multi-node configuration will be displayed.

23.2 Specification format for the adbls -d cnct command

This section explains the specification format and options for the `adbls -d cnct` command.

The following OS users can execute the `adbls -d cnct` command:

- HADB administrator
- OS users who belong to the HADB administrators group

23.2.1 Explanation of the specification format and options

(1) Specification format

```
adbls -d cnct [-t] [-c]
```

(2) Explanation of options

- `-t`

  Suppresses display of the title line. For example, if you need to regularly redirect information to a file, you can specify this option to prevent duplicate title lines.

- `-c`

  Displays comma delimited values. For example, if you want to use spreadsheet software to read an output file to which data has been redirected, you can specify this option to output information delimited by commas.

  If this option is specified, the following types of information are enclosed in double quotation marks (`"`).

  - Character string in the *title of an output item*
  - *Character string* information in an output item

  Numeric value information in an output item is not enclosed in double quotation marks (`"`). Double quotation marks are not used if an output item is empty.

23.3 Example

This example executes the adbls -d cnct command to display the HADB server's connection status.

```
adbls -d cnct

CID   CNUMBER    CONNECT_TIME        PROGRAM                        C-PID      IP-ADDRESS        STATUS
1     2          2016-06-08 10:11:54 SAMPLE                         20770      (127.0.0.1      ) STARTED

TRN_ISO_LV      ACCESS_MODE CONNECTION_INFORMATION           NODE_NO CLIENT_TYPE      CLIENT_GROUP
READ_COMMITTED  READ_ONLY   000010000000002-000050c5df220700         AP(C Library)    GROUP1

SQL_ELAPSED_TIME
3005125
```

Note:

- If no connection has been established, only the titles of the output items are displayed.
- In the preceding example, the line is divided at the following points. However, in the actual output all items are displayed side-by-side on the same line.
  - Between STATUS and TRN_ISO_LV
  - Between CLIENT_GROUP and SQL_ELAPSED_TIME

Explanation of the output items

CID

Displays the connection ID.

CNUMBER

Displays the connection sequence numbers since the HADB server started.

CONNECT_TIME

Displays the date and time the connection was established.

PROGRAM

Displays the application identifier.

- For application programs or the adbsql command

  The application identifier specified in the adb_clt_ap_name operand in the client definition is displayed. If the adb_clt_ap_name operand is omitted, ******** is displayed.
- For commands (excluding the adbsql command)

  If the connection was established by a command, such as adbimport, the command name is displayed (example: adbimport).

C-PID

Displays the process ID of the HADB client that established the connection. Note that 0 is displayed for the connection from a JDBC driver.

IP-ADDRESS

Displays the IP address of the HADB client that established the connection.

STATUS

Displays the connection status. One of the following is displayed:

- NOT_STARTED: No transaction has started.
- THREAD_WAITING: Waiting for allocation of a processing real thread

- STARTING: A transaction is starting.
- STARTED: A transaction is executing.
- COMMITTING: Commit processing is underway.
- ROLLBACKING: Rollback processing is underway.

The following is displayed only if the multi-node function is used:

- SWITCH_WAITING: Pending switchover to the master node
- RETURN_WAITING: Pending node recovery
- NODE_WAITING: Waiting for ongoing transactions executed on other nodes to finish

▪ **If you are using the multi-node function**

- For the node that is executing a transaction, the following status might be displayed (because the connection might be changed to the following status). For nodes that are not executing transactions, the following status is not displayed.
  - THREAD_WAITING: Waiting for allocation of a processing real thread
  - STARTING: A transaction is starting.
  - COMMITTING: Commit processing is underway.
  - ROLLBACKING: Rollback processing is underway.
- For the master node, the following status might be displayed (because the connection might be changed to the following status). For slave nodes, the following status is not displayed.
  - SWITCH_WAITING: Pending switchover to the master node
  - RETURN_WAITING: Pending node recovery
  - NODE_WAITING: Waiting for ongoing transactions executed on other nodes to finish

TRN_ISO_LV

Displays the transaction isolation level. One of the following is displayed:

- READ_COMMITTED: The transaction isolation level is READ COMMITTED.
- REPEATABLE_READ: The transaction isolation level is REPEATABLE READ.

ACCESS_MODE

Displays the transaction access mode. One of the following is displayed:

- READ_ONLY: Read-only transaction
- READ_WRITE: Read/write transaction

CONNECTION_INFORMATION

Displays the connection information. This is the same information as was displayed in the KFAA81000-I message that was output to the message log file when the connection was established.

NODE_NO

Displays the node number of the node that is executing a transaction over the connection.

However, if a connection has been established from the JDBC driver, when a ResultSet object is created with HOLD_CURSORS_OVER_COMMIT specified for the holdability, the node number is displayed even if the transaction has terminated normally. In this case, the node number of the node that was executing the transaction before normal termination is displayed.

When the multi-node function is not being used, nothing is displayed.

CLIENT_TYPE

Displays the client type that established the connection.

- AP(JDBC): An application program connected from a JDBC driver.
- AP(C Library): An application program connected from a driver other than a JDBC driver.
- command: A command is connected.

CLIENT_GROUP

- For an application program or the adbsql command:

  Displays the client group name to which the HADB client belongs. If the HADB client belongs to a command group, command is displayed.

  If the HADB client does not belong to any group, nothing is displayed.
- For commands (excluding the adbsql command):

  If a command group is set up, command is displayed.

  If no command group is set up, nothing is displayed.

SQL_ELAPSED_TIME

Displays the elapsed time (microseconds) of the SQL statement for which the longest time has elapsed among the SQL statements being executed on the connection since the execution started.

If no SQL statement is being executed, nothing is displayed.

If you are using the multi-node function, the elapsed time of the SQL statement is displayed only on the node that is executing transactions.

If the adbls -d cnct command is executed after the HADB server has terminated abnormally or forcibly during execution of SQL statements, the difference between the following two times is output as the elapsed time:

- The start time of the SQL statement for which the longest time has elapsed among the SQL statements being executed since the execution started
- The time when the HADB server terminated abnormally or forcibly

23.4 Return codes

For details about the return codes of the `adbls -d cnct` command, see 22.4 Return codes.

23.5 Notes

For notes about the `adbls -d cnct` command, see 22.5 Notes.

24 adbls -d gbuf (Display Global Buffer Information)

This chapter explains the functionality provided by the `adbls -d gbuf` command and how to use the command.

24.1 Description

The `adbls -d gbuf` command displays information about global buffers.

When the multi-node function is being used, only the information about the node where the command is executed is displayed.

24.2 Specification format for the adbls -d gbuf command

This section explains the specification format and options for the `adbls -d gbuf` command.

The following OS users can execute the `adbls -d gbuf` command:

- HADB administrator
- OS users who belong to the HADB administrators group

24.2.1 Explanation of the specification format and options

(1) Specification format

```
adbls -d gbuf [-t] [-c]
```

(2) Explanation of options

- `-t`

  Suppresses display of the title line. For example, if you need to regularly redirect information to a file, you can specify this option to prevent duplicate title lines.

- `-c`

  Displays comma delimited values. For example, if you want to use spreadsheet software to read an output file to which data has been redirected, you can specify this option to output information delimited by commas.

  If this option is specified, the following types of information are enclosed in double quotation marks (`"`).

  - Character string in the *title of an output item*
  - *Character string* information in an output item

  Numeric value information in an output item is not enclosed in double quotation marks (`"`). Double quotation marks are not used if an output item is empty.

24.3 Examples

This example executes the `adbls -d gbuf` command to display information about the global buffers.

```
BUFF_NAME                                 DBAREA_NAME                    PAGE_SIZE PAGE_NUM
##ADBOTHER#0000004096                     ADBDIC                         4         8580 D
                                          ADBSTBL
                                          area3
##ADBOTHER#0000032768                     area4                          32        8580 D
ADBMST                                    ADBMST                         4         132 D
ADBWRK                                    ADBWRK                         256       128 S
buf1                                      area1                          4         1000 S
                                          area2

RANGE_IDX_PAGE_NUM TABLE_SCAN_BUFFER_SIZE(MB) TABLE_SCAN_BUFFER_THERAD_MAX_SIZE(MB)
0                  0                          0


0                  0                          0
0                  0                          0
0                  0                          0
0                  400                        40
```

Explanation of the output items

`BUFF_NAME`

Displays a global buffer name specified in the `-g` option of the `adbbuff` operand in the server definition or a global buffer name determined automatically by HADB.

> **Note**
>
> If the `-o` option was specified in the `adbbuff` operand, the global buffer name determined automatically by HADB is displayed as *global-buffer-name*`#`*nnnnnnnnnn* (*n*: page size as a 10-digit decimal number).
>
> If the `-o` option was omitted, the global buffer name determined automatically by HADB is displayed as `##ADBOTHER#`*nnnnnnnnnn* (*n*: page size as a 10-digit decimal number).

`DBAREA_NAME`

Displays the DB area name specified in the `-n` option of the `adbbuff` operand in the server definition or the DB area name for which HADB allocated the global buffer automatically.

If multiple DB areas are allocated to the global buffer, each of those DB area names is displayed on a separate line.

`PAGE_SIZE`

Displays in kilobytes the global buffer page size. This is the same value as the page size for the DB area.

If the page size of the work table DB area was changed by specifying the `adb_dbarea_wrk_page_size` operand in the server definition when the HADB server started, the page size after the change is displayed for `ADBWRK` (global buffer for global work table).

`PAGE_NUM`

Displays the number of pages in the global buffer specified in the `-p` option of the `adbbuff` operand in the server definition.

For `ADBWRK` (global buffer for global work table), the `adb_dbbuff_wrktbl_glb_blk_num` operand value specified in the server definition is displayed.

If the -p option was omitted in the adbbuff operand in the server definition or the adb_dbbuff_wrktbl_glb_blk_num operand was omitted in the server definition, the default value is displayed.

You can determine which value applies to the number of global buffer pages by checking the character that follows the number of pages value. The following table explains the meanings of the suffixed characters.

Table 24-1: Meanings of the characters displayed following the number of global buffer pages value

No.	Suffixed character	Meaning of the character	Corresponding global buffer	Value that is displayed as the number of global buffer pages
1	S	Value specified in the server definition is used.	Global buffer defined in the adbbuff operand in the server definition	-p option value in the adbbuff operand in the server definition
2			Global buffer for global work table (ADBWRK)	adb_dbbuff_wrktbl_glb_blk_num operand value in the server definition
3	D	Default value is used.	Global buffer defined in the adbbuff operand in the server definition or a global buffer that is defined automatically	-p option value in the adbbuff operand in the server definition
4			Global buffer for global work table (ADBWRK)	Default value for the adb_dbbuff_wrktbl_glb_blk_num operand in the server definition
5			Global buffer for the master directory (ADBMST)	Default value for ADBMST

RANGE_IDX_PAGE_NUM

Displays the number of pages in the global buffer dedicated to the range index specified in the -a option of the adbbuff operand in the server definition.

If the -a option was omitted in the adbbuff operand in the server definition, 0 is displayed.

TABLE_SCAN_BUFFER_SIZE(MB)

Displays the memory size (in megabytes) used for the table scan buffer.

The displayed value varies depending on the specification of the -v or -k option of the adbbuff operand in the server definition. The following table shows the details.

Table 24-2: Value displayed for TABLE_SCAN_BUFFER_SIZE(MB)

No.	Whether the table scan buffer is specified	Value displayed for TABLE_SCAN_BUFFER_SIZE(MB)
1	The -v option of the adbbuff operand is specified.	Value specified for *memory-size-used-for-the-table-scan-buffer* in the -v option
2	The -k option of the adbbuff operand is specified.	Value determined from the following formula: Value specified for the -k option × 4
3	Neither the -v nor -k option of the adbbuff operand is specified.	0

TABLE_SCAN_BUFFER_THREAD_MAX_SIZE(MB)

Displays the maximum memory size (in megabytes) that can be used for each real thread of the table scan buffer.

The displayed value varies depending on the specification of the -v or -k option of the adbbuff operand in the server definition. The following table shows the details.

Table 24-3: Value displayed for TABLE_SCAN_BUFFER_THREAD_MAX_SIZE(MB)

No.	Whether the table scan buffer is specified	Whether *maximum-memory-size-that-can-be-used-for-each-real-thread-of-the-table-scan-buffer* is specified in the -v option	Value displayed for TABLE_SCAN_BUFFER_THREAD_MAX_SIZE(MB)
1	The -v option of the adbbuff operand is specified.	Specified	Value specified for *maximum-memory-size-that-can-be-used-for-each-real-thread-of-the-table-scan-buffer* in the -v option
2		Not specified	0
3	The -k option of the adbbuff operand is specified.	--	Value determined from the following formula: Value specified for the -k option × 4 ÷ value of the adb_sys_rthd_num operand in the server definition If the value determined by the preceding calculation is less than 5 megabytes, the value is rounded up to 5 megabytes.
4	Neither the -v nor -k option of the adbbuff operand is specified.	--	0

Legend:

--: Not applicable

Note

For details about the server definition operands, see the topic *Detailed descriptions of the server definition operands* in *Designing the Server Definition* in the *HADB Setup and Operation Guide*.

For details about DB area names, see 20.5.1 Attributes of DB areas that are created.

24.4 Return codes

For details about the return codes of the `adbls -d gbuf` command, see 22.4 Return codes.

24.5 Notes

For notes about the `adbls -d gbuf` command, see 22.5 Notes.

25 adbls -d lbuf (Display Local Work Table Buffer Information)

This chapter explains the functionality provided by the `adbls -d lbuf` command and how to use the command.

25.1 Description

The `adbls -d lbuf` command displays information about local work table buffers.

When the multi-node function is used, the following information is displayed:

- The number of pages and page size of the local work table buffer for the node on which the `adbls -d lbuf` command is executed#
- The number of pages and page size of the local work table buffer for each connection connected to the HADB server in the multi-node configuration (the number of pages and page size used when executing transactions on the node on which the `adbls -d lbuf` command was executed)

#

- The value specified in the server definition `adb_dbbuff_wrktbl_clt_blk_num` operand is displayed for the number of pages.
- The value specified in the initialization option of the `adbinit` command (adb_init_wrk_page_size) is displayed for the page size.

Note

The page size of the local work table buffer is the same value as the work table DB area page size (`adb_init_wrk_page_size`).

25.2 Specification format for the adbls -d lbuf command

This section explains the specification format and options for the `adbls -d lbuf` command.

The following OS users can execute the `adbls -d lbuf` command:

- HADB administrator
- OS users who belong to the HADB administrators group

25.2.1 Explanation of the specification format and options

(1) Specification format

```
adbls -d lbuf [-t] [-c]
```

(2) Explanation of options

- `-t`

  Suppresses display of the title line. For example, if you need to regularly redirect information to a file, you can specify this option to prevent duplicate title lines.

- `-c`

  Displays comma delimited values. For example, if you want to use spreadsheet software to read an output file to which data has been redirected, you can specify this option to output information delimited by commas.

  If this option is specified, the following types of information are enclosed in double quotation marks (`"`).

  - Character string in the *title of an output item*
  - *Character string* information in an output item

  Numeric value information in an output item is not enclosed in double quotation marks (`"`). Double quotation marks are not used if an output item is empty.

25.3 Examples

This example executes the `adbls -d lbuf` command to display information about the local work table buffer.

```
adbls -d lbuf

CNUMBER     PAGE_SIZE PAGE_NUM
0           256       256 D
1           256       128 C
3           256       10 C
4           256       256 D
5           256       256 D
```

Explanation of the output items

`CNUMBER`

Displays the connection sequence number since the HADB server started.

To check the application identifier or command that corresponds to a displayed connection sequence number, use the `adbls -d cnct` command.

In the execution results of the `adbls -d cnct` command, check the `PROGRAM` column for the same connection sequence number.

If `0` is displayed under `CNUMBER`, one of the following values is displayed under `PAGE_NUM`:

- The value specified for the `adb_dbbuff_wrktbl_clt_blk_num` operand in the server definition or the default value
- The value specified for the `adb_dbbuff_wrktbl_clt_blk_num` buffer-modifying option

For details about the `adbls -d cnct` command, see 23. adbls -d cnct (Display the Connection Status).

`PAGE_SIZE`

Displays in kilobytes the page size for the local work table buffer used for application program or command processing. This is the same value as the page size for the work table DB area.

If the page size of the work table DB area was changed by specifying the `adb_dbarea_wrk_page_size` operand in the server definition when the HADB server started, the page size after the change is displayed.

`PAGE_NUM`

Displays the number of local work table buffer pages used for application program or command processing.

One of the following values is displayed:

- The value specified for the `adb_dbbuff_wrktbl_clt_blk_num` operand in the server definition, or the default value for this operand
- The value specified for the `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition
- The value specified for the `adb_export_wrktbl_blk_num` export option
- The value specified for the `adb_dbbuff_wrktbl_clt_blk_num` buffer-modifying option

You can determine which value applies to the number of local work table buffer pages by checking the character that follows the number of pages value. The following table explains the meanings of the suffixed characters.

Table 25-1: Meaning of the characters displayed following the value displayed for the number of local work table buffer pages

No.	Suffixed character	Meaning of the character	Value that is displayed as the number of local work table buffer pages
1	`S`	Value specified in the server definition is used.	Value specified for the `adb_dbbuff_wrktbl_clt_blk_num` operand in the server definition
2	`S*`	Value specified in the buffer-modifying option is used.	Value specified for the `adb_dbbuff_wrktbl_clt_blk_num` buffer-modifying option
3	`C`	Value specified in the client definition or the export option is used.	Value specified for the `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition
4			Value specified for the `adb_export_wrktbl_blk_num` export option
5	`D`	Default value is used.	Default value for the `adb_dbbuff_wrktbl_clt_blk_num` operand in the server definition

The number of local work table buffer pages that is displayed by this command depends on the settings in the server definition and the client definition and whether the `adbmodbuff` command has been executed. The following examples explain these variations in detail.

Example 1: When the `adb_dbbuff_wrktbl_clt_blk_num` operand is omitted in the server definition

```
adbls -d lbuf

CNUMBER     PAGE_SIZE  PAGE_NUM
0           256        256 D
1           256        128 C
3           256        10 C
4           256        256 D
5           256        256 D
```

- On the row where `CNUMBER` is `0`, the default value of the `adb_dbbuff_wrktbl_clt_blk_num` operand in the server definition is displayed.
- For connection sequence numbers 1 and 3, the `adb_dbbuff_wrktbl_clt_blk_num` operand is specified in the client definition. Therefore, on the rows where `CNUMBER` is `1` and `3`, the value specified for the `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition is displayed under `PAGE_NUM`.
- For connection sequence numbers 4 and 5, the `adb_dbbuff_wrktbl_clt_blk_num` operand is omitted in the client definition. Therefore, on the rows where `CNUMBER` is `4` and `5`, the default value of the `adb_dbbuff_wrktbl_clt_blk_num` operand in the server definition is displayed under `PAGE_NUM`.

Example 2: When the `adb_dbbuff_wrktbl_clt_blk_num` operand is specified in the server definition

```
adbls -d lbuf

CNUMBER     PAGE_SIZE  PAGE_NUM
0           256        128 S
1           256        128 C
3           256        10 C
4           256        128 S
5           256        128 S
```

- On the row where `CNUMBER` is `0`, the value specified for the `adb_dbbuff_wrktbl_clt_blk_num` operand in the server definition is displayed.

- For connection sequence numbers 1 and 3, the `adb_dbbuff_wrktbl_clt_blk_num` operand is specified in the client definition. Therefore, on the rows where `CNUMBER` is `1` and `3`, the value specified for the `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition is displayed under `PAGE_NUM`.
- For connection sequence numbers 4 and 5, the `adb_dbbuff_wrktbl_clt_blk_num` operand is omitted in the client definition. Therefore, on the rows where `CNUMBER` is `4` and `5`, the value specified for the `adb_dbbuff_wrktbl_clt_blk_num` operand in the server definition is displayed under `PAGE_NUM`.

Example 3: When the `adbmodbuff` command was executed

```
adbls -d lbuf

CNUMBER     PAGE_SIZE  PAGE_NUM
0           256        128 S*
1           256        128 C
3           256        10 C
4           256        128 S*
5           256        128 S*
```

- On the row where `CNUMBER` is `0`, the value specified for the `adb_dbbuff_wrktbl_clt_blk_num` buffer-modifying option is displayed.
- For connection sequence numbers 1 and 3, the `adb_dbbuff_wrktbl_clt_blk_num` operand is specified in the client definition. Therefore, on the rows where `CNUMBER` is `1` and `3`, the value specified for the `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition is displayed under `PAGE_NUM`.
- For connection sequence numbers 4 and 5, the `adb_dbbuff_wrktbl_clt_blk_num` operand is omitted in the client definition. Therefore, on the rows where `CNUMBER` is `4` and `5`, the value specified for the `adb_dbbuff_wrktbl_clt_blk_num` buffer-modifying option is displayed under `PAGE_NUM`.

Note

- For details about the `adb_dbbuff_wrktbl_clt_blk_num` operand in the server definition, see the topic *Operands related to performance (set format)* in *Detailed descriptions of the server definition operands* in *Designing the Server Definition* in the *HADB Setup and Operation Guide*.
- For details about the `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition, see the topic *Operands related to performance* in *Contents of operands in the client definition* in *Designing Client Definitions* in the *HADB Application Development Guide*.
- For details about the `adb_export_wrktbl_blk_num` export option, see 14.2.2 Format of export options.
- For details about the `adb_dbbuff_wrktbl_clt_blk_num` buffer-modifying option, see 35.2.2 Format of the buffer-modifying option.

25.4 Return codes

For details about the return codes of the `adbls -d lbuf` command, see 22.4 Return codes.

25.5 Notes

For notes about the `adbls -d lbuf` command, see 22.5 Notes.

26 adbls -d lock (Display the Status of Locked Resources)

This chapter explains the functionality provided by the `adbls -d lock` command and how to use the command.

26.1 Description

The `adbls -d lock` command displays the status of each locked resource.

If the multi-node function is used, the information on the lock acquired by the connection connected to the node on which an `adbls -d lock` command was executed is displayed.

26.2 Specification format for the adbls -d lock command

This section explains the specification format and options for the `adbls -d lock` command.

The following OS users can execute the `adbls -d lock` command:

- HADB administrator
- OS users who belong to the HADB administrators group

26.2.1 Explanation of the specification format and options

(1) Specification format

```
adbls -d lock [-t] [-c]
```

(2) Explanation of options

- `-t`

  Suppresses display of the title line. For example, if you need to regularly redirect information to a file, you can specify this option to prevent duplicate title lines.

- `-c`

  Displays comma delimited values. For example, if you want to use spreadsheet software to read an output file to which data has been redirected, you can specify this option to output information delimited by commas.

  If this option is specified, the following types of information are enclosed in double quotation marks (`"`).

  - Character string in the *title of an output item*
  - *Character string* information in an output item

  Numeric value information in an output item is not enclosed in double quotation marks (`"`). Double quotation marks are not used if an output item is empty.

26.3 Examples

This example executes the `adbls -d lock` command to display the status of each locked resource.

```
adbls -d lock

RESOURCETYPE      RESOURCEID LOCK_MODE NODE_NO CID        STATUS  COUNT                LOCK_TIME           ESCALAT_MODE
SINGLE_CONNECTION *          SHARE             1          LOCKING 1                    2016-02-25 01:18:46
TABLE             0x00020191 SHARE             1          LOCKING 1                    2016-02-25 01:18:49
TABLE             0x000200C9 SHARE             1          LOCKING 2                    2016-02-25 01:18:49
TABLE             0x000200CA SHARE             1          LOCKING 2                    2016-02-25 01:18:49
TABLE             0x000200CB SHARE             1          LOCKING 2                    2016-02-25 01:18:49
PREPARE_TABLE     0x000200CC SHARE             1          LOCKING 1                    2016-02-25 01:18:49
PREPARE_TABLE     0x00020191 SHARE             1          LOCKING 1                    2016-02-25 01:18:49
SINGLE_CONNECTION *          SHARE             2          LOCKING 1                    2016-02-25 01:19:09
DBAREA            0x0000000D SHARE             2          LOCKING 1                    2016-02-25 01:19:09
DBAREA            0x0000000E SHARE             2          LOCKING 1                    2016-02-25 01:19:09
DBAREA            0x00000004 SHARE             2          LOCKING 1                    2016-02-25 01:19:09
TABLE             0x00020191 PRTCT_UPD         2          LOCKING 1                    2016-02-25 01:19:09
TABLE             0x000200C9 SHARE             2          LOCKING 3                    2016-02-25 01:19:09
TABLE             0x000200CA SHARE             2          LOCKING 3                    2016-02-25 01:19:09
TABLE             0x000200CB SHARE             2          LOCKING 3                    2016-02-25 01:19:09
PREPARE_TABLE     0x00020191 SHARE             2          ESCALAT 2                    2016-02-25 01:19:09 EXCLUSIVE
```

Explanation of the output items

`RESOURCETYPE`

Displays the type of locked resources for which an allocation request is issued by the application program or command. One of the following is displayed:

- `SINGLE_CONNECTION`: Single connection
- `DATABASE_ACCESS`: Database access
- `DICTIONARY`: Dictionary
- `DBAREA`: DB area

  This is a data DB area, dictionary DB area, or a system-table DB area.
- `TABLE`: Table

  This is a table to be processed, a system table, or a dictionary table.
- `PREPARE_TABLE`: This is a pre-processing table.

`RESOURCEID`

Displays the ID of each locked resource for which an allocation request was issued by an application program or a command.

- If the type of locked resource is single connection, database access, or dictionary

  Asterisk (`*`) is displayed.
- If the type of locked resource is DB area

  The DB area's ID is displayed.
- If the type of locked resource is table or pre-processing table

  The table's ID is displayed.

  If a locked resource allocation request has been issued for a dictionary table, the table ID of the dictionary table (base table) is displayed. If a locked resource allocation request has been issued for a system table, the table ID of the system table (base table) is displayed.

`LOCK_MODE`

Displays the lock mode. One of the following is displayed:

- `EXCLUSIVE`: Exclusive mode (EXCLUSIVE) (EX)

- PRTCT_UPD: Protected update mode (PROTECTED UPDATE) (PU)
- SHARE_UPD: Shared update mode (SHARED UPDATE) (SU)
- SHARE: Shared retrieval mode (SHARED RETRIEVE) (SR)

NODE_NO

When the multi-node function is being used, displays the node number of the node that issued the locked resource allocation request.

When the multi-node function is not being used, nothing is displayed.

CID

Displays the connection ID of the connection that issued the locked resource allocation request.

If the maintenance processing of the updated-row columnizing facility issues a locked resource allocation request, "SYSTEM" is displayed.

STATUS

Displays the status of the locked resource allocation request. One of the following is displayed:

- WAITING: Waiting for allocation of the locked resource
- LOCKING: Allocating the locked resource
- ESCALAT: Waiting for the transition of the lock mode
- SYSWAIT: Waiting until the resource locked previously by the maintenance processing of the updated-row columnizing facility is unlocked

COUNT

Displays the number of times the locked resource has been allocated by the same transaction.

If the locked resource has never been allocated, nothing is displayed.

LOCK_TIME

Displays the first time a locked resource allocation request was issued.

ESCALAT_MODE

If ESCALAT is output to the STATUS column, the lock mode that is applied after transition from the current lock mode is displayed.

If a value other than ESCALAT is output to the STATUS column, nothing is displayed.

- EXCLUSIVE: Exclusive mode (EXCLUSIVE) (EX)
- PRTCT_UPD: Protected update mode (PROTECTED UPDATE) (PU)
- SHARE_UPD: Shared update mode (SHARED UPDATE) (SU)
- SHARE: Shared retrieval mode (SHARED RETRIEVE) (SR)

Note

For details about locked resources and the lock mode, see the topic *Locking* in *Architecture* in the *HADB Setup and Operation Guide*.

26.4 Return codes

For details about the return codes of the `adbls -d lock` command, see 22.4 Return codes.

26.5 Notes

For notes about the `adbls -d lock` command, see 22.5 Notes.

27 adbls -d mem (Display the Memory Usage Status)

This chapter explains the functionality provided by the `adbls -d mem` command and how to use the command.

27.1 Description

The `adbls -d mem` command displays the memory usage status of the server process.

When the multi-node function is being used, only the information about the node where the command is executed is displayed.

27.2 Specification format for the adbls -d mem command

This section explains the specification format for the `adbls -d mem` command and its options.

The following OS users can execute the `adbls -d mem` command:

- HADB administrator
- OS users who belong to the HADB administrators group

27.2.1 Explanation of the specification format and options

(1) Specification format

```
adbls -d mem [-a] [-t] [-c]
```

(2) Explanation of options

- `-a`

  Specifies that the status of the memory allocated by the server process (shared memory and heap memory) is to be displayed.

  If this option is omitted, the usage status of the process common memory and real thread private memory out of the shared memory is displayed.

- `-t`

  Suppresses display of the title line. For example, if you need to regularly redirect information to a file, you can specify this option to prevent duplicate title lines.

- `-c`

  Displays comma delimited values. For example, if you want to use spreadsheet software to read an output file to which data has been redirected, you can specify this option to output information delimited by commas.

  If this option is specified, the following types of information are enclosed in double quotation marks (`"`).

  - Character string in the *title of an output item*
  - *Character string* information in an output item

  Numeric value information in an output item is not enclosed in double quotation marks (`"`). Double quotation marks are not used if an output item is empty.

27.3 Examples

This subsection explains how to use the adbls -d mem command to display the memory usage status.

When the adbls -d mem command is executed, the output items vary depending on whether the -a option is specified.

27.3.1 Displaying the usage status of the process common memory and real thread private memory (-a option not specified)

The adbls -d mem command is executed to display the usage status of the process common memory and real thread private memory out of the shared memory allocated by the server process.

```
adbls -d mem

THREAD-NO   LIMIT         USE          FREE         MAX
0           1073741824    105840224    162660848    105840224
1           1073741824    901024       267600048    901056
2           1073741824    900512       267600560    900512
3           1073741824    900512       267600560    900512
4           1073741824    900512       267600560    900512
5           1073741824    900512       267600560    900512
:               :           :              :          :
```

Explanation of the output items

THREAD-NO

Row 0 displays information about the process common memory.

Each row starting with row 1 displays information about real thread private memory.

LIMIT

Displays the maximum amount of memory (in bytes) that is available for use by the HADB server.

If the adb_sys_memory_limit operand in the server definition is specified, an asterisk (*) is displayed.

USE

Displays the amount of memory (in bytes) the HADB server is currently using.

FREE

Displays the amount of available memory that is not being used (in bytes).

MAX

Displays the maximum amount of memory (in bytes) the HADB server has used until now.

27.3.2 Displaying the total memory size (-a option specified)

The adbls -d mem command is executed to display the status of the memory allocated by the server process (shared memory and heap memory).

```
adbls -d mem -a

 TOTAL_LIMIT(MB) TOTAL_USE(MB) TOTAL_USE_PER(%) TOTAL_MAX(MB) TOTAL_MAX_TIME      TOTAL_MAX_1h(MB) TOTAL_MAX_TIME_1h
 TOTAL_MAX_1d(MB) TOTAL_MAX_TIME_1d   GBUFFER_USE(MB) PROCESS_USE(MB) THREAD_USE(MB) HEAP_USE(MB) OTHER_USE(MB)
 64000           37417         58.5             37417         2017/06/28 12:10:25 37417            2017/06/29 08:30:00
 37417            2017/06/28 12:10:25 45              4001            32208          1018         145
```

Explanation of the output items

`TOTAL_LIMIT`

Displays the upper limit (in megabytes) for the total amount of memory that can be allocated by the HADB server (the value specified for the `adb_sys_memory_limit` operand in the server definition).

If the `adb_sys_memory_limit` operand in the server definition is not specified, an asterisk (`*`) is displayed.

`TOTAL_USE`

Displays the total amount of memory (in megabytes) currently allocated by the HADB server.

`TOTAL_USE_PER`

Displays the ratio (in percent) between the total amount of memory (in megabytes) currently allocated by the HADB server and the upper limit for the total amount of memory that can be allocated by the HADB server.

If the `adb_sys_memory_limit` operand in the server definition is not specified, an asterisk (`*`) is displayed.

`TOTAL_MAX`

Displays the largest amount of total memory (in megabytes) that has been allocated since the HADB server started.

`TOTAL_MAX_TIME`

Displays the time at which the total amount of HADB memory reached the value of the output item `TOTAL_MAX`. If the value of `TOTAL_MAX` has been reached several times since the HADB server started, the oldest time is displayed.

`TOTAL_MAX_1h`

Displays the largest amount of total memory (in megabytes) that was allocated by the HADB server within the most resent hour.

If the `adbls -d mem` command with the `-a` option specified is executed after the HADB server terminated abnormally, the largest amount of total memory allocated by the HADB server is also displayed. In this case, the range of displayed information is within one hour from the time the HADB server terminated abnormally.

Depending on when the `adbls -d mem` command with the `-a` specified is executed, the maximum size of allocated memory for only the most recent 59 minutes might be displayed.

`TOTAL_MAX_TIME_1h`

Displays the time at which the total amount of HADB memory reached the value of the output item `TOTAL_MAX_1h`. If the value of `TOTAL_MAX_1h` has been reached several times within the most recent hour, the oldest time is displayed.

`TOTAL_MAX_1d`

Displays the largest amount of total memory (in megabytes) that was allocated by the HADB server within the most recent day.

If the `adbls -d mem` command with the `-a` option specified is executed after the HADB server has terminated abnormally, the largest amount of total memory allocated by the HADB server is also displayed. In this case, the range of displayed information is within one day from the time the HADB server terminated abnormally.

Depending on when the `adbls -d mem` command with the `-a` specified is executed, the maximum size of allocated memory for only the most recent 23 hours might be displayed.

`TOTAL_MAX_TIME_1d`

Displays the time at which the total amount of HADB memory reached the value of the output item `TOTAL_MAX_1d`. If the value of the `TOTAL_MAX_1d` has been reached several times within the most recent day, the oldest time is displayed.

`GBUFFER_USE`

Displays the total amount of global buffer pages (in megabytes) currently allocated by the HADB server.

`PROCESS_USE`

Displays the total amount of process common memory (in megabytes) currently allocated by the HADB server.

If the `adb_sys_shm_huge_page_size` operand in the server definition is specified, the sum of the values rounded up by the value of the `adb_sys_shm_huge_page_size` operand in the server definition is displayed when shared memory is allocated. Therefore, this value might differ from the sum of the values of `USE` and `FREE` for process common memory that are displayed when the `adbls -d mem` command is executed without the `-a` option specified.

`THREAD_USE`

Displays the total amount of real thread private memory (in megabytes) currently allocated by the HADB server.

If the `adb_sys_shm_huge_page_size` operand in the server definition is specified, the sum of the values rounded up by the value of the `adb_sys_shm_huge_page_size` operand in the server definition is displayed when shared memory is allocated. Therefore, this value might differ from the sum of the values of `USE` and `FREE` for real thread private memory that are displayed when the `adbls -d mem` command is executed without the `-a` option specified.

`HEAP_USE`

Displays the total amount of heap memory (in megabytes) currently allocated by the HADB server.

`OTHER_USE`

Displays the total amount of memory for the shared memory management area (in megabytes) currently allocated by the HADB server.

27.4 Return codes

For details about the return codes of the `adbls -d mem` command, see 22.4 Return codes.

27.5 Notes

For notes about the `adbls -d mem` command, see 22.5 Notes.

28 adbls -d node (Display the HADB Server Status on Each Node)

This chapter explains the functionality provided by the `adbls -d node` command and how to use the command.

28.1 Description

The `adbls -d node` command displays the status of each HADB server in a multi-node configuration.

When the multi-node function is not being used, the command displays only the titles of the output items.

28.2 Specification format for the adbls -d node command

This section explains the specification format and options for the `adbls -d node` command.

The following OS users can execute the `adbls -d node` command:

- HADB administrator
- OS users who belong to the HADB administrators group

28.2.1 Explanation of the specification format and options

(1) Specification format

```
adbls -d node [-t] [-c]
```

(2) Explanation of options

- `-t`

  Suppresses display of the title line. For example, if you need to regularly redirect information to a file, you can specify this option to prevent duplicate title lines.

- `-c`

  Displays comma delimited values. For example, if you want to use spreadsheet software to read an output file to which data has been redirected, you can specify this option to output information delimited by commas.

  If this option is specified, the following types of information are enclosed in double quotation marks (`"`).

  - Character string in the *title of an output item*
  - *Character string* information in an output item

  Numeric value information in an output item is not enclosed in double quotation marks (`"`). Double quotation marks are not used if an output item is empty.

28.3 Examples

These examples display the status of each HADB server in a multi-node configuration.

Example 1 (HADB servers are running on all nodes)

```
adbls -d node

NODE_NO NODE_TYPE SVID       IP-ADDRESS        STATUS   START_TIME
1       MASTER    22602      (XX.XXX.XX.14   ) ACTIVE   2014-12-05 18:03:22
2       SLAVE     16226      (XX.XXX.XX.16   ) ACTIVE   2014-12-05 18:03:13
3       SLAVE     7749       (XX.XXX.XX.6    ) ACTIVE   2014-12-05 18:03:19
```

Example 2 (one node has been disconnected from the multi-node configuration)

```
adbls -d node

NODE_NO NODE_TYPE SVID       IP-ADDRESS        STATUS   START_TIME
1       MASTER               (XX.XXX.XX.14   ) ABORT
2       MASTER    16226      (XX.XXX.XX.16   ) ACTIVE   2014-12-05 18:03:13
3       SLAVE     7749       (XX.XXX.XX.6    ) ACTIVE   2014-12-05 18:03:19
```

In this example, the HADB server on node number 1 terminated abnormally and has been disconnected from the multi-node configuration.

Note

If the `adbls -d node` command is executed on a node that has been disconnected from the multi-node configuration, different information is output on the disconnected node (the status of each node immediately before the node was disconnected).

Explanation of the output items

`NODE_NO`

Displays the node number of the HADB server.

`NODE_TYPE`

Displays the HADB server's node type:

- `MASTER`: Master node
- `SLAVE`: Slave node

If the `STATUS` column displays `ABORT`, the `NODE_TYPE` column displays the node type immediately before the HADB server terminated abnormally. Therefore, `MASTER` might be output more than once to the `NODE_TYPE` column.

`SVID`

Displays the process ID of the HADB server.

If the HADB server is stopped, nothing is displayed.

`IP-ADDRESS`

Displays the IP address of the HADB server.

This is the IP address for the host name specified in the `adb_sys_multi_node_info` operand in the server definition.

`STATUS`

Displays the status of the HADB server:

- ACTIVE: Running
- STOP: Stopped
- ABORT: Terminated abnormally

START_TIME

Displays the time the HADB server started.

If the HADB server is stopped, nothing is displayed.

28.4 Return codes

For details about the return codes of the `adbls -d node` command, see 22.4 Return codes.

28.5 Notes

For notes about the `adbls -d node` command, see 22.5 Notes.

29 adbls -d shm (Display Shared Memory Information)

This chapter explains the functionality provided by the `adbls -d shm` command and how to use the command.

29.1 Description

The `adbls -d shm` command displays information about the shared memory that the HADB server is using.

When the multi-node function is being used, only the information about the node where the command is executed is displayed.

29.2 Specification format for the adbls -d shm command

This section explains the specification format and options for the `adbls -d shm` command.

The following OS users can execute the `adbls -d shm` command:

- HADB administrator
- OS users who belong to the HADB administrators group

29.2.1 Explanation of the specification format and options

(1) Specification format

```
adbls -d shm [-a] [-t] [-c]
```

(2) Explanation of options

- `-a`

  Specifies that information about the shared memory being used by the HADB server is to be displayed separately for each shared memory segment.

- `-t`

  Suppresses display of the title line. For example, if you need to regularly redirect information to a file, you can specify this option to prevent duplicate title lines.

- `-c`

  Displays comma delimited values. For example, if you want to use spreadsheet software to read an output file to which data has been redirected, you can specify this option to output information delimited by commas.

  If this option is specified, the following types of information are enclosed in double quotation marks (`"`).

  - Character string in the *title of an output item*
  - *Character string* information in an output item

  Numeric value information in an output item is not enclosed in double quotation marks (`"`). Double quotation marks are not used if an output item is empty.

29.3 Examples

This section explains how to use the `adbls -d shm` command to display information about the shared memory.

The items that are output depend on whether the `-a` option is specified.

29.3.1 Displaying information about the shared memory (when the -a option is omitted)

This example executes the `adbls -d shm` command to display information about the shared memory.

```
adbls -d shm

SHM_NUMBER    SHM_SIZE
7             250626988
```

Explanation of the output items

`SHM_NUMBER`

Displays the number of shared memory segments that the HADB server is using.

`SHM_SIZE`

Displays the total size of the shared memory (in bytes) that the HADB server is using.

29.3.2 Displaying information about the shared memory (when the -a option is specified)

This example executes the `adbls -d shm -a` command to display information about the shared memory.

```
adbls -d shm -a

SYSID    SHMID     SIZE
1        720906    229376
4        753675    42845080
5        786444    24
6        819213    143359
7        851982    201326591
8        884751    4198399
9        917520    1884159
```

Explanation of the output items

`SYSID`

Displays the sequence number assigned to each shared memory segment that the HADB server is using.

`SHMID`

Displays the ID of each shared memory segment.

`SIZE`

Displays the size of each shared memory segment.

29.4 Return codes

For details about the return codes of the `adbls -d shm` command, see 22.4 Return codes.

29.5 Notes

For notes about the `adbls -d shm` command, see 22.5 Notes.

30 adbls -d srv (Display the HADB Server Status)

This chapter explains the functionality provided by the `adbls -d srv` command and how to use the command.

30.1 Description

The `adbls -d srv` command displays the status of the HADB server.

When the multi-node function is being used, only the information about the node where the command is executed is displayed.

30.2 Specification format for the adbls -d srv command

This section explains the specification format and options for the `adbls -d srv` command.

The following OS users can execute the `adbls -d srv` command:

- HADB administrator
- OS users who belong to the HADB administrators group

30.2.1 Explanation of the specification format and options

(1) Specification format

```
adbls -d srv [-t] [-c]
```

(2) Explanation of options

- `-t`

  Suppresses display of the title line. For example, if you need to regularly redirect information to a file, you can specify this option to prevent duplicate title lines.

- `-c`

  Displays comma delimited values. For example, if you want to use spreadsheet software to read an output file to which data has been redirected, you can specify this option to output information delimited by commas.

  If this option is specified, the following types of information are enclosed in double quotation marks (`"`).

  - Character string in the *title of an output item*
  - *Character string* information in an output item

  Numeric value information in an output item is not enclosed in double quotation marks (`"`). Double quotation marks are not used if an output item is empty.

30.3 Examples

These examples display the status of the HADB server.

Example 1 (When the HADB server is running in normal mode)

```
adbls -d srv

SVID   STATUS   START_TIME             MSGLOG_STATUS
5360   ACTIVE   2017/03/02 09:31:04    NORMAL
```

Example 2 (When the HADB server is inactive)

```
adbls -d srv

SVID   STATUS   START_TIME             MSGLOG_STATUS
       STOP                            NORMAL
```

Explanation of the output items

`SVID`

Displays the process ID of the HADB server.

When the HADB server is inactive, the command displays nothing.

`STATUS`

Displays the status of the HADB server:

- `ACTIVE`: Running (normal mode)
- `STOP`: Inactive
- `STARTING`: Starting
- `STOPPING`: Terminating
- `ABORT`: Abnormally terminated

  See *Steps to take when the HADB server terminated abnormally* in *Error-handling flow* in *Error Handling* in the *HADB Setup and Operation Guide*.
- `QUIESCE`: Running (quiescence mode)
- `OFFLINE`: Running (offline mode)
- `MAINTNCE`: Running (maintenance mode)
- `CHGMODE`: Running (changing the HADB server operation mode)
- `STOPWAIT`: Termination standby processing
- `COREDUMP`: Outputting the core file
- `FORCE`: Forcibly terminated

`START_TIME`

Displays the date and time the HADB server started.

If the HADB server is inactive, the command displays nothing.

`MSGLOG_STATUS`:

Displays status information for the message log file of the HADB server.

- `NORMAL`: Normal status

- `FALLBACK`: Fall-back mode

  In this mode, because the HADB server detected a shortage of free space on the disk that contains the message log file, message log data is output as much as possible according to free disk space. In this case, you need to return the message log file to the normal status. For details about how to return the message log file to the normal status, see *Releasing the message log file from fall-back mode* in the *HADB Setup and Operation Guide*.

30.4 Return codes

For details about the return codes of the `adbls -d srv` command, see 22.4 Return codes.

30.5 Notes

For notes about the `adbls -d srv` command, see 22.5 Notes.

31 adbls -d thd (Display the Thread Status)

This chapter explains the functionality provided by the `adbls -d thd` command and how to use the command.

31.1 Description

The `adbls -d thd` command displays information about the real threads and pseudo-threads.

When the multi-node function is being used, only the information about the node where the command is executed is displayed.

31.2 Specification format for the adbls -d thd command

This section explains the specification format and options for the `adbls -d thd` command.

The following OS users can execute the `adbls -d thd` command:

- HADB administrator
- OS users who belong to the HADB administrators group

31.2.1 Explanation of the specification format and options

(1) Specification format

```
adbls -d thd [-t] [-c]
```

(2) Explanation of options

- `-t`

  Suppresses display of the title line. For example, if you need to regularly redirect information to a file, you can specify this option to prevent duplicate title lines.

- `-c`

  Displays comma delimited values. For example, if you want to use spreadsheet software to read an output file to which data has been redirected, you can specify this option to output information delimited by commas.

  If this option is specified, the following types of information are enclosed in double quotation marks (`"`).

  - Character string in the *title of an output item*
  - *Character string* information in an output item

  Numeric value information in an output item is not enclosed in double quotation marks (`"`). Double quotation marks are not used if an output item is empty.

31.3 Example

This example displays information about the real threads and pseudo-threads.

```
adbls -d thd

THREAD-NO           T-ID(OS)            TOTAL_PSEUDO_THREAD TYPE    CNUMBER    SQL_ID              CLIENT_GROUP
1                   4102559552          6                   SYSTEM
2                   0951879424          6                   SYSTEM
3                   0962369280          6                   SYSTEM
4                   2897991424          6                   SYSTEM
5                   3882063616          6                   CONNECT
6                   3867367168          6                   CONNECT
7                   3852678912          6                   CONNECT
8                   3837990656          6                   CONNECT
9                   3619669760          6                   CONNECT
10                  3604981504          6                   CONNECT 6                              GROUP01
11                  3590293248          6                   CONNECT
12                  3575604992          6                   CONNECT
13                  3221223168          6                   CONNECT
14                  3206526720          6                   CONNECT
15                  3191838464          6                   CONNECT
16                  2942297856          6                   PROCESS
17                  1912383232          6                   PROCESS
18                  0972859136          6                   PROCESS
19                  0033335040          6                   PROCESS 6          1                   GROUP01
20                  3388778240          6                   PROCESS
21                  2449254144          6                   PROCESS
22                  1509730048          6                   PROCESS
23                  0570205952          6                   PROCESS
24                  3925649152          6                   PROCESS
25                  2986125056          6                   PROCESS 6          1                   GROUP01
```

Explanation of the output items

`THREAD-NO`

Displays the real thread numbers.

`T-ID(OS)`

Displays the real thread IDs (OS).

The value having the same format as the real thread ID output to the message log file is displayed.

`TOTAL_PSEUDO_THREAD`

Displays the number of pseudo-threads that are running within each real thread. Pseudo-threads used in HADB are included.

`TYPE`

Displays the real thread type. One of the following is displayed:

- `SYSTEM`: Thread being used by the system
- `CONNECT`: Connection thread (including a connection thread being used by the system)
- `PROCESS`: Processing thread

`CNUMBER`

Displays the connection sequence number of the processing that is using the real thread.

`SQL_ID`

Displays the SQL statement sequence number of the SQL statement that is running using the real thread.

`CLIENT_GROUP`

Displays the name of the client group to which the HADB client that executed the SQL statement belongs. If the command or HADB client that executed the SQL statement belongs to a command group, `command` is displayed.

If the command or HADB client that executed the SQL statement does not belong to any group, nothing is displayed.

31.4 Return codes

For details about the return codes of the `adbls -d thd` command, see 22.4 Return codes.

31.5 Notes

For notes about the `adbls -d thd` command, see 22.5 Notes.

32 adbls -d ver (Display the HADB Server's Version Information)

This chapter explains the functionality provided by the `adbls -d ver` command and how to use the command.

32.1 Description

The `adbls -d ver` command displays the HADB server's version information.

When the multi-node function is being used, only the information about the node where the command is executed is displayed.

32.2 Specification format for the adbls -d ver command

This section explains the specification format for the `adbls -d ver` command.

```
adbls -d ver
```

The following OS users can execute the `adbls -d ver` command:

- HADB administrator
- OS users who belong to the HADB administrators group

32.3 Example

This example display the HADB server's version information.

```
adbls -d ver

Hitachi Advanced Data Binder 02-01 Nov  5 2013 17:03:58
```

The underlined part displays the HADB server's version.

32.4 Return codes

For details about the return codes of the `adbls -d ver` command, see 22.4 Return codes.

32.5 Notes

For notes about the `adbls -d ver` command, see 22.5 Notes.

33 adbmergechunk (Merge Chunks)

This chapter explains the functionality provided by the `adbmergechunk` command and how to use the command.

Note that the term *tables* in this chapter means base tables.

33.1 Description

This section explains the functionality provided by the `adbmergechunk` command and the threads it uses.

When executed, the `adbmergechunk` command establishes a connection to the HADB server.

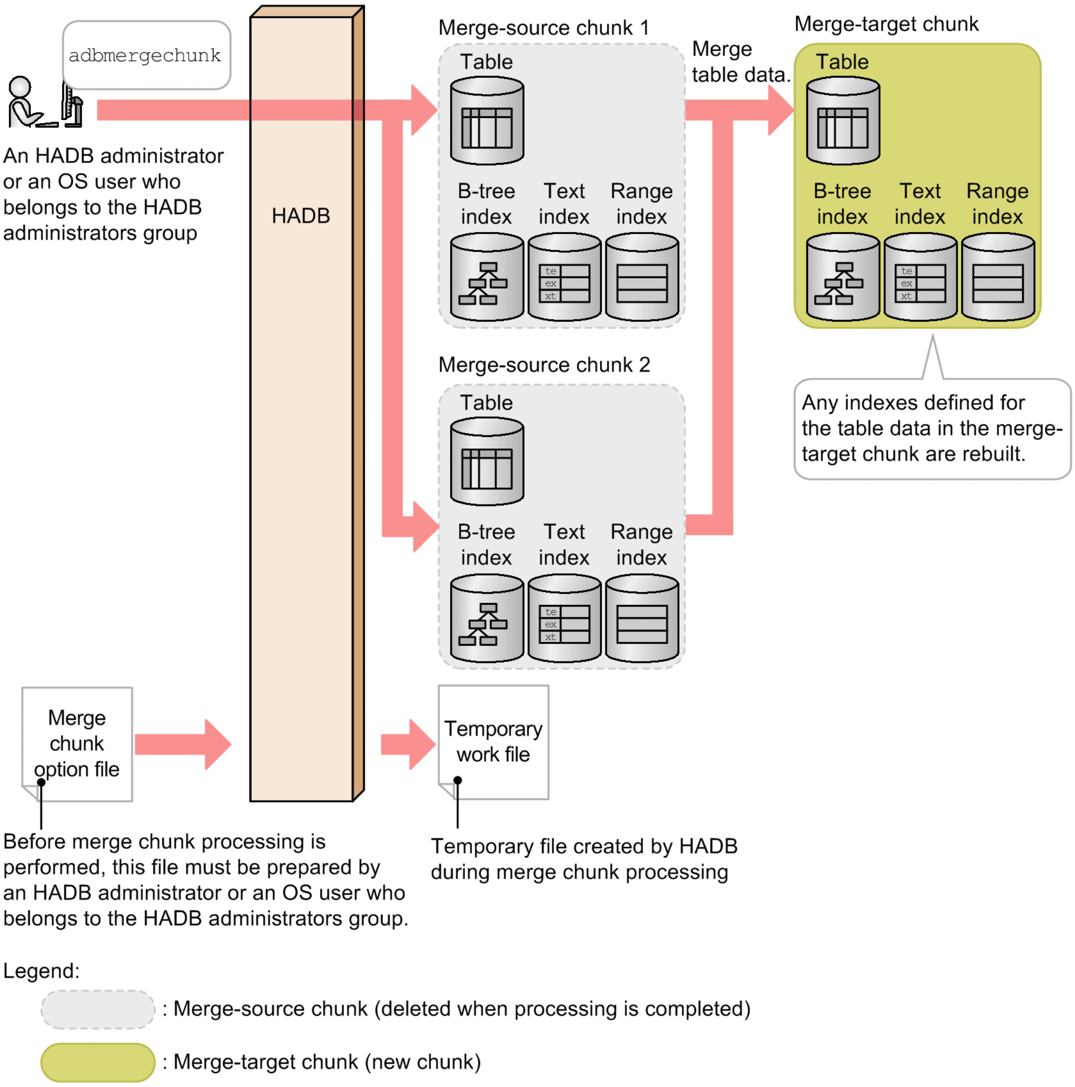
33.1.1 Functional overview

The `adbmergechunk` command is used to merge multiple chunks in multi-chunk tables into one new chunk. This is called *merge chunk processing*.

You can reduce the number of chunks in a table by performing merge chunk processing.

The following figure provides an overview of merge chunk processing.

Figure 33-1: Overview of merge chunk processing


adbmergechunk
An HADB administrator or an OS user who belongs to the HADB administrators group
HADB
Merge-source chunk 1
Table
B-tree index
Text index
Range index
Merge table data.
Merge-target chunk
Table
B-tree index
Text index
Range index
Any indexes defined for the table data in the merge-target chunk are rebuilt.
Merge-source chunk 2
Table
B-tree index
Text index
Range index
Merge chunk option file
Temporary work file
Before merge chunk processing is performed, this file must be prepared by an HADB administrator or an OS user who belongs to the HADB administrators group.
Temporary file created by HADB during merge chunk processing
Legend:
: Merge-source chunk (deleted when processing is completed)
: Merge-target chunk (new chunk)

Explanation:

The `adbmergechunk` command merges table data that exists in any number of chunks (merge-source chunks) into a single new chunk (merge-target chunk). The command then rebuilds the indexes defined for the table data in the merge-target chunk. Finally, the command deletes the merge-source chunks. This allows you to use a single chunk to manage table data and indexes that had previously been managed in multiple chunks.

> **Important**
>
> You cannot merge chunks that are in a different status (chunks in normal status and chunks in wait status).

> **Note**
>
> As the number of chunks increases, the performance of retrievals using B-tree or text indexes might decrease. You can avoid such deterioration of retrieval performance by merging chunks and reducing their number.

33.1.2 Relationships among threads used by the adbmergechunk command

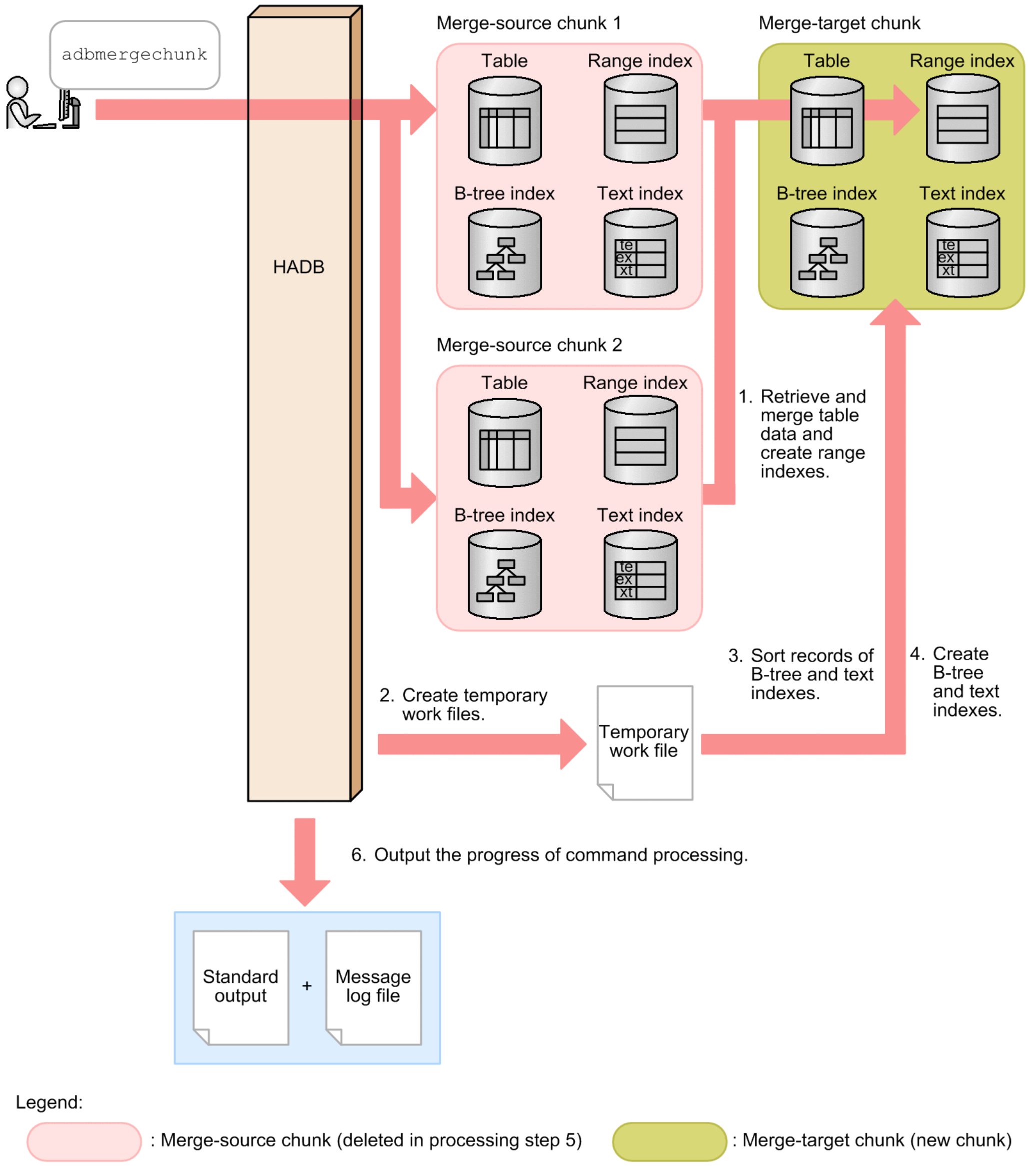
The `adbmergechunk` command uses processing real threads in the following processing:

1. Retrieving and merging table data and creating range indexes
   One or more processing real threads are used.
2. Creating temporary work files
   One or more processing real threads are used.
3. Sorting records of B-tree and text indexes
   One or more processing real threads are used.
4. Creating B-tree and text indexes
   One or more processing real threads are used.
5. Deleting merge-source chunks
   One or more processing real threads are used.
6. Outputting the progress of command processing
   One processing real thread is used.

Steps 1 and 2 are performed at the same time. Then steps 3 to 5 are performed in sequence. Step 6 is always performed while steps 1 to 5 are in progress.

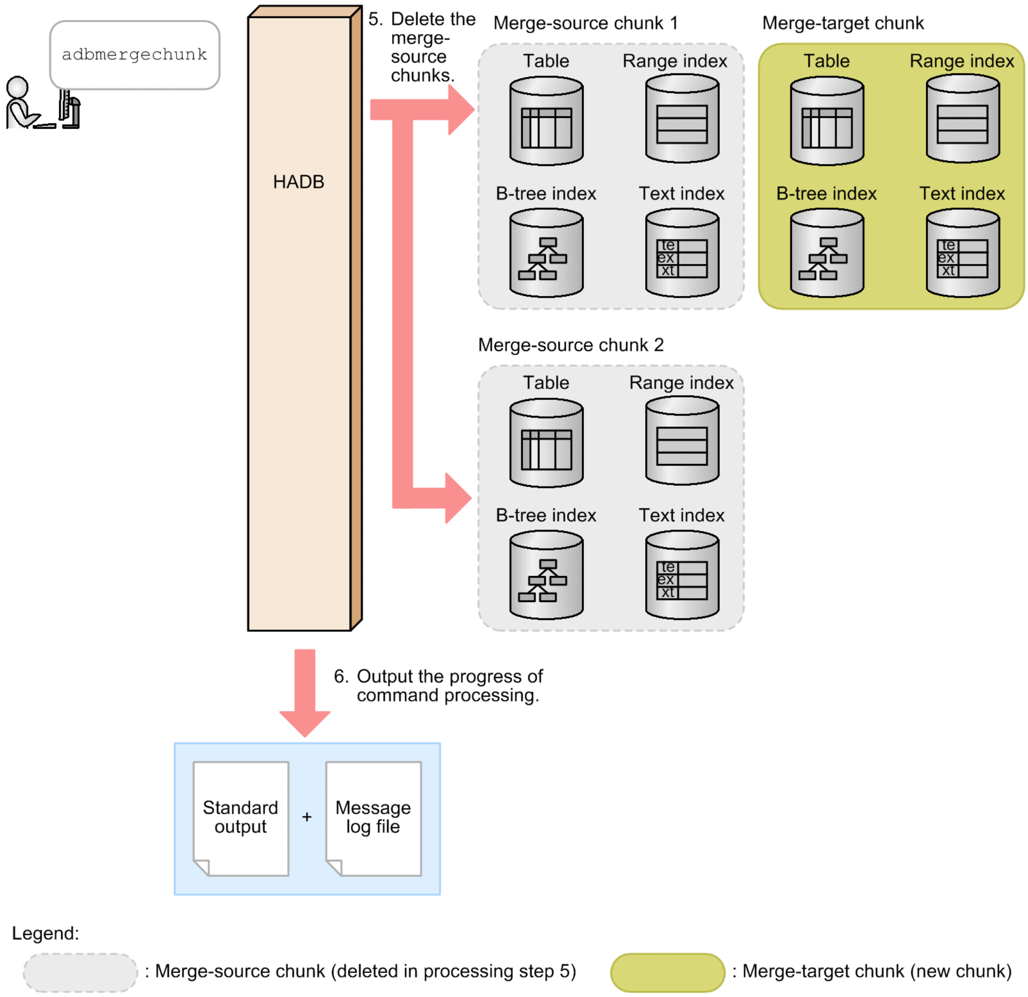
The following two figures show the processing of the `adbmergechunk` command using processing real threads. The preceding numbers correspond to the numbers in the figure.

Figure 33-2: Processing of the adbmergechunk command using processing real threads (1)



adbmergechunk
HADB
Merge-source chunk 1
Table
Range index
B-tree index
Text index
Merge-target chunk
Table
Range index
B-tree index
Text index
Merge-source chunk 2
Table
Range index
B-tree index
Text index
1. Retrieve and merge table data and create range indexes.
2. Create temporary work files.
Temporary work file
3. Sort records of B-tree and text indexes.
4. Create B-tree and text indexes.
6. Output the progress of command processing.
Standard output
+
Message log file
Legend:
: Merge-source chunk (deleted in processing step 5)
: Merge-target chunk (new chunk)

Figure 33-3: Processing of the adbmergechunk command using processing real threads (2)



The number of processing real threads used by the `adbmergechunk` command for each step is automatically calculated from the *number of processing real threads to be used by the* `adbmergechunk` *command*. For details about the *number of processing real threads to be used by the* `adbmergechunk` *command*, see the description of the merge chunk option adb_mergechunk_rthd_num in (2) Explanation of specification format in 33.2.2 Format of merge chunk options.

33.2 Specification format for the adbmergechunk command

This section explains the specification format for the `adbmergechunk` command and how to specify the merge chunk options.

The following OS users can execute the `adbmergechunk` command:

- HADB administrator
- OS users who belong to the HADB administrators group

33.2.1 Explanation of the specification format and options

(1) Specification format

```
adbmergechunk
  -u authorization-identifier
  [-p password]
  [-g output-interval-for-merge-chunk-processing-progress-messages]
  [-w {temporary-work-file-storage-directory-name
       |name-of-directory-path-file-specifying-temporary-work-file-storage-locations}
]
  [-z merge-chunk-option-file-name]
  [-m comment-to-be-set-in-chunk]
  [--purge-chunk {WAIT|NOWAIT}]
  -c {specification-of-merge-source-chunk-ID[,specification-of-merge-source-chunk-ID]
...|ALL}
  table-to-be-processed
```

Important

You must specify the table to be processed as the last option that is specified.

(2) Explanation of options

- `-u` *authorization-identifier*

  ~<character string>((1 to 100 bytes))

  Specifies the authorization identifier of the HADB user who executes the `adbmergechunk` command. To execute the `adbmergechunk` command, the user must have the following two privileges:

  - `CONNECT` privilege
  - `MERGE CHUNK` privilege for the table to be processed

  For this option, specify an authorization identifier in the range from 1 to 100 bytes. Note that the byte count (1 to 100 bytes) does not include the double quotation marks used to enclose the authorization identifier.

  Important

  If the character string used as the authorization identifier includes any lowercase letter or a backslash (`\`), make sure that you check the rules for specifying authorization identifiers. For the rules for specifying authorization identifiers, see 1.4.2 Rules for specifying authorization identifiers and passwords.

- `-p` *password*

  ~<character string>((1 to 255 bytes))

  Specifies the password for the authorization identifier that is specified in the `-u` option.

  > **Important**
  >
  > If the password includes a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), make sure that you check the rules for specifying passwords. For the rules for specifying passwords, see 1.4.2 Rules for specifying authorization identifiers and passwords.

  If the `adbmergechunk` command is executed with this option omitted, a message prompting the user to enter a password is displayed. In an environment in which a password cannot be entered from the standard input, such as when the command is executed in the background, make sure that you do not omit this option.

- `-g` *output-interval-for-merge-chunk-processing-progress-messages*

  ~<integer>((0 to 1,000,000,000))<<0>>(in 1,000 rows)

  Specifies the interval (in units of 1,000 rows) at which the merge chunk processing progress message is to be output. When this option is specified, the command outputs the number of rows of data that have been retrieved from the table whose indexes are being rebuilt to the progress message, in order to indicate the progress of the merge chunk processing.

  For example, if `2` is specified in this option, the `KFAA80205-I` progress message is output each time 2,000 rows of table data have been retrieved to rebuild indexes.

  If this option is omitted or `0` is specified in this option, the progress message is not output.

- `-w` `{`*temporary-work-file-storage-directory-name* | *name-of-directory-path-file-specifying-temporary-work-file-storage-locations*`}`

  ~<OS path name>((2 to 518 bytes))

  Specifies a storage location for the temporary work files that are created during merge chunk processing, expressed as either of the following:

  1. Absolute path name of the directory that stores temporary work files
  2. Absolute path name of a file (directory path file) that specifies directories for storing temporary work files

  If frequent input/output operations are performed on the temporary work files stored under a single directory, performance might be adversely affected. We recommend that you use method 2 and store temporary work files in directories on multiple disks to distribute the workload of input/output operations on the disk. The following shows an example specification using method 2 (specifying a directory path file).

  Example of directory path file

  ```
  /mnt/diska/wwwww
  /mnt/diskb/xxxxx
  /mnt/diskc/yyyyy
  /mnt/diskd/zzzzz
  ```

  The following is a guideline for the number of directories to be specified:

  Guideline-for-number-of-directories-to-be-specified = *A* × *B*

  A: Maximum number of scan processing threads

  Use the following formula to determine the value:

↓ (Value of the merge chunk option adb_mergechunk_rthd_num - 1) ÷ 2↓

B: Number of B-tree and text indexes that are defined for the table to be processed

If you do not know the number of B-tree and text indexes, check the number of B-tree and text indexes that are defined for the table by using the following SQL statement:

```
SELECT "N_INDEX"-"N_RANGE_INDEX"
    FROM "MASTER"."SQL_TABLES"
        WHERE "TABLE_SCHEMA"='authorization-identifier'
           AND "TABLE_NAME"='name-of-table-to-be-processed'
```

- A maximum of 255 directories can be specified in a directory path file.
- The directories specified in the directory path file are used as the storage locations of temporary work files in the order specified. When the last directory specified in the directory path file is used (in this example, `/mnt/diskd/zzzzz`), the first directory (in this example, `/mnt/diska/wwwww`) is then used again.

The following specification rules apply to this option (for both methods 1 and 2):

- Do not specify a forward slash (`/`) at the end of a path name.

For details about how to estimate the sizes of the temporary work files that will be created under the directories specified in the `-w` option, see *Estimating the size of the temporary work file for executing the adbmergechunk command* in *Estimating the size of the temporary work file for executing a command* in *Preparing Resources* in the *HADB Setup and Operation Guide*. If this option is omitted, temporary work files are created under `$DBDIR/ADBWORK`.

It is advisable to provide a dedicated directory to store temporary work files. If a directory storing other files is used to store temporary work files, data in the existing files might be corrupted. Also make sure that symbolic links and relative paths are specified correctly. For details about the files whose names must differ from the temporary work file storage directory name, see 33.6 Notes.

If the disk that stores temporary work files runs out of free space, specify another disk with a larger capacity as the storage destination. For details about the actions to take if the disk that stores temporary work files runs out of free space, see *Steps to take in the event of a shortage of disk space for storing temporary work files during command execution* in *Command-related problems* in *Troubleshooting* in the *HADB Setup and Operation Guide*.

- `-z` *merge-chunk-option-file-name*

  ~<OS path name>((2 to 510 bytes))

  Specifies the absolute path name of the merge chunk option file that contains the merge chunk options. If this option is omitted, all merge chunk options are set to their default values because no merge chunk option file is specified.

  For details about the merge chunk options, see 33.2.2 Format of merge chunk options.

  Make sure that the merge chunk option name is unique from all other file names. If the merge chunk option file has the same name as another file, data in the file might become corrupted. For details about the files whose names must differ from the merge chunk option file name, see 33.6 Notes.

- `-m` *comment-to-be-set-in-chunk*

  ~<character string>((1 to 1,024 bytes))

  Specifies a comment that is to be set in the merge-target chunk.

  The following rules apply to specifying this information:

  - The specified comment is case sensitive.
  - If the comment contains a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), specify the escape character (`\`) immediately before that character. Alternatively, enclose the entire comment in single quotation marks (`'`).

    Example 1: `-m abc\|def`

    Example 2: `-m 'abc|def'`

  In both examples, `abc|def` is set as the comment.

  - If a character string is enclosed in double quotation marks (`"`) and then the entire string is enclosed in single quotation marks (`'`), both the character string and the double quotation marks are treated as a comment.

    Example: `-m '"abcdef"'`

    In this example, `"abcdef"` is set as the comment.

  - The character encoding specified in the HADB server's `ADBLANG` environment variable is used for the character string set as a comment.

- `--purge-chunk {`<u>`WAIT`</u>`|NOWAIT}`

  Specifies whether the merge-source chunks are to be deleted after the merge-target chunk is created if processing of any another SQL statement or command references the table subject to merge chunk processing.

  - **If `WAIT` is specified or this option is omitted**

    Merge-source chunk deletion processing waits for the completion of processing of other SQL statements and commands that reference the table subject to merge chunk processing. The merge-source chunk deletion processing starts when there is no more processing that references the table subject to merge chunk processing. After the merge-source chunks are deleted, the `adbmergechunk` command terminates.

  - **If `NOWAIT` is specified**

    Merge-source chunk deletion processing is skipped. Therefore, the `adbmergechunk` command terminates while the merge-source chunks remain in the deletion-pending status.

    In this case, you need to delete the merge-source chunks manually. See the `KFAA96785-E` message to check the chunk IDs of the chunks to be deleted, and then execute the `PURGE CHUNK` statement. If you do not delete the merge-source chunks, you cannot execute the next `adbmergechunk` command.

  Note that if processing of any other SQL statement or command does not reference the table subject to merge chunk processing, the merge-source chunks are deleted after the merge-target chunk is created regardless of the `--purge-chunk` option specification.

  For details about considerations when specifying the `--purge-chunk` option, see *Points to consider when specifying the --purge-chunk option of the adbmergechunk command* in *Merging chunks (to reduce the number of chunks)* in the *HADB Setup and Operation Guide*.

- `-c {`*specification-of-merge-source-chunk-ID*`[,`*specification-of-merge-source-chunk-ID*`]...|ALL}`

  ~<integer> ((1 to 9,223,372,036,854,775,807))

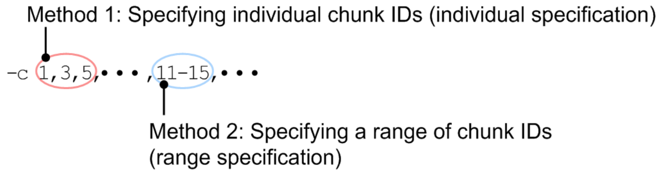
  Specifies the chunk IDs of the merge-source chunks that are to be processed by merge chunk processing.

  When `ALL` is specified, the command merges all chunks in the target table into a single new chunk. When merge-source chunk IDs are specified, the command merges only the chunks with the specified IDs into a new chunk.

  > **Important**
  >
  > You cannot merge chunks that are in a different status (chunks in normal status and chunks in wait status).

  When you specify merge-source chunk IDs, specify at least two chunk IDs. There are two methods that you can use to specify the merge-source chunk IDs, which can be specified at the same time as shown in the following figure.

Figure 33-4: Specification example of merge-source chunk IDs in the -c option


Method 1: Specifying individual chunk IDs (individual specification)
-c 1,3,5,...,11-15,...
Method 2: Specifying a range of chunk IDs
(range specification)

- **Specifying individual chunk IDs (individual specification)**
  If you specify the IDs of individual merge-source chunks, separate the chunk IDs with the comma (`,`).

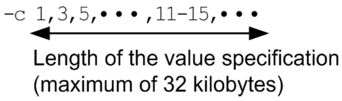
> **Important**
>
> There must be no spaces before or after a comma (,).

- **Specifying a range of chunk IDs (range specification)**
  If you specify a range of merge-source chunk IDs, use the format *smallest-chunk-ID-in-the-range*`-`*largest-chunk-ID-in-the-range* (connect the two chunk IDs with a hyphen (`-`)).

> **Important**
>
> There must be no spaces before or after the hyphen (`-`). The chunk ID specified to the right of the hyphen must be greater than the chunk ID specified to the left of the hyphen.

The following rules apply to specifying this information:

- Do not specify both `ALL` and chunk IDs.
- If the target table does not include a specified chunk ID, the command ignores that chunk ID and merges only those specified chunks whose IDs do exist in the target table.
- You can specify maximum of 30,000 chunk IDs. The following explains how chunk IDs are counted:
  • For a range specification, all chunk IDs in the specified range are counted.
  • If any duplicate chunk IDs are specified, the duplicate chunk IDs are counted.
  Specification examples of merge-source chunk IDs and how to count them
  Example 1: `-c 1,3,5`: The command assumes that three chunk IDs are specified (`1`, `3`, and `5`).
  Example 2: `-c 1,3,5-8,10`: The command assumes that seven chunk IDs are specified (`1`, `3`, `5`, `6`, `7`, `8`, and `10`).
  Example 3: `-c 1,3,5,5-8,10`: The command assumes that eight chunk IDs are specified (`1`, `3`, `5`, `5`, `6`, `7`, `8`, and `10`).
- The maximum permitted length of the entire value specified in the `-c` option is 32 kilobytes. If the length would exceed 32 kilobytes, use range specifications or execute the `adbmergechunk` command more than once.



- For details about specifications that result in an error and specification examples, see (3) Errors that can be caused by the -c option.

The following explains how to determine the chunk IDs that exist in a table to be processed:

- Searching system tables for `STATUS_CHUNKS`
  Search system tables for `STATUS_CHUNKS`, and check the chunk IDs. For details, see *Checking the information about all chunks in a table based on a table name* in *Searching system tables* in *System Tables* in the *HADB Setup and Operation Guide*.
- Executing the `adbdbstatus` command
  Execute the following command to output the usage information for DB areas, tables, and indexes, and then check the chunk IDs (`Chunk_ID`) (the results are output in CSV format). For details about the items that are output as usage information for DB areas, tables, and indexes, see 13.7.2 List of items that are output as usage information for DB areas, tables, and indexes.

```
adbdbstatus -d used -c table -n name-of-table-to-be-processed
```

Specify *name-of-table-to-be-processed* in the format *schema-name*.*table-identifier*.

- *table-to-be-processed*

  Specifies the multi-chunk table that is to be processed by merge chunk processing.

  The following rules apply to specifying this information:

  - Specify in the format *schema-name*.*table-identifier*.

    If the user owns this table, the schema name can be omitted. If another HADB user owns this table, schema name must be specified.

  - For details about the specification rules that apply when a schema name or table identifier includes any lowercase letter or backslash (\), see 1.4.3 Table name specification rules.

  - The user can specify only a table for which the user has the `MERGE CHUNK` privilege.

  - Tables other than multi-chunk tables cannot be specified.

  - A viewed table cannot be specified.

(3) Errors that can be caused by the -c option

The following are two possible causes of errors in the `adbmergechunk` command that can be caused by the `-c` option's specification:

- The specification format of the `-c` option is invalid.
- An invalid chunk ID is specified in the `-c` option.

In either case, correct the specification, and then re-execute the `adbmergechunk` command. For details about the `-c` option, see -c in (2) Explanation of options.

(a) When the specification format of the -c option is invalid

The table below lists the message IDs that are displayed when the specification format of the `-c` option is invalid and describes the possible errors. If any of these errors occurs, correct the error, and then re-execute the `adbmergechunk` command.

Table 33-1: Message IDs that are displayed when the specification format of the `-c` option is invalid and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	`KFAA50225-E`	The number of specified chunk IDs exceeds 30,000.	`-c 1-30001`	`-c 1-30000`
2		`ALL` and merge-source chunk IDs are both specified.	`-c 1,3,ALL,10`	• `-c ALL` • `-c 1,3,10`
3		The same chunk ID is specified on both sides of a hyphen (-).	`-c 3-3,7`	`-c 3-5,7`
4		The value of the chunk ID on the right side of a hyphen (-) is less than the value of the chunk ID on the left side.	`-c 10-3`	`-c 3-10`
5	`KFAA50296-E`	Only one chunk ID is specified (including when only one chunk ID is	• `-c 3` • `-c 3,3,3`	`-c 3,4`

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
		specified and it is specified more than once).		
6	• KFAA90002-E • KFAA96780-I	There is a space between a chunk ID and a comma.	`-c 1,3,Δ10`	`-c 1,3,10`
7		There is a space between a chunk ID and a hyphen (`-`).	`-c 1,3Δ-Δ10`	`-c 1,3-10`
8	KFAA90003-E	The length of the specified value exceeds 32 kilobytes.	`-c 1,2,••• ,`*xxxxx* ↔ 33 kilobytes or greater	`-c 1-`*xxxxx* Change from individual specification to range specification (use a hyphen (`-`), not a comma (`,`))

Legend:

Δ: Space

(b) When an invalid chunk ID is specified in the -c option

The table below lists the message IDs that are displayed when an invalid chunk ID is specified in the `-c` option and describes the possible errors. If any of these errors occurs, correct the error, and then re-execute the `adbmergechunk` command.

Table 33-2: Message IDs that are displayed when an invalid chunk ID is specified in the -c option and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification and corrective action to take
1	KFAA50294-E	A chunk ID that does not exist in the target table is specified (a chunk ID that is outside the range from minimum chunk ID to maximum chunk ID).	**Only chunk IDs 3 and 6 exist in the target table:** • `-c 2-6` • `-c 6,7` • `-c 2-7`	**Only chunk IDs 3 and 6 exist in the target table:** • `-c 3,6` • `-c 3-6` Search the target table for its chunk IDs and specify chunk IDs that exist in the table.
2	KFAA50296-E	There is only one merge-source chunk because a chunk ID that does not exist in the target table is specified.	**Only chunk IDs 3 and 6 exist in the target table:** • `-c 3-5` • `-c 4,5`	**Only chunk IDs 3 and 6 exist in the target table:** • `-c 3,6` • `-c 3-6` Search the target table for its chunk IDs and correct the specification so that multiple merge-source chunks are specified.
3	KFAA50299-E	Among the merge-source chunks there are chunks in normal status intermixed with chunks in wait status.	**Chunks with IDs 1 and 2 are in normal status, and chunks with IDs 3 and 4 are in wait status:** • `-c 1,3` • `-c 1-3`	**Chunks with IDs 1 and 2 are in normal status, and chunks with IDs 3 and 4 are in wait status:** • `-c 1,2` • `-c 3-4` Correct the specification so that the merge-source chunks include

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification and corrective action to take
				only chunks in normal status or only chunks in wait status.
4		Among the merge-source chunks there are archived chunks intermixed with chunks that are not archived.	**Chunks with IDs 1 and 2 are archived, and chunks with IDs 3 and 4 are not archived:** • `-c 1,3` • `-c 1-3`	**Chunks with IDs 1 and 2 are archived, and chunks with IDs 3 and 4 are not archived:** • `-c 1,2` • `-c 3-4` Correct the specification so that the merge-source chunks include only archived chunks or only chunks that are not archived.

Note:

For details about how to search for the chunk IDs in the table to be processed, see -c in(2) Explanation of options.

33.2.2 Format of merge chunk options

(1) Specification format

```
[set adb_mergechunk_rthd_num = number-of-processing-real-threads-to-be-used-by-adbmer
gechunk-command]
[set adb_mergechunk_scan_buff_size = table-data-scan-buffer-size]
[set adb_mergechunk_buff_blk_num = number-of-buffer-blocks-for-merge-chunks]
[set adb_mergechunk_sort_buff_size = sort-buffer-size]
[set adb_mergechunk_dvix_rdbuff_size = B-tree-and-text-index-input-buffer-size]
[set adb_mergechunk_dvix_wtbuff_size = B-tree-and-text-index-output-buffer-size]
[set adb_mergechunk_txt_buff_size = text-index-creation-buffer-size]
```

The specification rules for these merge chunk options are the same as for the server definition. For details about the syntax rules for the server definition, see the topic *Syntax rules for the server definition* in the *HADB Setup and Operation Guide*.

(2) Explanation of specification format

- `adb_mergechunk_rthd_num` = *number-of-processing-real-threads-to-be-used-by-adbmergechunk-command*

  ~<integer>((3 to 4,096))

  Specifies the number of processing real threads to be used for executing the `adbmergechunk` command.

  Specification of this option is recommended. The `adbmergechunk` command uses the number of processing real threads specified by this option. Therefore, specify an appropriate number of processing real threads for use by the `adbmergechunk` command. For details about the processing of the `adbmergechunk` command that uses processing real threads and the number of processing real threads to be used for each step of processing, see 33.1.2 Relationships among threads used by the adbmergechunk command.

  If this option is omitted, the number of processing real threads to be used for executing the `adbmergechunk` command is determined by the `adb_sql_exe_max_rthd_num` operand in the server definition. The following table shows the relationship between the value of this option or the `adb_sql_exe_max_rthd_num` operand in the server definition and the number of processing real threads used for executing the command.

Table 33-3: Number of processing real threads used for executing the command

No.	Value in this option or in the server definition		Number of processing real threads used for executing the command
1	When this option is omitted	If the adb_sql_exe_max_rthd_num operand in the server definition is not specified	The default value of the adb_sql_exe_max_rthd_num operand in the server definition is used.
2		If a value in the range from 0 to 2 is specified in the adb_sql_exe_max_rthd_num operand in the server definition	--
3		If 3 or a greater value is specified for the adb_sql_exe_max_rthd_num operand in the server definition	The value specified for the adb_sql_exe_max_rthd_num operand in the server definition is used.
4	If 3 or a greater value is specified in this option		The value specified in this option is used.

Legend:

--: The adbmergechunk command results in an error.

Note:

For details about the adb_sql_exe_max_rthd_num operand in the server definition, see *Operands related to performance (set format)* in the *HADB Setup and Operation Guide*.

- adb_mergechunk_scan_buff_size = *table-data-scan-buffer-size*

  ~<integer>((1 to 2,048))<<8>> (megabytes)

  Specifies the size (in megabytes) of the buffer to be used for retaining table data retrieval results. A larger value can reduce overhead in retrieving table data.

  As a guideline, specify the value obtained from the following formula.

  Formula

  ```
  Key length of the B-tree indexes defined for the table to be processed × 100,000
  ```

  For details about how to determine the key length of a B-tree index, see *Determining the key length (KEYSZ) of a B-tree index* in *Estimating the size of the data DB area* in *Designing a Database* in the *HADB Setup and Operation Guide*.

- adb_mergechunk_buff_blk_num = *number-of-buffer-blocks-for-merge-chunks*

  ~<integer>((5 to 4,096))<<64>>

  Specifies the number of buffer blocks to be used for merge chunk processing. You can use the default value, but a larger value might improve the speed of merge chunk processing.

- adb_mergechunk_sort_buff_size = *sort-buffer-size*

  ~<integer>((1 to 2,048))<<256>> (megabytes)

  Consider specifying this option when B-tree or text indexes are defined for the table to be processed.

  Specifies the size (in megabytes) of the sort buffer to be used for creating B-tree and text indexes. Specify a large value if there is sufficient memory. Better performance can be expected as this option's value becomes larger.

- adb_mergechunk_dvix_rdbuff_size = *B-tree-and-text-index-input-buffer-size*

  ~<integer>((32 to 51,200))<<1,024>> (kilobytes)

  Consider specifying this option if B-tree or text indexes are defined for the table to be processed.

  Specifies the size (in kilobytes) of the input buffer to be used for creating B-tree and text indexes. Specify a large value if there is sufficient memory. Better performance can be expected as this option's value becomes larger.

- `adb_mergechunk_dvix_wtbuff_size` = *B-tree-and-text-index-output-buffer-size*

  ~<integer>((32 to 51,200))<<1,024>> (kilobytes)

  Consider specifying this option if B-tree or text indexes are defined for the table to be processed.

  Specifies the size (in kilobytes) of the output buffer to be used for creating B-tree and text indexes. Specify a large value if there is sufficient memory. Better performance can be expected as this option's value becomes larger.

- `adb_mergechunk_txt_buff_size` = *text-index-creation-buffer-size*

  ~<integer>((1 to 2,048))<<256>> (megabytes)

  Consider specifying this option if text indexes are defined for the table to be processed.

  Specifies the size (in megabytes) of the buffer to be used for creating text indexes.

  Increasing the value specified for this option often improves performance. However, the amount of buffer memory specified by this option will be used for all processing real threads on which text indexes will be built. Therefore, be careful when you increase the value specified for this option.

33.3 Examples

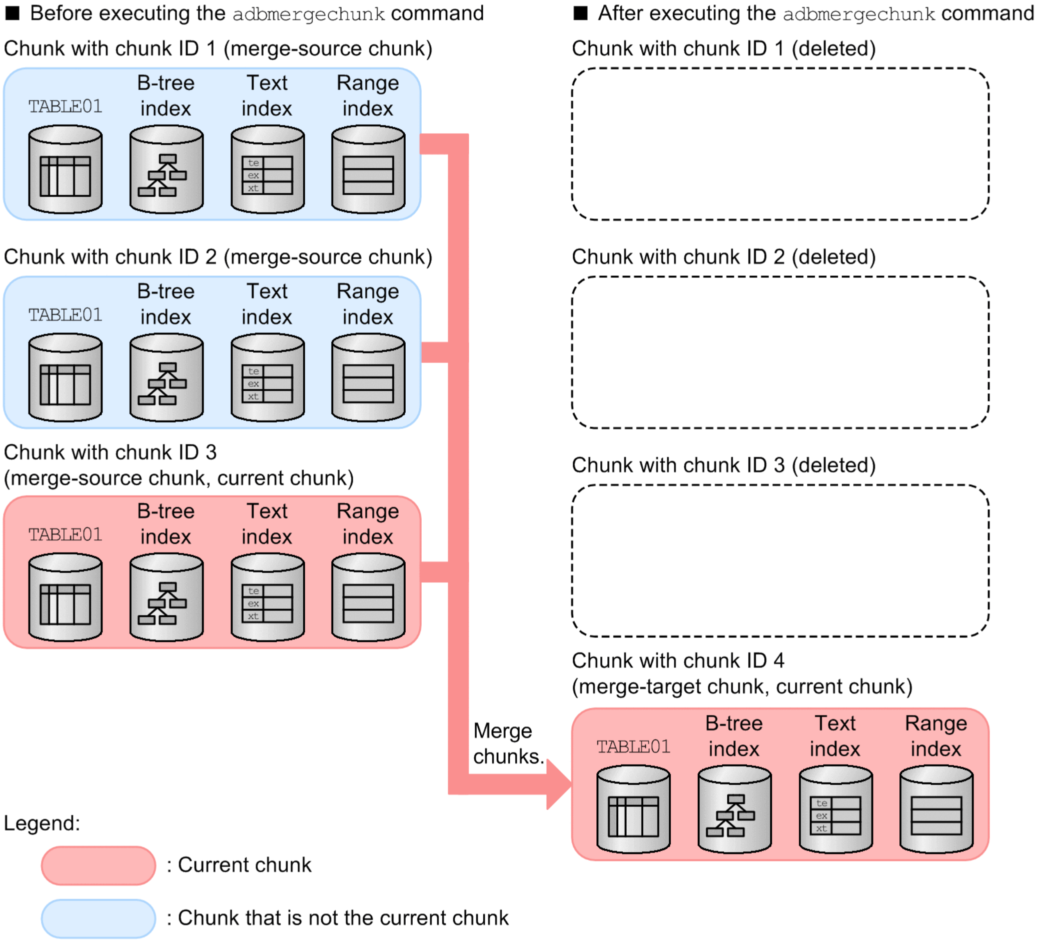
This section explains by way of examples how to use the adbmergechunk command to merge multiple chunks.

If the adbmergechunk command terminates abnormally during execution, take appropriate action based on the explanation in 33.5 Handling abnormal termination of the adbmergechunk command.

33.3.1 Merging all chunks in a table

This example merges under the following conditions all chunks in table TABLE01 into a new chunk:

1. CHUNK is specified when table TABLE01 is defined (table TABLE01 is a multi-chunk table).
2. B-tree, text, and range indexes are defined for table TABLE01.
3. Data was imported into table TABLE01 by applying the background-import facility twice.
4. Table TABLE01 has chunks with chunk IDs 1, 2, and 3.
5. The chunk ID of table TABLE01's current chunk is 3.


■ Before executing the adbmergechunk command
Chunk with chunk ID 1 (merge-source chunk)
TABLE01
B-tree index
Text index
Range index
Chunk with chunk ID 2 (merge-source chunk)
TABLE01
B-tree index
Text index
Range index
Chunk with chunk ID 3 (merge-source chunk, current chunk)
TABLE01
B-tree index
Text index
Range index
Merge chunks.
■ After executing the adbmergechunk command
Chunk with chunk ID 1 (deleted)
Chunk with chunk ID 2 (deleted)
Chunk with chunk ID 3 (deleted)
Chunk with chunk ID 4 (merge-target chunk, current chunk)
TABLE01
B-tree index
Text index
Range index
Legend:
: Current chunk
: Chunk that is not the current chunk

This example uses the following procedure to execute the adbmergechunk command.

Procedure

1. Create a merge chunk option file.

2. Execute the adbmergechunk command.

Using an example, the following subsections explain each of these steps.

(1) Creating a merge chunk option file

This example creates a merge chunk option file and specifies merge chunk options in that file.

Example of a merge chunk option file

```
set adb_mergechunk_rthd_num = 3
set adb_mergechunk_scan_buff_size = 2
      :
```

(2) Executing the adbmergechunk command

```
adbmergechunk -u ADBUSER01                                         ...1
              -p '#HelloHADB_01'                                   ...2
              -z /home/adbmanager/merge_file/mergechunk_opt.txt  ...3
              -m 'January 2014'                                    ...4
              -c ALL                                               ...5
              TABLE01                                              ...6
```

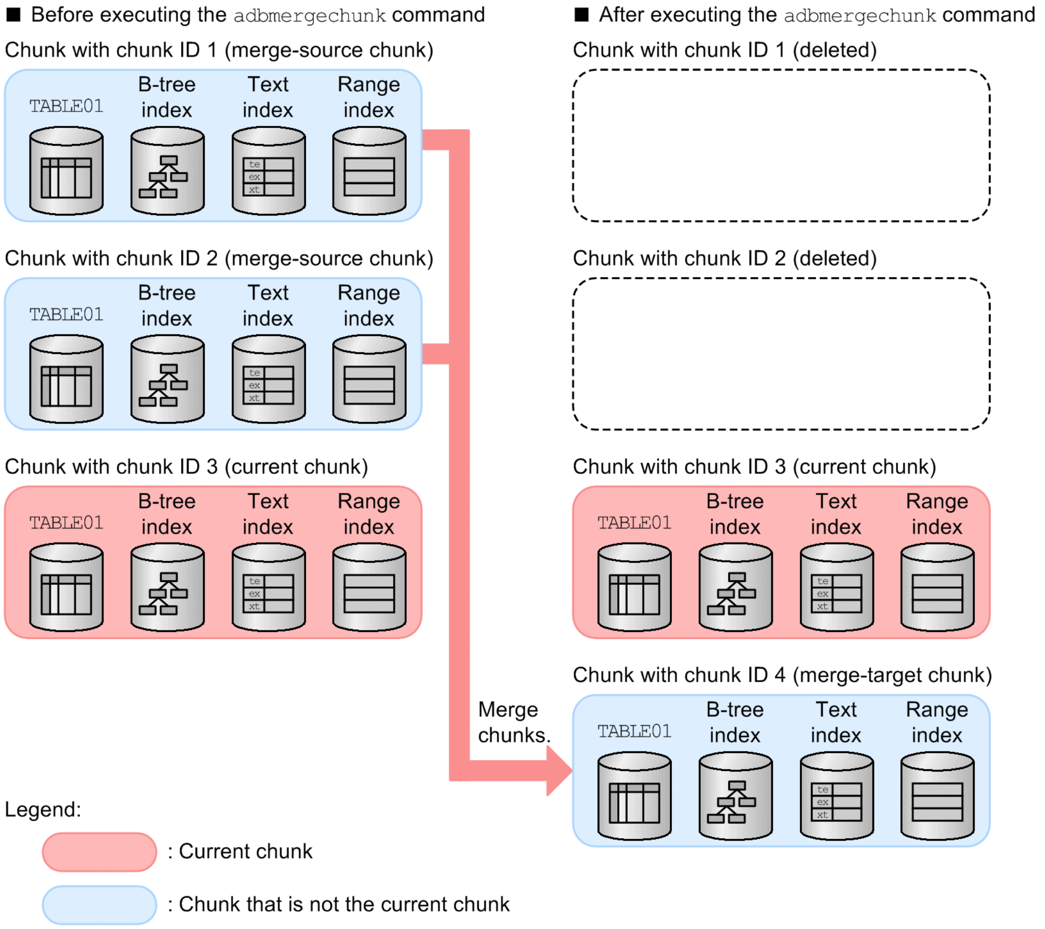
Explanation:

1. Specifies the authorization identifier of the HADB user who executes the adbmergechunk command.
2. Specifies the password for the authorization identifier that is specified in the -u option.
3. Specifies the absolute path name of the merge chunk option file created in (1) Creating a merge chunk option file.
4. Specifies a comment to be set in the merge-target chunk.
5. Specifies ALL in the -c option to merge all chunks in the target table (TABLE01).
   This example merges chunks with chunk IDs 1, 2, and 3. After the adbmergechunk command has executed, the merge-target chunk becomes the current chunk.
6. Specifies the table to be processed (TABLE01).

33.3.2 Merging any chunks (other than the current chunk)

This example merges under the following conditions any chunks other than the current chunk into a new chunk:

1. CHUNK is specified when table TABLE01 is defined (table TABLE01 is a multi-chunk table).
2. B-tree, text, and range indexes are defined for table TABLE01.
3. Data was imported into table TABLE01 by applying the background-import facility twice.
4. Table TABLE01 has chunks with chunk IDs 1, 2, and 3.
5. The chunk ID of table TABLE01's current chunk is 3.


■ Before executing the adbmergechunk command
Chunk with chunk ID 1 (merge-source chunk)
TABLE01
B-tree index
Text index
Range index
Chunk with chunk ID 2 (merge-source chunk)
TABLE01
B-tree index
Text index
Range index
Chunk with chunk ID 3 (current chunk)
TABLE01
B-tree index
Text index
Range index
■ After executing the adbmergechunk command
Chunk with chunk ID 1 (deleted)
Chunk with chunk ID 2 (deleted)
Chunk with chunk ID 3 (current chunk)
TABLE01
B-tree index
Text index
Range index
Chunk with chunk ID 4 (merge-target chunk)
TABLE01
B-tree index
Text index
Range index
Merge chunks.
Legend:
: Current chunk
: Chunk that is not the current chunk

This example uses the following procedure to execute the adbmergechunk command.

Procedure

1. Create a merge chunk option file.
2. Execute the adbmergechunk command.

Using an example, the following subsections explain each of these steps.

(1) Creating a merge chunk option file

This example creates a merge chunk option file and specifies merge chunk options in that file.

Example of a merge chunk option file

```
set adb_mergechunk_rthd_num = 3
set adb_mergechunk_scan_buff_size = 2
      :
```

(2) Executing the adbmergechunk command

```
adbmergechunk -u ADBUSER01                                   ...1
              -p '#HelloHADB_01'                             ...2
              -z /home/adbmanager/merge_file/mergechunk_opt.txt  ...3
              -m 'January 2014'                              ...4
```

```
               -c 1,2                                                 ...5
               TABLE01                                                ...6
```

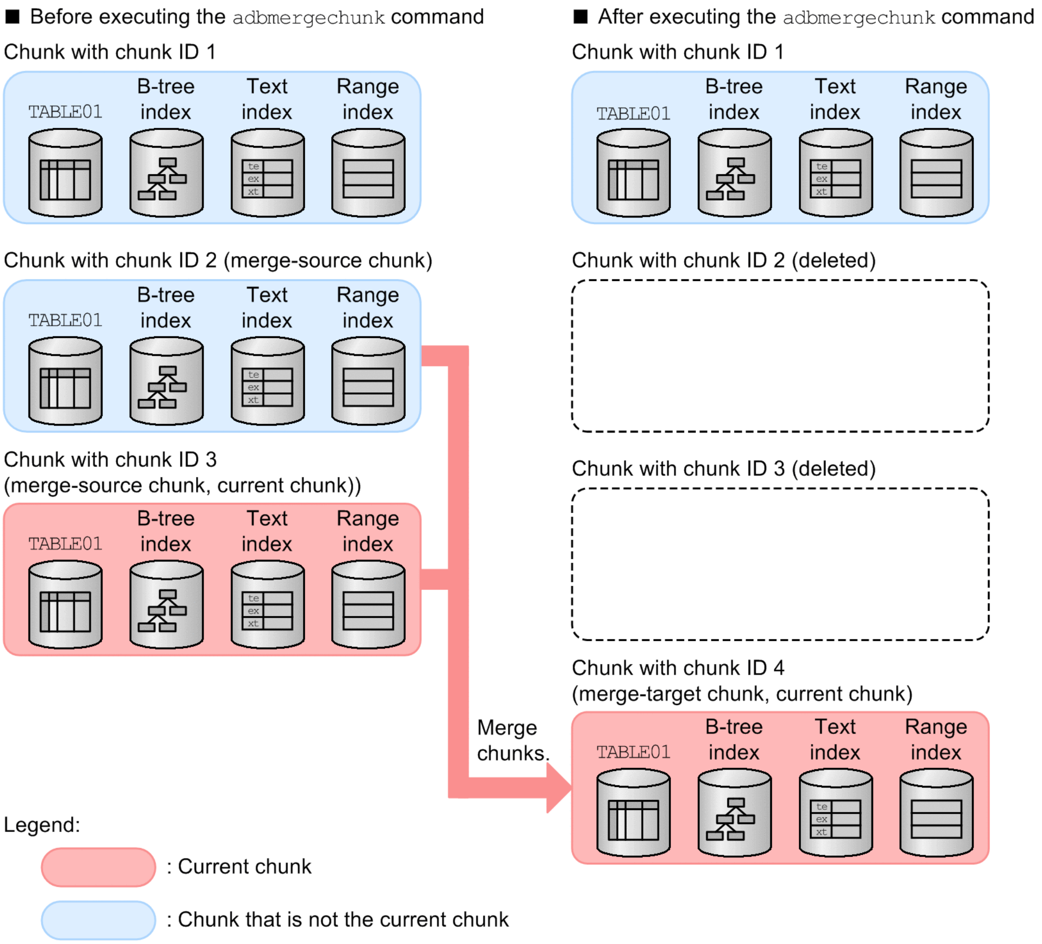
Explanation:

1. Specifies the authorization identifier of the HADB user who executes the `adbmergechunk` command.
2. Specifies the password for the authorization identifier that is specified in the `-u` option.
3. Specifies the absolute path name of the merge chunk option file created in (1) Creating a merge chunk option file.
4. Specifies a comment to be set in the merge-target chunk.
5. Specifies the merge-source chunk IDs.
   This example merges chunks with chunk IDs 1 and 2. After the `adbmergechunk` command has executed, the chunk with chunk ID 3 remains as the current chunk.
6. Specifies the table to be processed (`TABLE01`).

33.3.3 Merging any chunks (including the current chunk)

This example merges under the following conditions any chunks including the current chunk into a new chunk:

1. `CHUNK` is specified when table `TABLE01` is defined (table `TABLE01` is a multi-chunk table).
2. B-tree, text, and range indexes are defined for table `TABLE01`.
3. Data was imported into table `TABLE01` by applying the background-import facility twice.
4. Table `TABLE01` has chunks with chunk IDs 1, 2, and 3.
5. The chunk ID of table `TABLE01`'s current chunk is 3.


■ Before executing the adbmergechunk command
Chunk with chunk ID 1
TABLE01
B-tree index
Text index
Range index
Chunk with chunk ID 2 (merge-source chunk)
Chunk with chunk ID 3 (merge-source chunk, current chunk))
Merge chunks.
■ After executing the adbmergechunk command
Chunk with chunk ID 1
Chunk with chunk ID 2 (deleted)
Chunk with chunk ID 3 (deleted)
Chunk with chunk ID 4 (merge-target chunk, current chunk)
te ex xt
Legend:
: Current chunk
: Chunk that is not the current chunk

This example uses the following procedure to execute the adbmergechunk command.

Procedure

1. Create a merge chunk option file.
2. Execute the adbmergechunk command.

Using an example, the following subsections explain each of these steps.

(1) Creating a merge chunk option file

This example creates a merge chunk option file and specifies merge chunk options in that file.

Example of a merge chunk option file

```
set adb_mergechunk_rthd_num = 3
set adb_mergechunk_scan_buff_size = 2
      :
```

(2) Executing the adbmergechunk command

```
adbmergechunk -u ADBUSER01                                    ...1
              -p '#HelloHADB_01'                              ...2
              -z /home/adbmanager/merge_file/mergechunk_opt.txt ...3
              -m 'January 2014'                               ...4
```

```
               -c 2,3                                                   ...5
               TABLE01                                                  ...6
```

Explanation:

1. Specifies the authorization identifier of the HADB user who executes the `adbmergechunk` command.
2. Specifies the password for the authorization identifier that is specified in the `-u` option.
3. Specifies the absolute path name of the merge chunk option file created in (1) Creating a merge chunk option file.
4. Specifies a comment to be set in the merge-target chunk.
5. Specifies the merge-source chunk IDs.

   This example merges chunks with chunk IDs 2 and 3. After the `adbmergechunk` command has executed, the merge-target chunk becomes the current chunk.
6. Specifies the table to be processed (`TABLE01`).

33.4 Return codes

The following table lists and describes the return codes of the adbmergechunk command.

Table 33-4: Return codes of the adbmergechunk command

Return code	Description
0	The processing terminated normally.
4	The processing terminated normally, but a warning message was issued. Check the displayed message or the message output to a message log file and take appropriate action.
8	One of the following errors occurred: • The specified adbmergechunk command did not execute because it was invalid. • An error occurred during execution of the adbmergechunk command. For the cause of the error and corrective action to take, see the displayed error message or the message that has been output to a message log file. For details, see 33.5 Handling abnormal termination of the adbmergechunk command.
20	The processing terminated normally, but merge-source chunks could not be deleted. Therefore the merge-source chunks are in the *deletion-pending status*. Use the PURGE CHUNK statement to delete all the deletion-pending chunks. To check the deletion-pending chunks, see the displayed message or the KFAA96785-E message.

33.5 Handling abnormal termination of the adbmergechunk command

This section explains how to handle abnormal termination of the `adbmergechunk` command.

33.5.1 First thing to check

Check if the following messages have been issued:

- `KFAA50225-E`
- `KFAA96606-E`

If either of these messages has been output, there are errors in specified command options or merge chunk options. Correct the errors in the command options or merge chunk options, and then re-execute the `adbmergechunk` command.

For details about invalid `-c` option specifications, see (3) Errors that can be caused by the -c option in 33.2.1 Explanation of the specification format and options.

If neither of these messages was issued, go to 33.5.2 Identifying the cause of the error and taking corrective action.

33.5.2 Identifying the cause of the error and taking corrective action

The chunk status and corrective action to take depend on whether the `KFAA80242-I` message (`Purge-chunk processing start`) has been issued.

The following table describes the chunk status and corrective action to take depending on when the `adbmergechunk` command was canceled.

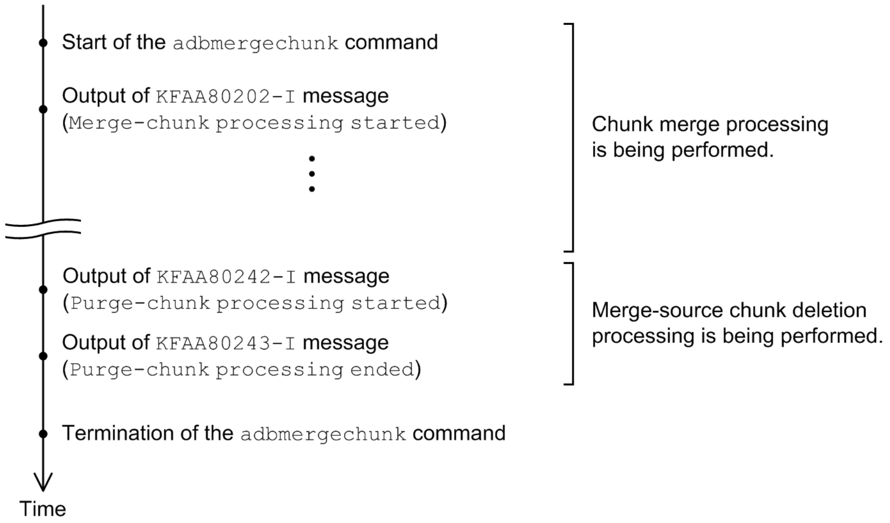
Table 33-5: Chunk status and corrective action to take depending on when the adbmergechunk command was canceled

When canceled#1	Chunk status	Corrective action to take
While merging chunks	**Status of merge-source chunks** Same as before the `adbmergechunk` command was executed **Status of merge-target chunk** Nonexistent (same as before the `adbmergechunk` command was executed)	See (1) If the adbmergechunk command was canceled while the command was merging chunks.
While deleting merge-source chunks#2	**Status of merge-source chunks** Delete-pending status. • If the ID of a delete-pending chunk is specified in the `adbexport` command or for `#GETDATA` in the `adbsql` subcommand, the count of output results will be 0. If the ID of a deletion-pending chunk is specified for `#GETCOUNT` in the `adbsql` subcommand, the number of data items to be collected is zero. Therefore, the count of output results will be 0. • The `adbmergechunk` command cannot be executed on a table that has a delete-pending chunk.	See (2) If the adbmergechunk command was canceled while the command was deleting merge-source chunks.

When canceled[#1]	Chunk status	Corrective action to take
	• A background import can be performed on a table that has a delete-pending chunk. Note, however, that a background import cannot be performed when the number of chunks has reached the maximum value. **Status of merge-target chunk** Merge processing has been completed.	

#1

You can determine when the command was canceled by the messages that are issued, as shown in the following:



Start of the adbmergechunk command
Output of KFAA80202-I message
(Merge-chunk processing started)
Chunk merge processing
is being performed.
Output of KFAA80242-I message
(Purge-chunk processing started)
Output of KFAA80243-I message
(Purge-chunk processing ended)
Merge-source chunk deletion
processing is being performed.
Termination of the adbmergechunk command
Time

#2

If the `KFAA80243-I` message (`Purge-chunk processing ended`) with return code `0` or `4` has been issued, merge chunk processing has been completed. In this case, no action is needed.

(1) If the adbmergechunk command was canceled while the command was merging chunks

This subsection explains the action to take if the `KFAA80242-I` message (`Purge-chunk processing start`) has not been issued.

Procedure

1. Check the messages that have been issued and identify the cause of the error.
2. Take appropriate action to eliminate the cause of the error.
3. Re-execute the `adbmergechunk` command.

(2) If the adbmergechunk command was canceled while the command was deleting merge-source chunks

If the `KFAA80242-I` message (`Purge-chunk processing start`) has been issued, you must delete the delete-pending chunk when the command was canceled. The following describes the corrective action to take.

Procedure

1. Check the messages that have been issued and identify the cause of the error.

2. Take appropriate action to eliminate the cause of the error.
3. As needed, check the delete-pending chunks.

   This operation is not required. Perform this operation only when you want to check the delete-pending chunks. For details about how to check the delete-pending chunks, see the following Note.
4. Use the `PURGE CHUNK` statement to simultaneously delete all of delete-pending chunks.

PURGE CHUNK statement specification example

```
PURGE CHUNK "schema-name"."table-identifier"
        WHERE CHUNKID=ANY(
                          SELECT "CHUNK_ID"
                            FROM "MASTER"."STATUS_CHUNKS"
                              WHERE "TABLE_SCHEMA" = 'schema-name'
                                AND "TABLE_NAME" = 'table-identifier'
                                AND "CHUNK_STATUS" = 'Pending Delete')
```

For the schema name, specify the schema name of the owner of the table on which the `adbmergechunk` command was executed. For the table identifier, specify the table identifier of the table.

Note

You can check the deletion-pending chunks on the usage information for DB areas, tables, and indexes, which is obtained by using the `adbdbstatus` command.

Example of the adbdbstatus command

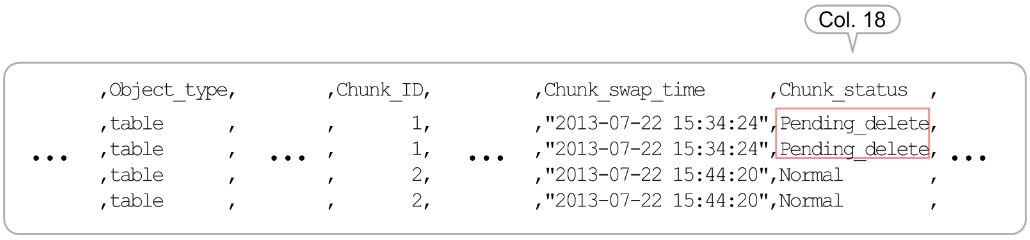
```
adbdbstatus -d used -c table -n name-of-table-for-which-adbmergechunk-command-was-canceled
```

Specify *name-of-table-for-which-adbmergechunk-command-was-canceled*, in the format *schema-name*.*table-identifier*.

Execute the command, and then check lines beginning with line 2 in the output results (usage information for DB areas, tables, and indexes). (The column titles occupy line 1 in the output results.)

- If there is a row in which the value of the `Chunk_status` column (the 18th column) is `Pending_delete`, the merge-source chunks have not been deleted. Delete-pending chunks remain.

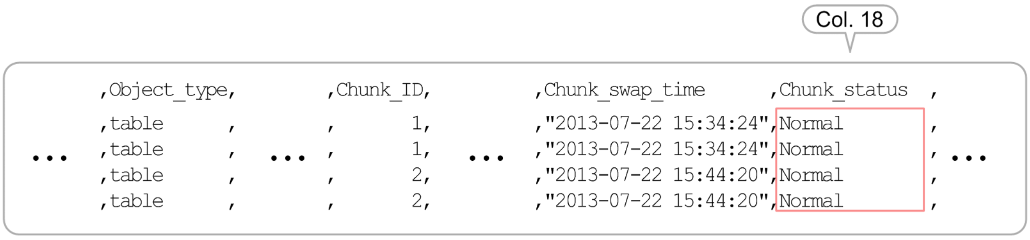
  Example:



```
      ,Object_type,       ,Chunk_ID,       ,Chunk_swap_time      ,Chunk_status  ,
      ,table      ,       ,       1,       ,"2013-07-22 15:34:24",Pending_delete,
...   ,table      ,  ...  ,       1,  ...  ,"2013-07-22 15:34:24",Pending_delete,  ...
      ,table      ,       ,       2,       ,"2013-07-22 15:44:20",Normal        ,
      ,table      ,       ,       2,       ,"2013-07-22 15:44:20",Normal        ,
```

- If `Pending_delete` is not output for the `Chunk_status` column (the 18th column), merge chunk processing has been completed. (The merge-source chunks have been deleted.)

  Example:



Col. 18

```
      ,Object_type,         ,Chunk_ID,          ,Chunk_swap_time      ,Chunk_status  ,
      ,table      ,         ,       1,          ,"2013-07-22 15:34:24",Normal        ,
 ...  ,table      ,   ...   ,       1,    ...   ,"2013-07-22 15:34:24",Normal        , ...
      ,table      ,         ,       2,          ,"2013-07-22 15:44:20",Normal        ,
      ,table      ,         ,       2,          ,"2013-07-22 15:44:20",Normal        ,
```

33.6 Notes

1. The adbmergechunk command uses SQL statements to retrieve table data that has been stored. For this reason, we recommend that you execute the command in an environment where out-of-order execution is applied. For details about environments in which out-of-order execution is applied, see *Database retrieval method (out-of-order execution)* in the *HADB Setup and Operation Guide*.
2. Use X'0A' (LF) or X'0D0A' (CRLF) for the line break code in any file specified in the options of the adbmergechunk command.
3. Use the adbcancel command to cancel execution of the adbmergechunk command. Alternatively, use a method such as the adbstop --force command or the OS's kill command to forcibly terminate the server process. After you have canceled the adbmergechunk command, see 33.5 Handling abnormal termination of the adbmergechunk command and take appropriate action as needed.

   For details about how to use the adbcancel command to suspend an executing command, see 1.6 When suspending (forcibly terminating) an executing command.
4. Merge chunk processing is performed by server processes. Therefore, using the **Ctrl**+**C** keys to forcibly terminate the adbmergechunk command process will not interrupt merge chunk processing that is being performed by server processes (such processing continues). When the merge chunk processing is completed, the command outputs the KFAA81001-I message to the message log file.
5. Work files (temporary work files) are created during merge chunk processing. These temporary work files are deleted automatically when the adbmergechunk command terminates normally (the return code is 0 or 4). If the adbmergechunk command does not terminate normally, these work files might not be deleted.

   Some of the temporary work files are needed when other commands are re-executed. For this reason, do not delete temporary work files before you re-execute other commands. If you delete them, the other commands will not execute correctly.
6. Because temporary work files might cause a shortage of disk space, we recommend that you delete temporary work files from time to time if there is no need to re-execute other commands.

   Check the following directories periodically for any remaining temporary work files and delete any that are found:

   - $DBDIR/ADBSYS/ADBUTL
   - Temporary work file storage directories specified in the -w option (or in $DBDIR/ADBWORK when the -w option was omitted)
7. You cannot execute the adbmergechunk command on the following tables:

   - A table for which any indexes that are in unfinished status are defined
   - A non-updatable table
   - A table for which execution of the adbimport or adbidxrebuild command has been canceled
8. Make sure that there is no duplication of the absolute paths for the directories and files specified in the adbmergechunk command's options. This applies to the following directories and files:

   - Temporary work file storage directories (specified in the -w option or specified in the directory path file specified in the -w option)
   - Directory path files specifying the storage locations of temporary work files (specified in the -w option)
   - Merge chunk option file (specified in the -z option)

   Make sure that the absolute paths of these directories and files are unique from those of any file names used by any programs other than the adbmergechunk command. If a duplicate file name is used, data in the file might be corrupted. Also make sure that symbolic links and relative paths are specified correctly.

9. If the `adbmergechunk` command is executed while another command or application program is running, HADB might not be able to allocate the required number of processing real threads. If this happens, HADB waits until the required number of processing real threads can be allocated before executing the `adbmergechunk` command. If allocation of processing real threads is placed in wait status, see the topic *Points to consider about the number of processing real threads to be used during command execution* in *Points to consider when executing commands concurrently* in *Preparing Resources* in the *HADB Setup and Operation Guide* and take appropriate action.
10. To reduce this command's execution time, see the topic *Reducing the execution time of the adbmergechunk command* in *Tuning to shorten command execution time* in *Tuning* in the *HADB Setup and Operation Guide*.

34

adbmodarea (Add and Change DB Areas)

This chapter explains the functionality provided by the `adbmodarea` command and how to use the command.

Note that the term *tables* in this chapter means base tables.

34.1 Description

The adbmodarea command is used to add, delete, and expand data DB areas.

Note that the adbmodarea command cannot be used to process the work table DB area, master directory DB area, dictionary DB area, or system-table DB area.

When executed, the adbmodarea command establishes a connection to the HADB server.

Important

- The adbmodarea command cannot be executed concurrently with any other application program or command.
- If using SCSI reservation for shared disk for the multi-node function, check the notes in 34.5.2 Notes about using the multi-node function before executing the adbmodarea command.
- If using SCSI reservation for shared disk for a cold standby configuration, check the notes in 34.5.1 Common notes before executing the adbmodarea command.

Note

For details about how to add, delete, or expand data DB areas, see *Handling data DB areas* in the *HADB Setup and Operation Guide*.

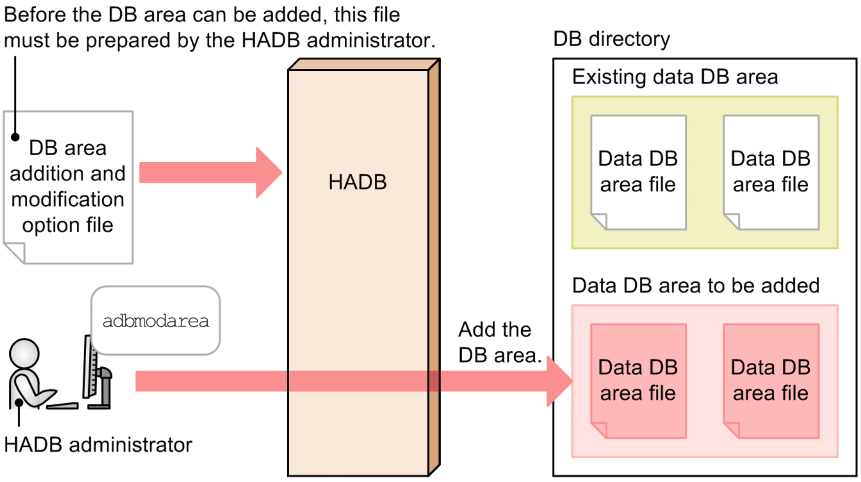
If you are using the multi-node function, see *Adding, deleting, or expanding data DB areas (when the multi-node function is being used)* in the *HADB Setup and Operation Guide*.

34.1.1 Adding a data DB area

The adbmodarea command enables you to add one data DB area to the DB directory.

The following figure provides an overview of adding a data DB area.

Figure 34-1: Adding data DB areas

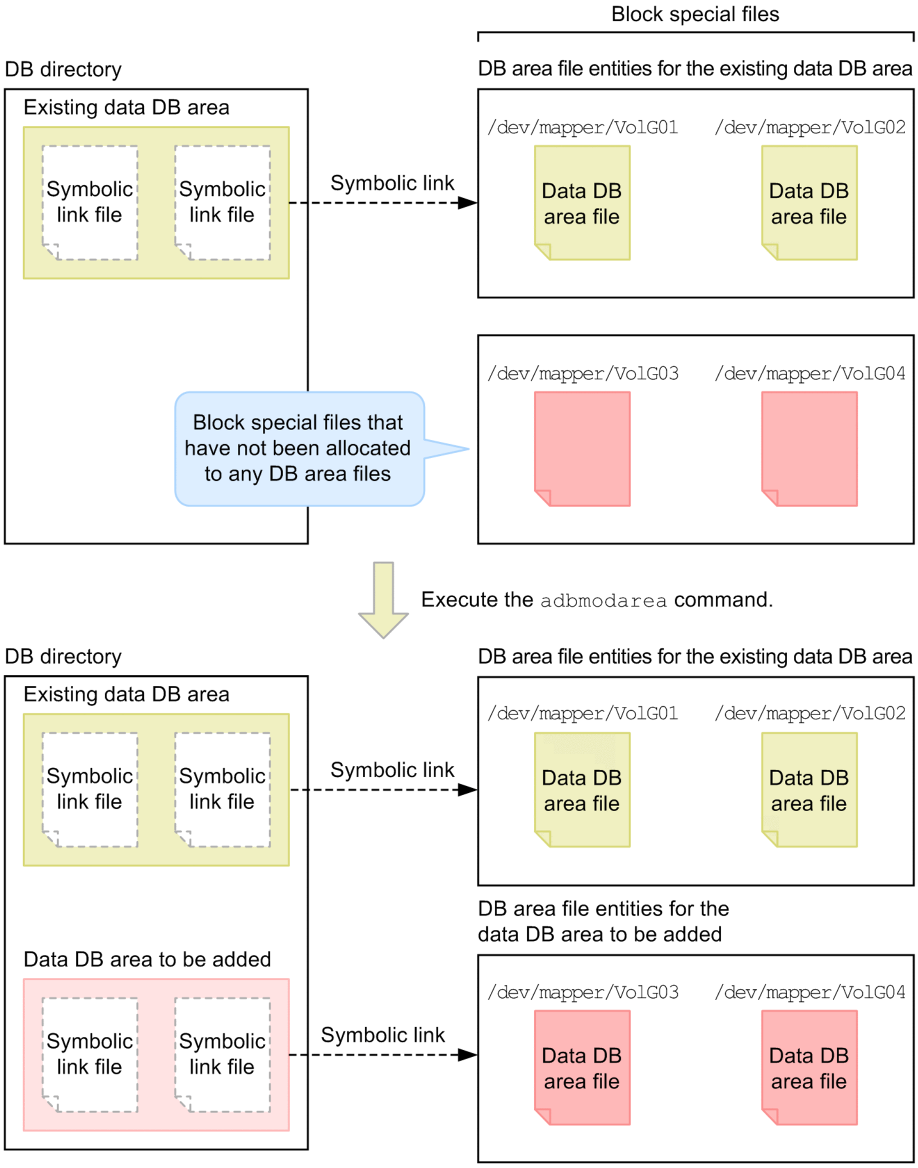


The following explains the file used to add a data DB area:

- **DB area addition and modification option file**
  This file is used to specify the options that configure an execution environment for the `adbmodarea` command. Specify information about the data DB area that is to be added.

You can allocate block special files as the DB area files for the DB area to be added. In this case, symbolic links to the block special files are created under the DB directory.

Figure 34-2: Allocating block special files as DB area files (adding a data DB area)



Important

A block special file that has already been allocated to an existing DB area file cannot be allocated as a new DB area file.

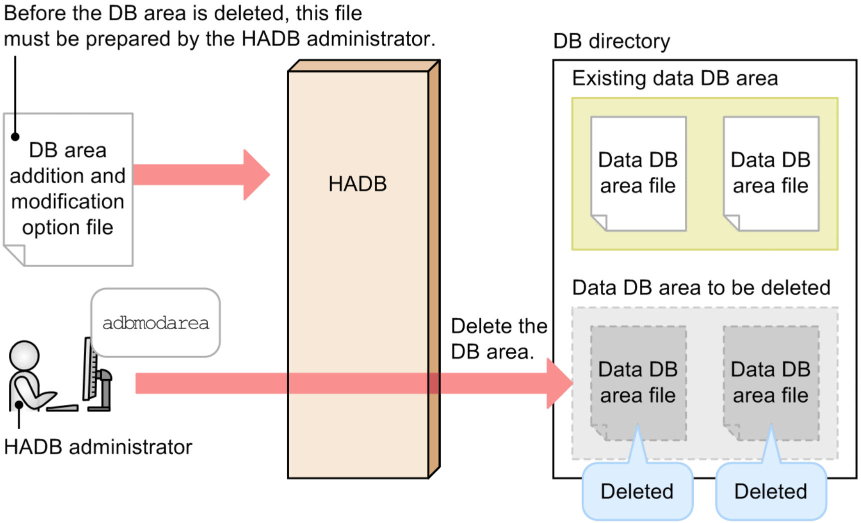
For details about the specification of the DB area addition and modification options that are specified to allocate block special files, see the adbaddarea operand in (b) Command format in (2) Explanation of specification format in 34.2.2 Format of DB area addition and modification options.

34.1.2 Deleting a data DB area

The `adbmodarea` command enables you to delete one data DB area from the DB directory.

The following figure provides an overview of deleting a data DB area.

Figure 34-3: Deleting a data DB area

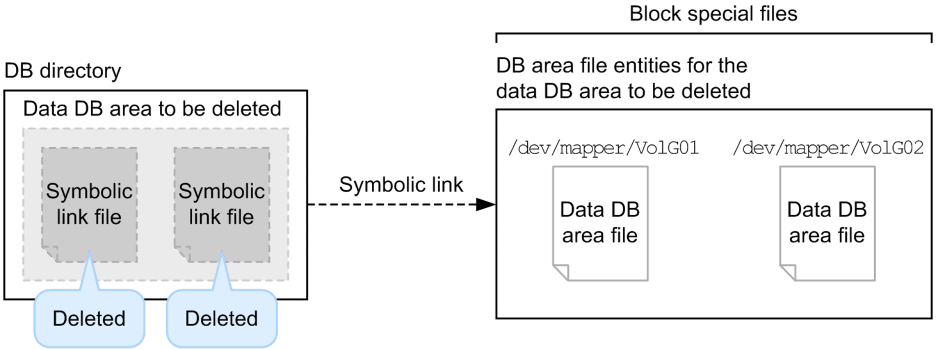


The following explains the file used to delete a data DB area:

- DB area addition and modification option file
  This file is used to specify the options that configure an execution environment for the `adbmodarea` command. Specify information about the data DB area that is to be deleted.

If the DB area files for the DB area to be deleted are block special files, symbolic link files under the DB directory are deleted.

Figure 34-4: Deleting data DB area files that are block special files

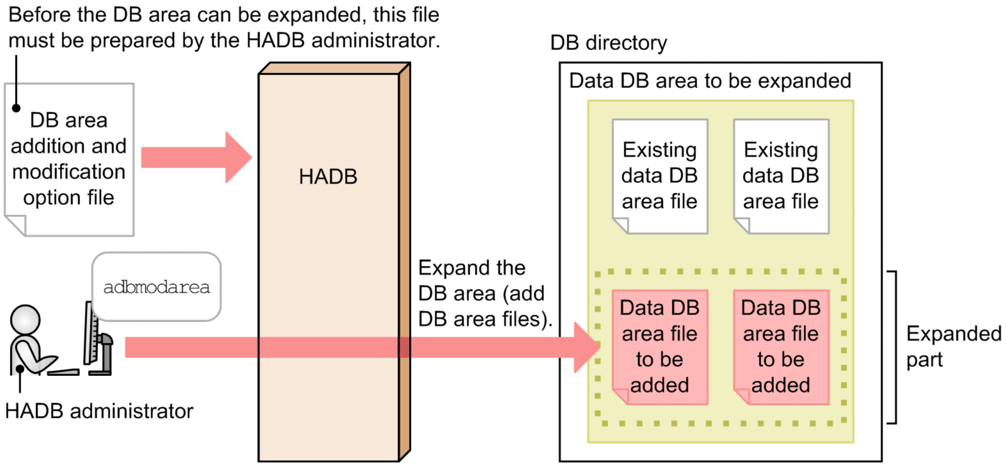


34.1.3 Expanding a data DB area (adding data DB area files)

The `adbmodarea` command enables you to expand one data DB area (add DB area files to a data DB area).

The following figure provides an overview of expanding a data DB area.

Figure 34-5: Expanding a data DB area (adding data DB area files)

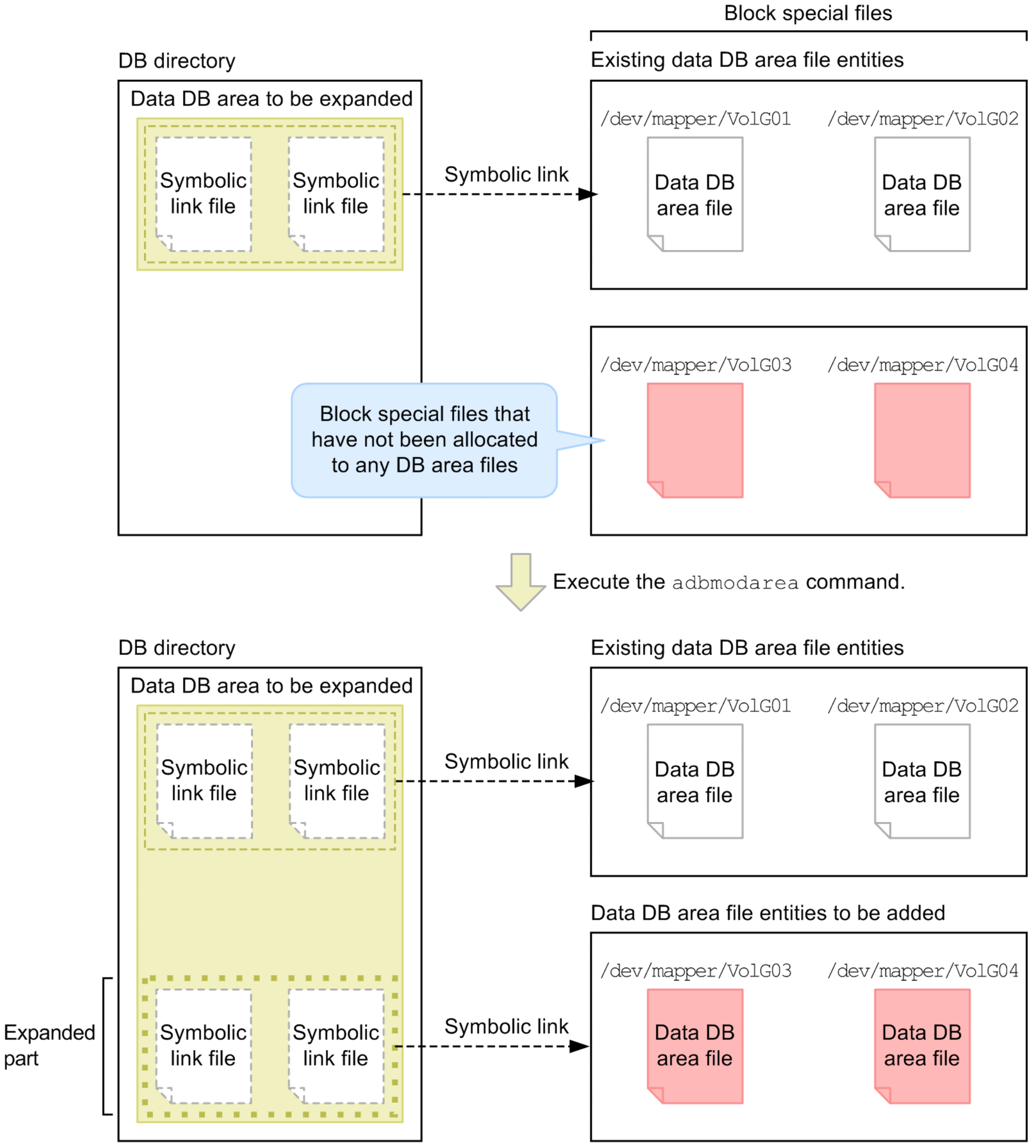

Before the DB area can be expanded, this file must be prepared by the HADB administrator.
DB area addition and modification option file
HADB
adbmodarea
HADB administrator
Expand the DB area (add DB area files).
DB directory
Data DB area to be expanded
Existing data DB area file
Existing data DB area file
Data DB area file to be added
Data DB area file to be added
Expanded part

The following explains the file used to expand a data DB area (add data DB area files):

- **DB area addition and modification option file**
  This file is used to specify the options that configure an execution environment for the `adbmodarea` command. Specify information about the data DB area that is to be expanded.

You can allocate block special files as the DB area files to be added. In this case, symbolic links to the block special files are created under the DB directory.

Figure 34-6: Allocating block special files as DB area files (expanding data DB areas)



Important

A block special file that has already been allocated to an existing DB area file cannot be allocated as a new DB area file.

For details about the DB area addition and modification options that are specified to allocate block special files, see the adbexpandarea operand in (b) Command format in (2) Explanation of specification format in 34.2.2 Format of DB area addition and modification options.

34.2 Specification format for the adbmodarea command

This section explains the specification format for the `adbmodarea` command and how to specify the DB area addition and modification options.

The only OS users who can execute the `adbmodarea` command are HADB administrators.

34.2.1 Explanation of the specification format and options

(1) Specification format

```
adbmodarea DB-area-addition-and-modification-option-file-name
```

(2) Explanation of options

- *DB-area-addition-and-modification-option-file-name*

  ~<OS path name>((2 to 510 bytes))

  Specifies the absolute path name of a file containing DB area addition and modification options. For details about the DB area addition and modification options see 34.2.2 Format of DB area addition and modification options.

  Note that if the specified path name includes a symbolic link, the length of the path might change when the symbolic link is resolved. If the length of the path name after the symbolic link is resolved is not within the range from 2 to 510 bytes, an error occurs (`KFAA96221-E` message is output).

34.2.2 Format of DB area addition and modification options

(1) Specification format

Specify one of the following operands to execute the desired operation:

- `adbaddarea` operand: To add a DB area
- `adbrmarea` operand: To delete a DB area
- `adbexpandarea` operand: To expand a DB area

```
To add a DB area
[set adb_mod_area_file_rthd_num = number-of-threads-used-to-create-DB-area-files]

  adbaddarea
    -n data-DB-area-name
  [-p {4|8|16|32}]
  [-v block-special-file-name[,block-special-file-name]...]
  [-q]
  [-i initial-allocation-size[,number-of-files-to-be-created]]
  [-f data-DB-area-file-storage-directory-name]
  [-g output-interval-for-progress-messages-for-DB-area-file-creation-processing]

To delete a DB area
  adbrmarea
    -n data-DB-area-name
```

```
To expand a DB area (adding DB area files)
[set adb_mod_area_file_rthd_num = number-of-threads-used-to-create-DB-area-files]

  adbexpandarea
    -n data-DB-area-name
  [-v block-special-file-name[,block-special-file-name]...]
  [-q]
  [-g output-interval-for-progress-messages-for-DB-area-file-creation-processing]
  [-i initial-allocation-size[,number-of-files-to-be-created]]
```

The specification rules for these DB area addition and modification options are the same as for the server definition. For details about the specification rules for the server definition, see *Syntax rules for the server definition* in the *HADB Setup and Operation Guide*.

(2) Explanation of specification format

(a) Set format

- `adb_mod_area_file_rthd_num` = *number-of-threads-used-to-create-DB-area-files*

  ~<integer>((1 to 1,024))

  Specifies the upper limit for the number of processing real threads to be used to create data DB area files concurrently.

  If this option is omitted, the number of processing real threads is determined automatically. Normally, you omit this option.

  If the following condition is satisfied, you might be able to reduce the overhead for CPU switchover by specifying this option to limit the number of processing real threads:

  - The number of processing real threads that are actually used exceeds the number of CPU cores in the server machine.

  If you specify `1` in this option or if no more than one real thread can be used, data DB area files are created one at a time.

  Note

  You can determine the actual number of processing real threads used by checking the `KFAA96207-I` message output to a message log file. The actual number of real threads that are used is equal to or less than the number of DB area files that are created and the maximum number of processing real threads that can be used by the HADB server.

  The maximum number of processing real threads that can be used by the HADB server varies depending on whether the client-group facility is applied.

  - If the client-group facility is not applied

    Value of the `adb_sys_rthd_num` operand in the server definition

  - If the client-group facility is applied

    Determine the maximum number of processing real threads that can be used by the HADB server based on the explanation in *How to determine the numbers of processing real threads when the client-group facility is used* in *Points to consider when specifying the number of processing real threads for a group* in *Points to consider when using the client-group facility* in *Preparing Resources* in the *HADB Setup and Operation Guide*.

(b) Command format

Specify the correct operand for the operation you will be performing.

- To add a data DB area
  Specify the `adbaddarea` operand.
- To delete a data DB area
  Specify the `adbrmarea` operand.
- To expand a data DB area (add data DB area files)
  Specify the `adbexpandarea` operand.

- `adbaddarea`

  Specifies the definition of a data DB area to be added.

  The options to be specified depend on the type of files to be allocated as DB area files for the data DB area that is to be added. The following table provides guidelines for specifying options.

Table 34-1: Type of files to be allocated as data DB area files and guidelines for specifying options

No.	Option name	Type of files to be allocated as DB area files		Description of option
		Block special files	Regular files	
1	`-n`	R	R	Specifies the name for the data DB area.
2	`-p`	R	R	Specifies the page size for the data DB area.
3	`-v`	R	N	Specifies the names of the block special files that are to be allocated as data DB area files.
4	`-q`	R	N	Specifies that the data DB area files are to be created in quick mode.
5	`-i`	R	R	Specifies the initial allocation size for each data DB area file and the number of data DB area files to be created.
6	`-f`	O	O	Specifies that symbolic links to the data DB area files or block special files are to be stored in a subdirectory of the DB directory.
7	`-g`	O	O	Specifies the interval for output of progress messages for data DB area file creation processing.

Legend:

R: Required/Recommended. Specification of this option is required or is recommended.

O: Optional. Specification of this option is optional. Read its description and decide whether it needs to be specified.

N: This option is not specified.

`-n` *data-DB-area-name*

~<character string>((1 to 30 bytes))

Specifies a name for the data DB area that is to be added. The name specified in this option is used as the name of the data DB area files.

The following notes apply:

- The characters permitted for a DB area name are single-byte alphanumeric characters, underscore (_), and hyphen (-). A DB area name must begin with an alphabetic character.

- If the DB area name to be used is the same as a reserved word for SQL statements, enclose the DB area name in backslashes and double quotation marks (`\"`). For details about the reserved words for SQL statements, see *Reserved words* in *SQL Basics* in the manual *HADB SQL Reference*.
- If the specified DB area name contains a lowercase letter, enclose the entire DB area name in backslashes and double quotation marks (`\"`). If the specified name is not enclosed in backslashes and double quotation marks, the specified character string will be treated as being in all uppercase letters.

  Example: `\"DBarea01\"`
- This name must be unique from any existing DB area name. To check the existing DB area names, ask the HADB administrator to execute the command shown below to output summary information for DB areas. Check the DB area names (`DBarea_name` column) displayed in the summary information. For details about the items that are output as summary information for DB areas, see 13.3.2 List of items that are output as summary information for DB areas.

  ```
  adbdbstatus
  ```
- This name must be unique from any directory name under the DB directory.

`-p {4|8|16|32}`

Specifies the page size (in kilobytes) for the data DB area that is to be added. Specify the page size you estimated when you designed the data DB area. For details about how to estimate the page size for the data DB area, see *Points to consider when determining the page size in data DB areas* in the *HADB Setup and Operation Guide*.

`-v` *block-special-file-name* `[,` *block-special-file-name* `]...`

~<path name>((2 to 255 bytes))

When (and only when) you allocate block special files as the data DB area files, specifies the absolute path names of the block special files to be used, expressed as file entities or symbolic links. These block special files must not already be allocated to DB area files.

The following notes apply:

- Specify block special files that have been created by LVM or device names that begin with `/dev/disk/` according to HADB's path naming rules.
- Make sure that the same block special file name is not specified more than once.
- A block special file that has already been allocated to a DB area file cannot be allocated as a new DB area file.
- For guidelines for the number of block special files that can be specified in this option when an LVM is set up (number of LVs constituting one DB area), see *Points to consider when setting up an LVM* in the *HADB Setup and Operation Guide*.
- Specify as many block special files as there are files to be created, as specified in the `-i` option. If fewer block special files are specified than the number of files to be created, the `adbmodarea` command will result in an error.

When you specify this option, the command creates symbolic links to the block special files under the DB directory.

Important

An improvement in performance can be expected when block special files are allocated instead of regular files.

However, note that when block special files are allocated, any existing data in the specified block special files will be deleted. If you are not sure that is it safe to delete existing data, do not allocate those block special files.

A block special file mounted on the OS cannot be allocated. A block special file to be allocated must be unmounted.

`-q`

Specifies that the data DB area files are to be created in quick mode. Specifying this option enables you to reduce the time required to create the data DB area files.

The `-q` option is applied if the files to be initialized are block special files.

Note

When the data DB area files are created in quick mode, only each file's management page (containing information such as the file size) is created. The pages for storing actual data are not created. This is why it takes less time to create the data DB area files.

`-i` *initial-allocation-size*`[,`*number-of-files-to-be-created*`]`

Specifies the initial allocation size for each data DB area file and the number of data DB area files to be created.

- *initial-allocation-size*

  ~<alphabetic and numeric characters>((0K to 127P))<<0K>>

  Specifies the initial allocation size for each data DB area file, using one of the following letters to indicate the units: `K` for kilobytes, `M` for megabytes, `G` for gigabytes, `T` for terabytes, or `P` for petabytes. The size of a data DB area file is determined according to the specified initial allocation size. However, depending on the location of pages during initial allocation, the initial allocation size specified in this option might not be allocated for data DB area files.

  The guidelines for the initial allocation size depend on the type of the data DB area file. Therefore, do not create different types of data DB area files at the same time. The following table shows the guidelines for the initial allocation size according to the types of data DB area file.

Table 34-2: Types of data DB area file and guidelines for the initial allocation size

No.	Type of data DB area file		Guidelines for the value of initial allocation size
1	Regular file		The value of *initial-allocation-size* multiplied by the *number-of-files-to-be-created* must be less than the value of *free-space-in-file-system-in-which-to-allocate-data-DB-area-files*. You can specify any value that meets this condition. When you determine the specific value, consider the disk space consumption and the time required for executing the `adbmodarea` command.
2	Block special file	Physical volume (normal block special file to which physical blocks are statically allocated)	If all physical volumes to be allocated have the same size, specify the size of one volume for *initial-allocation-size*. We do not recommend that physical volumes of different sizes are allocated at the same time. If you need to allocate physical volumes of different sizes at the same time, specify the size of the smallest volume.
3		Virtual volume (block special file with storage virtualization technology called thin provisioning# applied)	To use a virtual volume of less than 16 terabytes, specify `0K`. If a virtual volume might use 16 or more terabytes, specify the size you are planning to use. However, we do not recommend allocation of virtual volumes if the use of 16 or more terabytes is expected. If a size of 16 or more terabytes is specified, a large number of physical blocks are required from the beginning, which lacks the advantage of using lower cost virtual volumes.

#: For details about thin provisioning, see the manuals of storage products.

Note

The size of the data DB area file might not be allocated as specified in the `-i` option. The following describes the reason.

A data DB area file consists of the following pages:

- Directory pages
- Segments (pages for storing table data and indexes)

When the size of the data DB area file is specified in the `-i` option, pages are located so that the file is fit in the specified size. However, if the size specified in the `-i` option is reached in the middle of a directory page or segment, the number of pages is adjusted as follows.

- Consecutive directory pages are always created at the beginning of the data DB area file. Therefore, if a value smaller than the minimum size of the data DB area file (approximately 5.4 megabytes) is specified, the value is rounded up.
- Pages that make up segments are managed in units of a segment (that is, in units of the number of pages that make up a segment). Therefore, the number of pages is adjusted so that the specified file size is not exceeded in the middle of a segment. For example, if the file size is exceeded in the middle of the eighth segment, the eighth segment is not created (only seven segments are created).

For details about directory pages and segments in data DB area files, see *Existence of directory page groups and locations of directory pages* in *Page types* in *Pages* in *DB area structure (segments and pages)* in the *HADB Setup and Operation Guide*.

- *number-of-files-to-be-created*
  ~<integer>((1 to 1,024))<<1>> (count)
  Specifies the number of data DB area files to be created.
  If the `-v` option is specified, make sure that *number-of-files-to-be-created* is the same as the number of block special files specified in the `-v` option.

The initial allocation size, depending on the `-i` option value, is explained in the following by way of an example.

Example:

```
adbaddarea -n NEWAREA -i 2G,3
```

In this example, three data DB area files are to be created in the data DB area named `NEWAREA`, and two gigabytes of space are to be allocated as the initial allocation size for each data DB area file. The total initial allocation size for `NEWAREA` will be six gigabytes.

Furthermore, if you attempt to allocate an initial allocation size that exceeds the maximum according to the specification of the `-i` option, the `adbmodarea` command results in an error. Change the specification of the `-i` option according to the `KFAA96228-E` message that is output.

Important

If a large block special file exceeding the maximum value of automatic extension of the DB area file is allocated, an unusable area is generated in the block special file. As a result, the warning message `KFAA96244-W` is output when the `adbmodarea` command is executed. If you want to use areas in the block special file almost to the capacity limit, take the corrective action described for the `KFAA96244-W` message.

The maximum value of automatic extension of a DB area file is determined according to the specified initial allocation size. A multiple of 16 terabytes rounded up from the specified initial allocation size will be the maximum value of automatic extension of one DB area file. For example, if the specified initial allocation size is 20 terabytes, the maximum value of automatic extension of the DB area file is 32 terabytes.

Note that an unusable area cannot be made usable by any means.

To allocate small-capacity and large-capacity block special files to a single DB area, initialize the block special files by specifying a different initial allocation size for each of these files. The following shows an example.

Example: Allocate the following two block special files to DB area `AREA1` to be added:

- `/dev/mapper/vg_10TB_1` (size: 10 terabytes)
- `/dev/mapper/vg_30TB_1` (size: 30 terabytes)

Procedure

1. Add DB area `AREA1` by using the `adbmodarea` command. At this time, initialize the `/dev/mapper/vg_10TB_1` file with the DB area addition and modification option specified as follows:
   ```
   adbaddarea -n AREA1 ... -i 10T -v /dev/mapper/vg_10TB_1
   ```
2. Expand DB area `AREA1` by using the `adbmodarea` command. At this time, add the `/dev/mapper/vg_30TB_1` file with the DB area addition and modification option specified as follows.
   ```
   adbexpandarea -n AREA1 ... -i 30T -v /dev/mapper/vg_30TB_1
   ```

When the two block special files are concurrently initialized in step 1 above, a value exceeding 10 terabytes cannot be specified for the initial allocation size. Therefore, the maximum value of automatic extension of the DB area file is 16 terabytes. In this case, the maximum available space in the 30-terabyte block special file is only 16 terabytes, which is the maximum value of automatic extension of the DB area file.

`-f` *data-DB-area-file-storage-directory-name*

~<path name>((1 to 70 bytes))

Specifies the subdirectory that stores the data DB area files or a symbolic link to the block special files, expressed as a path name relative to the DB directory. A data DB area file with the DB area name as its name or a symbolic link to a block special file is stored directly under this subdirectory.

If you omit this option, the data DB area file names or the symbolic links to the block special files are stored directly under the DB directory.

The following shows specification examples.

Example:

```
adbaddarea -n NEWAREA -f DB02/
```

This example creates a subdirectory named `DB02` directly under the DB directory and then creates a data DB area file with file name `NEWAREA` or symbolic link to the block special file under that subdirectory.

The following notes apply when you specify this option:

- There is no need to specify a forward slash (`/`) at the beginning of a relative path name.
- Specify a path name that begins with a single-byte alphabetic character, ends with a forward slash (`/`), and consists of only single-byte alphanumeric characters, the underscore (_), and the hyphen (`-`). If the path name does not end with a forward slash, a forward slash is added automatically at the end of the path name. The permitted maximum length of a path name is 69 bytes.
- If the name of the DB area file storage directory includes a lowercase letter, enclose the entire storage directory name in backslashes and double quotation marks (`\"`). If it is not enclosed in backslashes and

double quotation marks, the character string specified as the DB area file storage directory name will be treated as being in all uppercase letters.

Example: `\"Db02/\"`

- No path under a directory that is created in the DB directory can be specified.
- If the specified path contains an intermediate directory with the same name as the DB area name, directory or file creation might fail. For this reason, you must make sure that the directory in which each DB area file is created does not contain any intermediate directory with the same name as the DB area name.

`-g` *output-interval-for-progress-messages-for-DB-area-file-creation-processing*

~<integer>((0 to 100))<<0>> (%)

Specify this option when you wish to issue messages that report the progress of data DB area file creation processing.

This option specifies as a percentage the message output interval. For example, if `30` is specified, the command issues the `KFAA96232-I` message reporting the progress when completion of DB area creation processing reaches 30%, 60%, and 90%.

If you specify `0` or nothing in this option, no progress messages are output.

- `adbrmarea`

Specifies the definition of a data DB area that is to be deleted.

`-n` *data-DB-area-name*

~<character string>((1 to 30 bytes))

Specifies the name of the data DB area that is to be deleted.

If the specified DB area name is for any of the following DB areas, the DB area cannot be deleted.

- DB area that is not a data DB area
- Last data DB area
- Data DB area that contains tables or indexes

  For details about how to check for whether tables or indexes are stored in a DB area, see *Searching a dictionary table* in the *HADB Setup and Operation Guide*.

The following notes apply:

- If the name of the DB area to be deleted is the same as a reserved word for SQL statements, enclose the DB area name in backslashes and double quotation marks (`\"`). For details about the reserved words for SQL statements, see *Reserved words* in *SQL Basics* in the manual *HADB SQL Reference*.
- If the name of the DB area to be deleted contains a lowercase letter, enclose the entire DB area name in backslashes and double quotation marks (`\"`). If it is not enclosed in backslashes and double quotation marks, the character string used as the DB area name will be treated as being in all uppercase letters.

  Example: `\"DBarea01\"`

- `adbexpandarea`

Specifies the definition of a data DB area that is to be expanded (to which data DB area files are to be added).

The options to be specified depend on the type of files that make up the data DB area that is to be expanded. The following table provides guidelines for specifying options.

Table 34-3: Type of files that make up the data DB area to be expanded and guidelines for specifying options

No.	Option name	Type of files that make up the data DB area to be expanded		Description of option
		Block special file	Regular file	
1	`-n`	R	R	Specifies the name for the data DB area.
2	`-v`	R	N	Specifies the names of the block special files that are to be allocated as data DB area files.
3	`-q`	R	N	Specifies that the data DB area files are to be created in quick mode.
4	`-i`	R	R	Specifies the initial allocation size for each data DB area file and the number of data DB area files to be created.
5	`-g`	O	O	Specifies the interval for output of progress messages for data DB area file creation processing.

Legend:

R: Required/Recommended. Specification of this option is required or is recommended.

O: Optional. Specification of this option is optional. Read its description and decide whether it needs to be specified.

N: This option is not specified.

Important

Make sure that regular files are not intermixed with block special files in the DB area. If the data DB area to be expanded contains even one DB area file that is a block special file, you must specify the `-v` option. For a data DB area consisting of DB area files that are all regular files, the `-v` option cannot be specified.

`-n` *data-DB-area-name*

~<character string>((1 to 30 bytes))

Specifies the name of the data DB area that is to be expanded (to which DB area files are to be added).

The following notes apply:

- If the name of the DB area that is to be expanded is the same as a reserved word for SQL statements, enclose the DB area name in backslashes and double quotation marks (`\"`). For details about the reserved words for SQL statements, see *Reserved words* in *SQL Basics* in the manual *HADB SQL Reference*.
- If the name of the DB area to be expanded contains a lowercase letter, enclose the entire DB area name in backslashes and double quotation marks (`\"`). If it is not enclosed in backslashes and double quotation marks, the character string specified as the DB area name will be treated as being in all uppercase letters.

  Example: `\"DBarea01\"`

`-v` *block-special-file-name*`[,`*block-special-file-name*`]...`

~<path name>((2 to 255 bytes))

When (and only when) you allocate block special files as the data DB area files, specifies the absolute path names of the block special files to be used, expressed as file entities or symbolic links. These block special files must not already be allocated to DB area files.

The following notes apply:

- If the data DB area to be expanded contains even one DB area file that is a block special file, you must specify this option. For a data DB area consisting of DB area files that are all regular files, this option cannot be specified.
- Specify block special files that have been created by LVM or device names that begin with `/dev/disk/` according to HADB's path naming rules.
- Make sure that the same block special file name is not specified more than once.
- A block special file that has already been allocated to a DB area file cannot be allocated as a new DB area file.
- For guidelines for the number of block special files that can be specified in this option when an LVM is set up (number of LVs constituting one DB area), see *Points to consider when setting up an LVM* in the *HADB Setup and Operation Guide*.
- Specify as many block special file names as there are DB area files to be created, as specified in the `-i` option. If fewer block special file are specified than the number of DB area files to be created, the `adbmodarea` command will result in an error.

When you specify this option, the command creates symbolic links to the block special files under the DB directory.

Important

An improvement in performance can be expected when block special files are allocated instead of regular files.

However, note that when block special files are allocated, any existing data in the specified block special files will be deleted. If you are not sure that it is safe to delete existing data, do not allocate those block special files

A block special file mounted on the OS cannot be allocated. A block special file to be allocated must be unmounted.

`-q`

Specifies that the data DB area files are to be created in quick mode. Specifying this option enables you to reduce the time required to create the data DB area files.

The `-q` option is applied if the files to be initialized are block special files.

Note

When the data DB area files are created in quick mode, only each file's management page (containing information such as the file size) is created. The pages for storing actual data are not created. This is why it takes less time to create the data DB area files.

`-i` *initial-allocation-size*`[,`*number-of-files-to-be-created*`]`

Specifies the initial allocation size for each data DB area file and the number of data DB area files to be created.

- *initial-allocation-size*

  ~<alphabetic and numeric characters>((0K to 127P))<<0K>>

  Specifies the initial allocation size for each data DB area file, using one of the following letters to indicate the units: `K` for kilobytes, `M` for megabytes, `G` for gigabytes, `T` for terabytes, or `P` for petabytes. The size of a data DB area file is determined according to the specified initial allocation size. However, depending on the location of pages during initial allocation, the initial allocation size specified in this option might not be allocated for data DB area files.

The guidelines for the initial allocation size depend on the type of the data DB area file. Therefore, do not create different types of data DB area files at the same time. The following table shows the guidelines for the initial allocation size according to the types of data DB area file.

Table 34-4: Types of data DB area file and guidelines for the initial allocation size

No.	Type of data DB area file		Guidelines for the value of initial allocation size
1	Regular file		The value of *initial-allocation-size* multiplied by the *number-of-files-to-be-created* must be less than the value of *free-space-in-file-system-in-which-to-allocate-data-DB-area-files*. You can specify any value that meets this condition. When you determine the specific value, consider the disk space consumption and the time required for executing the `adbmodarea` command.
2	Block special file	Physical volume (normal block special file to which physical blocks are statically allocated)	If all physical volumes to be allocated have the same size, specify the size of one volume for *initial-allocation-size*. We do not recommend that physical volumes of different sizes are allocated at the same time. If you need to allocate physical volumes of different sizes at the same time, specify the size of the smallest volume.
3		Virtual volume (block special file with storage virtualization technology called thin provisioning[#] applied)	To use a virtual volume of less than 16 terabytes, specify `0K`. If a virtual volume might use 16 or more terabytes, specify the size you are planning to use. However, we do not recommend allocation of virtual volumes if the use of 16 or more terabytes is expected. If a size of 16 or more terabytes is specified, a large number of physical blocks are required from the beginning, which lacks the advantage of using lower cost virtual volumes.

#: For details about thin provisioning, see the manuals of storage products.

Note

The size of the data DB area file might not be allocated as specified in the `-i` option. The following describes the reason.

A data DB area file consists of the following pages:

- Directory pages
- Segments (pages for storing table data and indexes)

When the size of the data DB area file is specified in the `-i` option, pages are located so that the file is fit in the specified size. However, if the size specified in the `-i` option is reached in the middle of a directory page or segment, the number of pages is adjusted as follows.

- Consecutive directory pages are always created at the beginning of the data DB area file. Therefore, if a value smaller than the minimum size of the data DB area file (approximately 5.4 megabytes) is specified, the value is rounded up.
- Pages that make up segments are managed in units of a segment (that is, in units of the number of pages that make up a segment). Therefore, the number of pages is adjusted so that the specified file size is not exceeded in the middle of a segment. For example, if the file size is exceeded in the middle of the eighth segment, the eighth segment is not created (only seven segments are created).

For details about directory pages and segments in data DB area files, see *Existence of directory page groups and locations of directory pages* in *Page types* in *Pages* in *DB area structure (segments and pages)* in the *HADB Setup and Operation Guide*.

- *number-of-files-to-be-created*

  ~<integer>((1 to 1,023))<<1>> (count)

  Specifies the number of data DB area files to be created.

  If the `-v` option is specified, make sure that *number-of-files-to-be-created* is the same as the number of block special files specified in the `-v` option.

  A value exceeding the value obtained from the following formula cannot be specified:

  1,024 - *current number of DB area files in the data DB area to be expanded*

The initial allocation size, depending on the `-i` option value, is explained in the following by way of an example.

Example:

```
adbexpandarea -n OLDAREA -i 2G,3
```

In this example, three data DB area files are added to data DB area `OLDAREA`, and two gigabytes of space are allocated as the initial allocation size to each data DB area file. Therefore, a total of six gigabytes are allocated as the initial allocation size for the DB area files that will be added to `OLDAREA`.

Furthermore, if you attempt to allocate an initial allocation size that exceeds the maximum according to the specification of the `-i` option, the `adbmodarea` command results in an error. Change the specification of the `-i` option according to the `KFAA96228-E` message that is output.

> **Important**
>
> If a block special file larger than 16 terabytes is added by specifying a small value for the initial allocation size, the area exceeding the maximum value of automatic extension of the DB area file cannot be used. To allow use of this area, you need to initialize the added block special file again. Note, however, that if the DB area to be expanded contains data you do not want to delete, a more complicated procedure is required to initialize the added block special file.
>
> If the whole area of the block special file will be unavailable due to the specification, the `KFAA96246-Q` message (confirming that you want to perform expansion of the DB area) is displayed when you execute the `adbmodarea` command. If this message is displayed, we recommend that you enter `n` or `N` to cancel expansion of the DB area, and then revise the specified initial allocation size.

`-g` *output-interval-for-progress-messages-for-DB-area-file-creation-processing*

~<integer>((0 to 100))<<0>> (%)

Specify this option when you wish to issue messages that report the progress of data DB area file creation processing.

This option specifies as a percentage the message output interval. For example, if `30` is specified, the command issues the `KFAA96232-I` message reporting the progress when completion of DB area expansion processing reaches 30%, 60%, and 90%.

If you specify `0` or nothing in this option, no progress messages are output.

(3) Model DB area addition and modification option

A model DB area addition and modification option is available (`$ADBDIR/sample/conf/adbmodarea.opt`). Use this model when you create DB area addition and modification options.

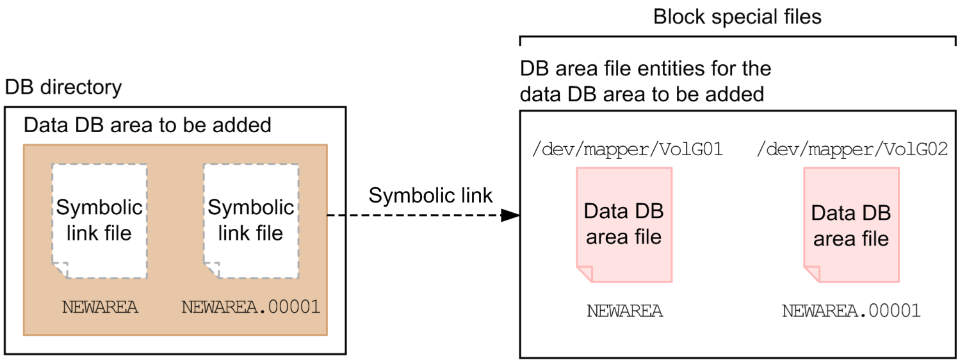
34.3 Examples

This section explains by way of examples how to use the `adbmodarea` command to add, delete, and expand data DB areas.

Only the OS user who is the HADB administrator can execute the `adbmodarea` command.

34.3.1 Adding a DB area (creating DB area files using block special files)

This example adds a data DB area to the DB directory by executing the `adbmodarea` command. It allocates block special files as the data DB area files, as shown in the following:



Explanation:

Block special files (`/dev/mapper/VolG01` and `/dev/mapper/VolG02`) are allocated as data DB area files for the added data DB area.

The following is the definition of the data DB area that is to be added:

- DB area name: `NEWAREA`
- Number of DB area files: 2
- Initial allocation size for each DB area file: 500 megabytes

This example assumes the existence of a symbolic link destination directory and block special files and that the required access permissions have been granted.

The example uses the following procedure to execute the `adbmodarea` command.

Procedure

1. Create a DB area addition and modification option file.
2. Execute the `adbmodarea` command.

Using an example, the following subsections explain each of these steps.

(1) Creating a DB area addition and modification option file

This example creates a DB area addition and modification option file and specifies the definition of a data DB area that is to be added.

Example of a DB area addition and modification option file

```
adbaddarea -n NEWAREA \                                  ...1
           -v /dev/mapper/VolG01,/dev/mapper/VolG02 \    ...2
           -q \                                          ...3
           -i 500M,2                                     ...4
```

Explanation:

1. Specifies the data DB area name (`NEWAREA`).
2. Specifies the block special files (`/dev/mapper/VolG01` and `/dev/mapper/VolG02`) that are to be allocated as the data DB area files.
3. Specifies that the data DB area files are to be created in quick mode.
4. Specifies the initial allocation size (500 megabytes) of each data DB area file and the number of files (2) to be created.

(2) Executing the adbmodarea command

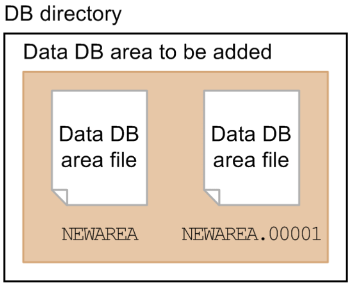
```
adbmodarea /HADB/server/conf/adbmodarea.opt
```

Explanation:

Specifies the name of the DB area addition and modification option file created in (1) Creating a DB area addition and modification option file.

34.3.2 Adding a DB area (creating DB area files using regular files)

This example adds a data DB area to the DB directory by executing the `adbmodarea` command. It creates DB area files under the DB directory, as shown in the following:



Explanation:

The following is the definition of the data DB area to be added:

- DB area name: `NEWAREA`
- Number of DB area files: 2
- Initial allocation size for each DB area file: 500 megabytes

The example uses the following procedure to execute the `adbmodarea` command.

Procedure

1. Create a DB area addition and modification option file.
2. Execute the `adbmodarea` command.

Using an example, the following subsections explain each of these steps.

(1) Creating a DB area addition and modification option file

This example creates a DB area addition and modification option file and specifies the definition of a data DB area that is to be added.

Example of a DB area addition and modification option file

```
adbaddarea -n NEWAREA \                                    ...1
           -i 500M,2                                       ...2
```

Explanation:

1. Specifies the data DB area name (NEWAREA).
2. Specifies the initial allocation size (500 megabytes) of each data DB area file and the number of files (2) to be created.

(2) Executing the adbmodarea command

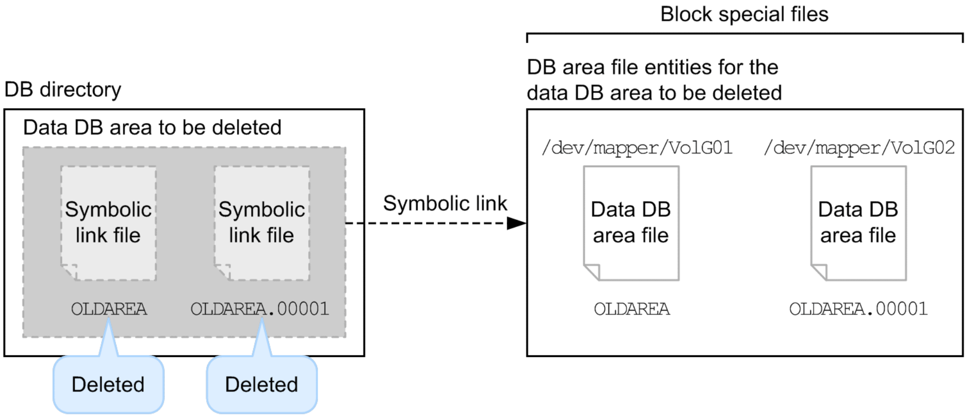
```
adbmodarea /HADB/server/conf/adbmodarea.opt
```

Explanation:

Specifies the name of the DB area addition and modification option file created in (1) Creating a DB area addition and modification option file.

34.3.3 Deleting a DB area (deleting DB area files that are block special files)

This example deletes a data DB area by executing the adbmodarea command. It deletes DB area files that are block special files, as shown in the following:



Explanation:

- The data DB area OLDAREA is deleted.
- Block special files (/dev/mapper/VolG01 and /dev/mapper/VolG02) are allocated as the data DB area files for OLDAREA.
- No tables or indexes are stored in OLDAREA.

- OLDAREA is not the only data DB area.

This example assumes that the required access permissions have been granted for the symbolic link destination directory and the block special files.

The example uses the following procedure to execute the adbmodarea command.

Procedure

1. Create a DB area addition and modification option file.
2. Execute the adbmodarea command.

Using an example, the following subsections explain each of these steps.

(1) Creating a DB area addition and modification option file

This example creates a DB area addition and modification option file and specifies the name of a data DB area (OLDAREA) that is to be deleted.

Example of a DB area addition and modification option file

```
adbrmarea -n OLDAREA
```

(2) Executing the adbmodarea command

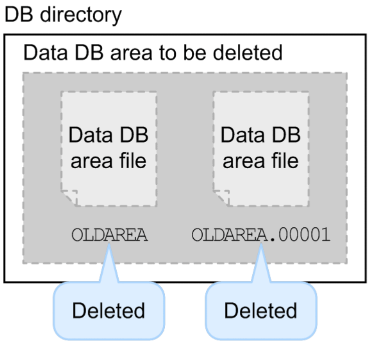
```
adbmodarea /HADB/server/conf/adbmodarea.opt
```

Explanation:

Specifies the name of the DB area addition and modification option file created in (1) Creating a DB area addition and modification option file.

34.3.4 Deleting a DB area (deleting DB area files that are regular files)

This example deletes a data DB area by executing the adbmodarea command. It deletes DB area files under the DB directory.



Explanation:

- The data DB area OLDAREA is deleted.
- No tables or indexes are stored in OLDAREA.

- OLDAREA is not the only data DB area.

The example uses the following procedure to execute the adbmodarea command.

Procedure

1. Create DB area addition and modification options.
2. Execute the adbmodarea command.

Using an example, the following subsections explain each of these steps.

(1) Creating a DB area addition and modification option file

This example creates a DB area addition and modification option file and specifies the name of a data DB area (OLDAREA) that is to be deleted.

Example of a DB area addition and modification option file

```
adbrmarea -n OLDAREA
```

(2) Executing the adbmodarea command

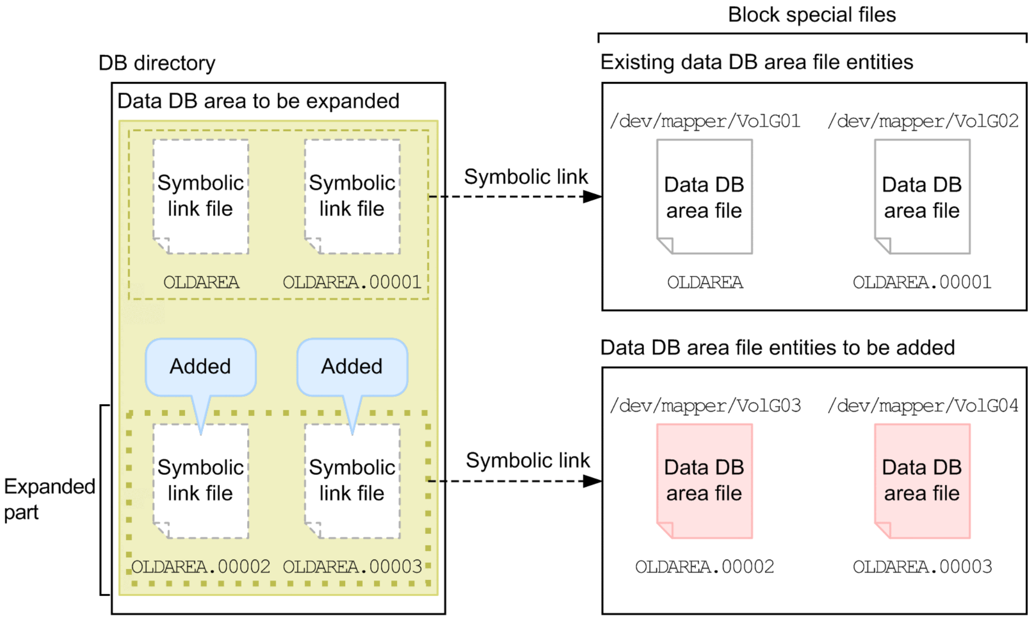
```
adbmodarea /HADB/server/conf/adbmodarea.opt
```

Explanation:

Specifies the name of the DB area addition and modification option file created in (1) Creating a DB area addition and modification option file.

34.3.5 Expanding a DB area (adding DB area files using block special files)

This example expands a data DB area (adds data DB area files) by executing the adbmodarea command. It allocates block special files as the data DB area files, as shown in the following:



Explanation:

- Block special files (/dev/mapper/VolG01 and /dev/mapper/VolG02) have been allocated as the data DB area files for data DB area OLDAREA.
- This example allocates block special files (/dev/mapper/VolG03 and /dev/mapper/VolG04) as new data DB area files in DB area OLDAREA.

  The following is the definition of the data DB area that is to be expanded:

  DB area name: OLDAREA

  Number of DB area files: 2

  Initial allocation size for each DB area file: 500 megabytes

This example assumes the existence of a symbolic link destination directory and block special files and that the required access permissions have been granted.

The example uses the following procedure to execute the adbmodarea command.

Procedure

1. Create a DB area addition and modification option file.
2. Execute the adbmodarea command.

Using an example, the following subsections explain each of these steps.

(1) Creating a DB area addition and modification option file

This example creates a DB area addition and modification option file and specifies the definition of a data DB area that is to be expanded.

Example of a DB area addition and modification option file

```
adbexpandarea -n OLDAREA \                                    ...1
              -v /dev/mapper/VolG03,/dev/mapper/VolG04 \      ...2
              -q \                                            ...3
              -i 500M,2                                       ...4
```

Explanation:

1. Specifies the data DB area name (OLDAREA).
2. Specifies the block special files (/dev/mapper/VolG03 and /dev/mapper/VolG04) that are to be allocated as the data DB area files.
3. Specifies that the data DB area files are to be added in quick mode.
4. Specifies the initial allocation size (500 megabytes) of each data DB area file and the number of files (2) to be added.

(2) Executing the adbmodarea command

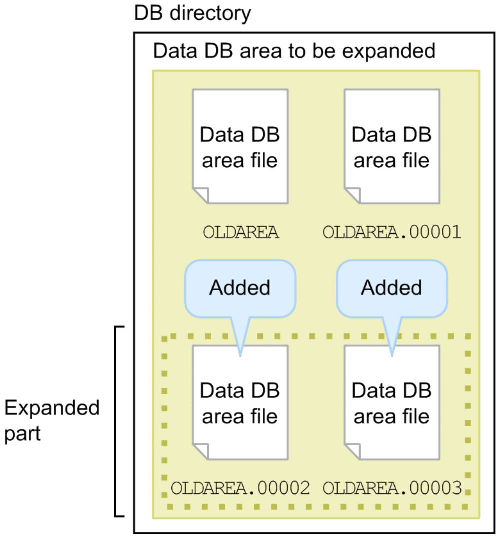
```
adbmodarea /HADB/server/conf/adbmodarea.opt
```

Explanation:

Specifies the name of the DB area addition and modification option file created in (1) Creating a DB area addition and modification option file.

34.3.6 Expanding a DB area (adding DB area files using regular files)

This example expands a data DB area (adds data DB area files) by executing the `adbmodarea` command. It adds data DB area files under the DB directory, as shown in the following:



Explanation:

- Data DB area file entities have been created in data DB area `OLDAREA` under the DB directory.
- The following is the definition of the data DB area to be expanded:
  DB area name: `OLDAREA`
  Number of DB area files to be added: 2
  Initial allocation size for each DB area file: 500 megabytes

The example uses the following procedure to execute the `adbmodarea` command.

Procedure

1. Create a DB area addition and modification option file.
2. Execute the `adbmodarea` command.

Using an example, the following subsections explain each of these steps.

(1) Creating a DB area addition and modification option file

This example creates a DB area addition and modification option file and specifies the definition of a data DB area that is to be expanded.

Example of a DB area addition and modification option file

```
adbexpandarea -n OLDAREA \                                 ...1
              -i 500M,2                                    ...2
```

Explanation:

1. Specifies the data DB area name (`OLDAREA`).
2. Specifies the initial allocation size (500 megabytes) for each DB area file and the number of files (2) to be added.

(2) Executing the adbmodarea command

```
adbmodarea /HADB/server/conf/adbmodarea.opt
```

Explanation:

Specifies the name of the DB area addition and modification option file created in (1) Creating a DB area addition and modification option file.

34.4 Return codes

The following table lists and explains the return codes of the adbmodarea command.

Table 34-5: Return codes of the adbmodarea command

Return code	Description
0	The adbmodarea command terminated normally.
4	The adbmodarea command terminated normally, but a warning message was issued. Check the displayed message or the message output to a message log file.
8	An error occurred during execution of the adbmodarea command. For the cause of the error, see the displayed message or the message that has been output to a message log file. If you are using the multi-node function, also check the message that has been output to a message log file on the HADB server on the slave node. An error might have occurred on the HADB server on the slave node. If the HADB server did not terminate abnormally, a message providing the HADB server's internal processing results is output to a message log file.

34.5 Notes

This section provides notes about executing the adbmodarea command.

34.5.1 Common notes

1. The adbmodarea command cannot be executed concurrently with any other application program or command.
2. A block special file that has already been allocated to a DB area file cannot be allocated as a new data DB area file.
3. Data DB area files cannot be created in a file system that does not support direct I/O.
4. In case you need to restore a DB area that was deleted by mistake, make sure that you make a backup of the database before you delete DB areas. For details about backups, see *Backing up a database* in the *HADB Setup and Operation Guide*.
5. When the adbmodarea command overwrites an existing file, it first deletes the target file, and then re-creates the file.
6. If the adbmodarea command terminates abnormally, eliminate the cause of the error, and then re-execute the command.
7. If the HADB terminates abnormally due to, for example, power outage during execution of the adbmodarea command, rollback might fail, causing the database to be placed in an invalid state. In such a case, execute the adbstart command to restart the HADB server.
8. If the HADB terminates abnormally due to, for example, power outage during execution of the adbmodarea command, unnecessary DB area files might remain. However, because remaining DB area files do not cause problems, you do not need to delete them.

   If you want to delete the remaining DB area files, use the following procedure:

   Step 1

   Execute the adbstart command to restart the HADB server. You cannot delete DB area files until the HADB server has restarted.

   Step 2

   Execute the adbdbstatus command with used specified for the -d option.

   Step 3

   Make sure that the DB area files you want to delete are not displayed in the DBarea_filename column in the usage information for DB areas, tables, and indexes output by the adbdbstatus command.

   DB area files not being used by the HADB server are not output to the DBarea_filename columns. Therefore, you can determine that those DB area files can be deleted. However, some files (such as ADBMST) that are not displayed in the DBarea_filename column might be used by the HADB server. Do not delete applicable files (such as ADBMST).

   Step 4

   Make sure that the HADB server is not using the DB area files you want to delete, and then delete them by using the OS's rm command or the like.
9. If the HADB server terminates abnormally due to, for example, power outage during execution of the adbmodarea command, a message indicating the processing results in the HADB server might not be output. If no message was output, restart the HADB server that terminated abnormally, and then execute the adbdbstatus command. From the output results of the adbdbstatus command, you can determine whether the changes of DB areas have been applied.

10. A global buffer is allocated automatically for a data DB area that is added by the adbmodarea command. If you want to allocate a specific buffer, specify the adbbuff operand in the server definition, and then restart the HADB server. For details about how to specify the adbbuff operand, see the topic *Operands and options related to global buffers (command format)* in *Detailed descriptions of the server definition operands* in *Designing the Server Definition* in the *HADB Setup and Operation Guide*.
11. If all data DB areas for which a global buffer has been allocated are deleted, the shared memory that was used by the global buffer is released.
12. When DB areas are added or deleted, the memory requirements listed below change; so, in such cases, re-estimate the memory requirements. For details about how to determine the variables, see the topic *Determining the memory requirement for starting the HADB server* in *Estimating the HADB server's memory requirement* in *Preparing Resources* in the *HADB Setup and Operation Guide*.
    - Memory size required for global buffer pages (SHM_BUFGLOBAL variable)
    - Required process common memory size (BUFGLOBAL variable)
13. If using SCSI reservation for shared disk for a cold standby configuration, you need to change the operand specified in the servers file for HA Monitor when you perform one of the following operations:
    - Adding a DB area by using the adbmodarea command
    - Expanding a DB area by using the adbmodarea command
    - Deleting a DB area by using the adbmodarea command

    Therefore, you need to terminate the cold standby configuration. For details about the operation procedure, see *Adding, deleting, or expanding data DB areas (in the case of the cold standby configuration)* in the *HADB Setup and Operation Guide*.

34.5.2 Notes about using the multi-node function

1. If you are using the multi-node function, we recommend that you execute the adbmodarea command while HADB servers are running on all nodes in the multi-node configuration. Execute the adbmodarea command on the HADB server on the master node. If any node has stopped, more complicated procedure is required for adding and expanding data DB areas. For details about the execution procedure, see the topic *Adding, deleting, or expanding data DB areas (when the multi-node function is being used)* in the *HADB Setup and Operation Guide*.
2. To add or expand a DB area while the multi-node function is being used, DB area files to be added must be allocated to block special files. The block special files must be able to be accessed from all nodes by using the same path name. Therefore, use block special files that include WWN. For details, see 20.3.2 Creating DB directories (when using the multi-node function).
3. If using SCSI reservation for shared disk, you need to change the operand specified in the servers file for HA Monitor when you perform one of the following operations:
    - Adding a DB area by using the adbmodarea command
    - Expanding a DB area by using the adbmodarea command
    - Deleting a DB area by using the adbmodarea command

    Therefore, you need to terminate the HADB server in the multi-node configuration. For details about the operation procedure, see *Adding, deleting, or expanding data DB areas (when the multi-node function is being used)* in the *HADB Setup and Operation Guide*.

35 adbmodbuff (Change Buffer)

This chapter explains the functionality provided by the `adbmodbuff` command and how to use the command.

35.1 Description

The `adbmodbuff` command changes the number of pages for the following buffer specified in the server definition while the HADB server is running:

- Local work table buffer

Note that a change made by the `adbmodbuff` command will be lost when the HADB server is terminated. To inherit the change at the next startup, apply the change to the server definition.

For details about how to modify the server definition, see the topic *Modifying the server definition* in *Creating and modifying a server definition* in *Building a System* in the *HADB Setup and Operation Guide*.

When executed, the `adbmodbuff` command establishes a connection to the HADB server.

35.2 Specification format for the adbmodbuff command

This section explains the specification format and the format of the buffer-modifying option for the `adbmodbuff` command.

The following OS users can execute the `adbmodbuff` command:

- HADB administrator

35.2.1 Explanation of the specification format and options

(1) Specification format

```
adbmodbuff buffer-modifying-option-file-name
```

(2) Explanation of options

- *buffer-modifying-option-file-name*

  ~<path name>((2 to 510bytes))

  Specifies the absolute path name of the buffer-modifying option file in which the buffer-modifying option is specified.

  For details about the buffer-modifying option, see 35.2.2 Format of the buffer-modifying option.

35.2.2 Format of the buffer-modifying option

(1) Specification format

```
set adb_dbbuff_wrktbl_clt_blk_num = number-of-local-work-table-buffer-pages
```

The specification rules for this buffer-modifying option are the same as for the server definition. For details about the specification rules for the server definition, see *Syntax rules for the server definition* in the *HADB Setup and Operation Guide*.

(2) Explanation of specification format

- `adb_dbbuff_wrktbl_clt_blk_num` = *number-of-local-work-table-buffer-pages*

  ~<integer>((5 to 100,000,000))

  Specifies the number of local work table buffer pages.

  Specifying this option enables you to change temporarily the value of the server definition's `adb_dbbuff_wrktbl_clt_blk_num` operand (in which case this option's value takes precedence).

  The value specified in this option will be lost when the HADB server is terminated.

  If either of the following is specified, the number of local work table buffer pages will not be changed despite that this option is specified:

  - `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition

- `adb_export_wrktbl_blk_num` export option

Note

- For details about the `adb_dbbuff_wrktbl_clt_blk_num` operand in the server definition, see the topic *Operands related to performance (set format)* in *Detailed descriptions of the server definition operands* in *Designing the Server Definition* in the *HADB Setup and Operation Guide*.
- For details about the `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition, see the topic *Operands related to performance* in *Contents of operands in the client definition* in *Designing Client Definitions* in the *HADB Application Development Guide*.
- For details about the `adb_export_wrktbl_blk_num` export option, see 14.2.2 Format of export options.
- For details about the SQL statements used to create local work tables, see the topic *Work tables created when SQL statements are executed* in *Considerations when executing an SQL statement that creates work tables* in *Designs Related to Improvement of Application Program Performance* in the *HADB Application Development Guide*.

35.3 Examples

This section explains by way of examples the use of the `adbmodbuff` command to change the number of buffer pages set in the server definition.

35.3.1 Changing the number of local work table buffer pages

This example uses the following procedure for using the `adbmodbuff` command to change the number of local work table buffer pages.

Procedure

1. Create a buffer-modifying option file.
2. Execute the `adbmodbuff` command.

Using an example, the following subsections explain each of these steps.

(1) Creating a buffer-modifying option file

This example creates a buffer-modifying option file and specifies the buffer-modifying option in that file.

Example of buffer-modifying option file

```
set adb_dbbuff_wrktbl_clt_blk_num = 256
```

Explanation:

Specifies 256 as the number of local work table buffer pages.

(2) Executing the adbmodbuff command

```
adbmodbuff /home/adbmanager/adbmodbuff.opt
```

Explanation:

Specifies the name of the buffer-modifying option file created in (1) Creating a buffer-modifying option file.

35.4 Return codes

The following table lists and describes the return codes of the `adbmodbuff` command.

Table 35-1: Return codes of the adbmodbuff command

Return code	Description
`0`	The `adbmodbuff` command terminated normally.
`4`	The `adbmodbuff` command terminated normally, but a warning message was output. Check the displayed warning message or the warning message output to the message log file and take appropriate action.
`8`	One of the following errors occurred: • The `adbmodbuff` command was not executed because it was invalid. • An error occurred during execution of the `adbmodbuff` command. For the cause of the error and the corrective action to take, see the displayed error message or the error message that has been output to a message log file.

35.5 Notes

1. The `adbmodbuff` command can be executed only while the HADB server is running.
2. While the `adbmodbuff` command is executing, another instance of the `adbmodbuff` command cannot be executed on the same node.
3. If you want to inherit the change made by the `adbmodbuff` command the next time the HADB server starts, make the same change in the server definition. For details about how to change the server definition, see *Modifying the server definition* in *Creating and modifying a server definition* in *Building a System* in the *HADB Setup and Operation Guide*.
4. If the multi-node function is used, only the buffer for the node on which the `adbmodbuff` command was executed is changed.

36 adbmonitor (Monitor the HADB Server)

This chapter explains the functionality provided by the `adbmonitor` command and how to use the command.

36.1 Description

The adbmonitor command monitors the status of the HADB server.

- When the multi-node function is not being used

  Execute the adbmonitor command to set the HADB server to be restarted automatically if it terminates abnormally. For details about the operating method, see the topic *Setting up the HADB server to restart automatically* in the *HADB Setup and Operation Guide*.

  In the cold standby configuration, the adbmonitor command is used with a shell command that is specified in HA Monitor's servers file (definition file used to set up an environment for a server).

- When the multi-node function is being used

  The adbmonitor command is used with a shell command that is specified in HA Monitor's servers file (definition file used to set up an environment for a server).

36.2 Specification format for the adbmonitor command

This section explains the specification format of and options for the `adbmonitor` command.

The following OS users can execute the `adbmonitor` command:

- HADB administrator

36.2.1 Explanation of the specification format and options

(1) Specification format

```
adbmonitor [-r] [-n]
```

(2) Explanation of options

- `-r`

  Specifies that the monitored HADB server is to be restarted automatically if it terminates abnormally. After the HADB server has been restarted automatically, this command will continue to monitor it.

  When the HADB server is restarted, a troubleshooting-information file (light version) is output under the server directory's `spool` directory. Delete this troubleshooting-information file (light version) on a regular basis. Otherwise, the disk that stores the server directory might run out of space.

  If this option is omitted and the monitored HADB server terminates abnormally, this command will terminate abnormally (with return code `8`).

- `-n`

  Specifies that monitoring of the target HADB server is to be continued without terminating this command if the HADB server terminates normally. If the monitored HADB server is restarted successfully after it has terminated normally, this command will start monitoring the HADB server again.

  If this option is omitted and the monitored HADB server terminates normally, this command will terminate normally (with return code `0`).

The following table shows the relationship between the HADB server's status and each option of the `adbmonitor` command.

Table 36-1: Relationship between HADB server's status and each option of the adbmonitor command

No.	HADB server status	Options			
		No options specified	**-r option only is specified**	**-n option only is specified**	**-r and -n options are both specified**
1	`ACTIVE`	Monitoring continues.	Monitoring continues.	Monitoring continues.	Monitoring continues.
2	`STOP`	`adbmonitor` command terminates normally.	`adbmonitor` command terminates normally.	Monitoring continues.	Monitoring continues.
3	`STARTING`	Monitoring continues.	Monitoring continues.	Monitoring continues.	Monitoring continues.
4	`STOPPING`				

No.	HADB server status	Options			
		No options specified	-r option only is specified	-n option only is specified	-r and -n options are both specified
5	`ABORT`	`adbmonitor` command terminates abnormally.	HADB server is restarted automatically and monitoring continues. A troubleshooting-information file (light version) is output.	`adbmonitor` command terminates abnormally.	HADB server is restarted automatically and monitoring continues. A troubleshooting-information file (light version) is output.
6	`QUIESCE`	Monitoring continues.	Monitoring continues.	Monitoring continues.	Monitoring continues.
7	`OFFLINE`				
8	`MAINTNCE`				
9	`CHGMODE`				
10	`STOPWAIT`				
11	`COREDUMP`				
12	`FORCE`	`adbmonitor` command terminates abnormally.	`adbmonitor` command terminates abnormally.	`adbmonitor` command terminates abnormally.	`adbmonitor` command terminates abnormally.

36.3 Return codes

The following table lists and describes the return codes of the `adbmonitor` command.

Table 36-2: Return codes of the adbmonitor command

Return code	Description
`0`	The HADB server being monitored by the `adbmonitor` command terminated normally.
`8`	An HADB server that is monitored by the `adbmonitor` command was forcibly or abnormally terminated.

37 adbreorgsystemdata (Reorganize System Table)

This chapter explains the functionality provided by the `adbreorgsystemdata` command and how to use the command.

Note that in this chapter, the term *system table* means *system table (base table)*.

37.1 Description

This section explains the functionality provided by the adbreorgsystemdata command.

When the adbreorgsystemdata command is executed, it establishes a connection to the HADB server.

37.1.1 Function overview

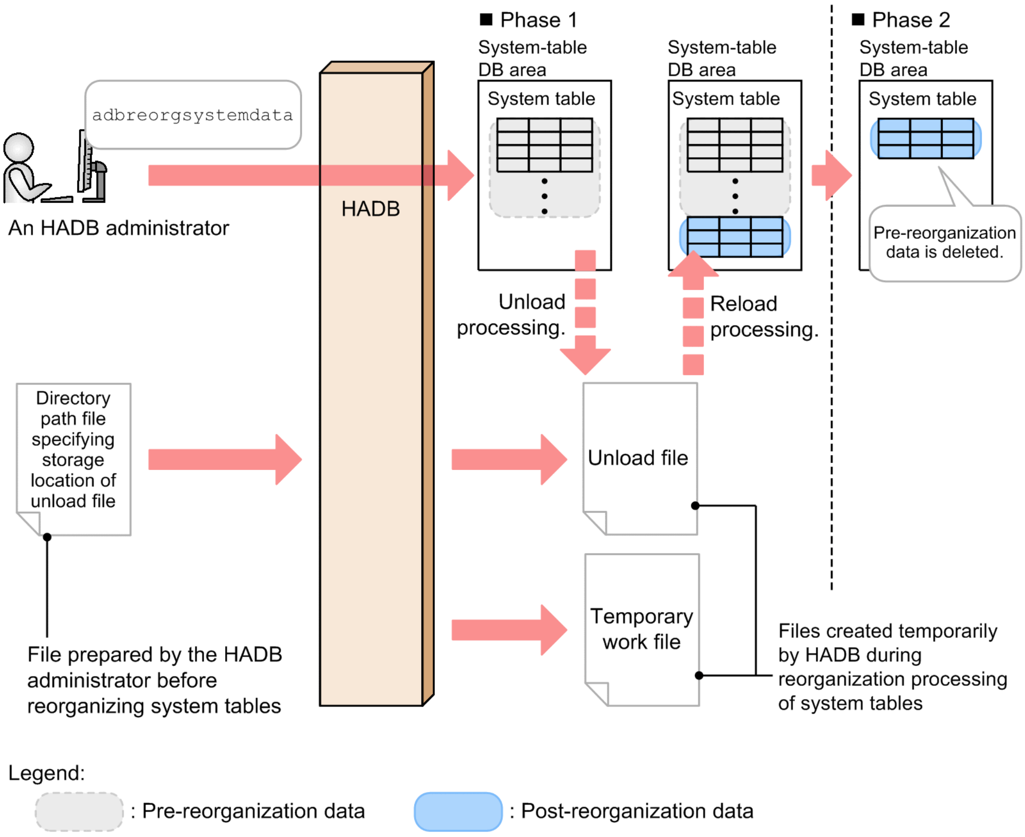
If you execute an SQL statement or command on a base table, invalid row data might be added to a system table. If invalid row data continues to increase in a system table, the following consequences occur:

- Free disk space for storing system tables becomes insufficient.
- The increased number of pages in the DB area for system tables, which are referenced during searches, degrades the performance of the search function.

Executing the adbreorgsystemdata command reorganizes the system table data, thus freeing up space occupied by invalid row data. This processing is called *system table reorganization processing*.

The following figure provides an overview of system table reorganization processing.

Figure 37-1: Overview of system table reorganization processing


■ Phase 1
System-table DB area
System table
System-table DB area
System table
■ Phase 2
System-table DB area
System table
Pre-reorganization data is deleted.
adbreorgsystemdata
An HADB administrator
HADB
Unload processing.
Reload processing.
Directory path file specifying storage location of unload file
Unload file
Temporary work file
File prepared by the HADB administrator before reorganizing system tables
Files created temporarily by HADB during reorganization processing of system tables
Legend:
: Pre-reorganization data
: Post-reorganization data

Explanation:

System table reorganization processing is executed in the following two phases:

1. Phase 1

System table data, with the invalid row data removed, is stored in an unload file (unload processing). Next, the data in the unload file is re-stored in system tables as reorganized data (reload processing).

2. Phase 2

   The pre-reorganization data is deleted, thus freeing up the space occupied by the invalid row data in the system tables.

The following explains the files:

- **Temporary work files**

  These are the work files that are created temporarily during reorganization processing of system tables. The created temporary work files are deleted automatically when the reorganization processing is finished.

- **Unload files**

  These are the files that are created temporarily during the reorganization processing of system tables. System table data, with the invalid row data removed, is stored in these files. The created unload files are deleted automatically when the reorganization processing is finished.

Note

For details about when invalid row data might increase in system tables, see *Reason for reorganizing a system table* in the *HADB Setup and Operation Guide*.

37.1.2 Optional functionality of the adbreorgsystemdata command

The `adbreorgsystemdata` command has the following optional functionality:

- Functionality for deleting pre-reorganization data

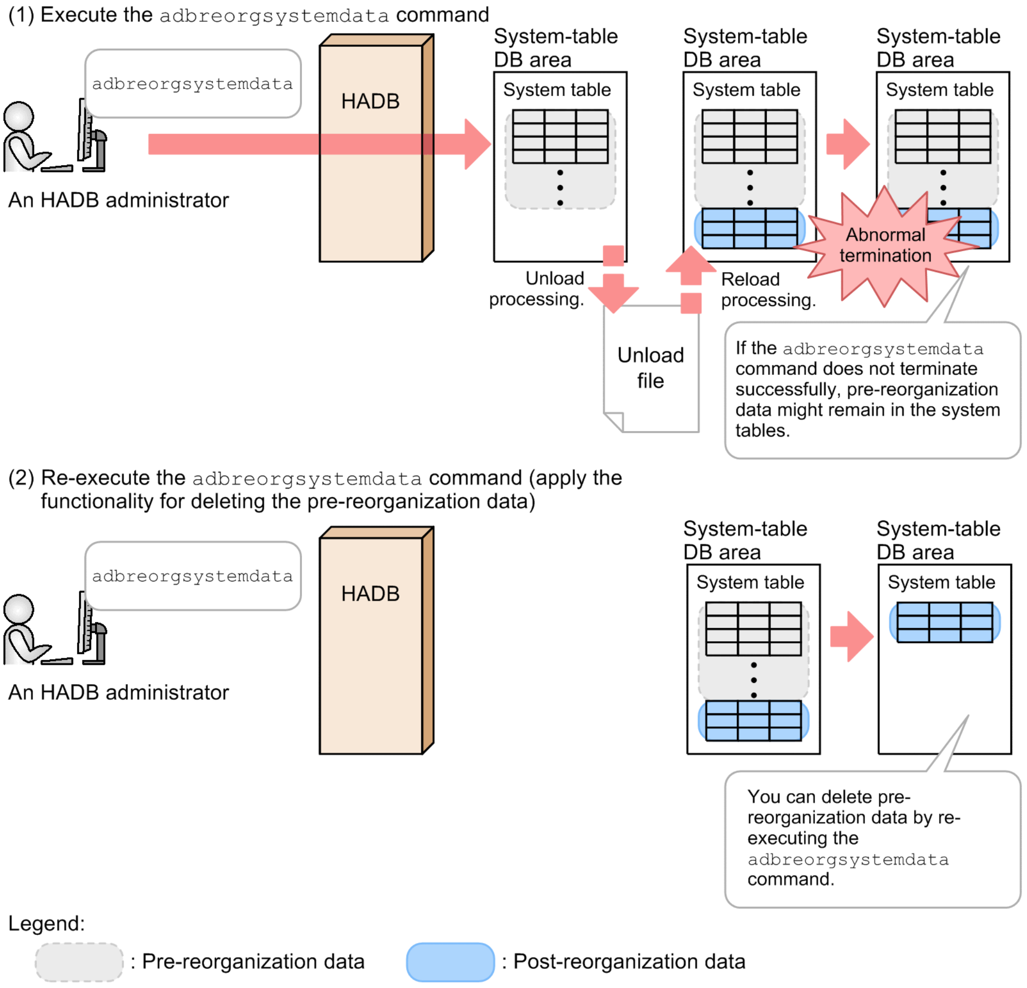
(1) Functionality for deleting pre-reorganization data

If the `adbreorgsystemdata` command does not terminate normally, pre-reorganization data might remain in the system table.

If this happens, you can delete the remaining data by re-executing the `adbreorgsystemdata` command on the same system table. This functionality is called the *functionality for deleting pre-reorganization data*.

The following figure gives an overview of the functionality for deleting pre-reorganization data.

Figure 37-2: Overview of the functionality for deleting pre-reorganization data


(1) Execute the adbreorgsystemdata command
adbreorgsystemdata
An HADB administrator
HADB
System-table DB area
System table
System-table DB area
System table
System-table DB area
System table
Abnormal termination
Unload processing.
Reload processing.
Unload file
If the adbreorgsystemdata command does not terminate successfully, pre-reorganization data might remain in the system tables.
(2) Re-execute the adbreorgsystemdata command (apply the functionality for deleting the pre-reorganization data)
adbreorgsystemdata
An HADB administrator
HADB
System-table DB area
System table
System-table DB area
System table
You can delete pre-reorganization data by re-executing the adbreorgsystemdata command.
Legend:
: Pre-reorganization data
: Post-reorganization data

Note

If other processing is referencing the pre-reorganization data, the `adbreorgsystemdata` command results in an error. The pre-reorganization data is not deleted.

For cases where pre-reorganization data remains in system tables, see 37.4 Return codes.

37.2 Specification format for the adbreorgsystemdata command

This section explains the specification format and options for the `adbreorgsystemdata` command.

The following OS user can execute the `adbreorgsystemdata` command:

- HADB administrator

37.2.1 Explanation of the specification format and options

(1) Specification format

```
adbreorgsystemdata [-g output-interval-for-reorganization-processing-progress-message
s]
                   [-w name-of-directory-storing-temporary-work-files]
                   [--timeout timeout-period]
                     -c table
                     -f directory-path-file-name-specified-for-storing-unload-files
                     -n table-to-be-processed
```

(2) Explanation of options

● `-g` *output-interval-for-reorganization-processing-progress-messages*

~<integer>((0 to 1,000,000,000))<<0>>(in 1,000 rows)

Specifies the interval at which to output progress messages for reorganization processing. For example, if `2` is specified in this option, the `KFAA80205-I` progress message is output each time 2,000 rows of data have been processed.

Note that separate progress messages are output for unload processing and reload processing.

If this option is omitted or `0` is specified in this option, no progress message is output.

● `-w` *name-of-directory-storing-temporary-work-files*

~<OS path name>((2 to 518 bytes))<<$DBDIR/ADBWORK>>

Specifies, using an absolute path, a storage location for the temporary work files that are created during reorganization processing.

The rule for specifying this option is as follows:

- Do not specify a forward slash (`/`) at the end of a path name.

For the estimated space occupied by the temporary work files that will be created under the directories specified in the `-w` option, see *Estimating the size of the temporary work file for executing the adbreorgsystemdata command* in the *HADB Setup and Operation Guide*. If this option is omitted, temporary work files are created under `$DBDIR/ADBWORK`.

If the disk that stores temporary work files runs out of free space, specify another disk with a larger capacity as the storage location. For details about what to do if the disk that stores temporary work files runs out of free space, see *Steps to take in the event of a shortage of disk space for storing temporary work files during command execution* in the *HADB Setup and Operation Guide*.

● `--timeout` *timeout-period*

~<integer>((0 to 65,535))<<3,600>>(units: seconds)

Specifies the amount of time until the `adbreorgsystemdata` command times out.

If the execution time of the `adbreorgsystemdata` command exceeds the time specified in this option, the processing of the `adbreorgsystemdata` command is canceled.

If this option is not specified, the `adbreorgsystemdata` command times out at 3,600 seconds (1 hour).

If `0` is specified for this option, no timeout period is set for the `adbreorgsystemdata` command. Even if execution of the `adbreorgsystemdata` command takes a long time, the command processing continues without timing out.

> **Note**
>
> If execution of the `adbreorgsystemdata` command takes such a long time that system operation is affected, consider setting a timeout period.

▪ **If you are using the multi-node function**

In the following statuses, the processing of the `adbreorgsystemdata` command is not canceled, even after the timeout period specified in this option has elapsed. The processing of the `adbreorgsystemdata` command is canceled after the pending status is resolved.

- Pending switch to master node
- Pending node recovery
- Pending ongoing transaction on another node

● `-c table`

Specifies a table as the processing target of the `adbreorgsystemdata` command.

● `-f` *directory-path-file-name-specified-for-storing-unload-files*

~<OS path name>((2 to 510 bytes))

Specifies, using an absolute path, the name of the file that specifies the storage locations for unload files that are created temporarily during reorganization processing.

The following shows an example specification of the directory path file that specifies the storage locations of unload files.

Example specification of a directory path file

```
/home/adbmanager/unld
```

The rules for specifying directory path files are as follows:

- One directory for storing unload files can be specified in a directory path file.
- The storage directory for unload files must be expressed as an absolute path consisting of 2 to 510 bytes.
- The same characters can be used in the path name for the storage directory for unload files, as for the name of the directory path file that specifies storage locations for unload files.
- Spaces in the path name of the directory for storing unload files are not ignored. Spaces at the beginning and end of a path name are also treated as part of the path name.
- The only rows that are ignored in a directory path file are those that include line break characters.

For the estimated space occupied by the unload files that will be created under the directories specified in the `-f` option, see *Estimating the size of an unload file* in the *HADB Setup and Operation Guide*.

If the disk that stores unload files runs out of free space, specify another disk with a larger capacity as the storage destination. For details about what to do if there is insufficient space on the disk that stores unload files, see *Steps to take in the event of a shortage of disk space for storing unload files* in the *HADB Setup and Operation Guide*.

● `-n` *table-to-be-processed*

Specifies the target system table of reorganization processing.

You can specify the names of the following system tables as tables to be processed:

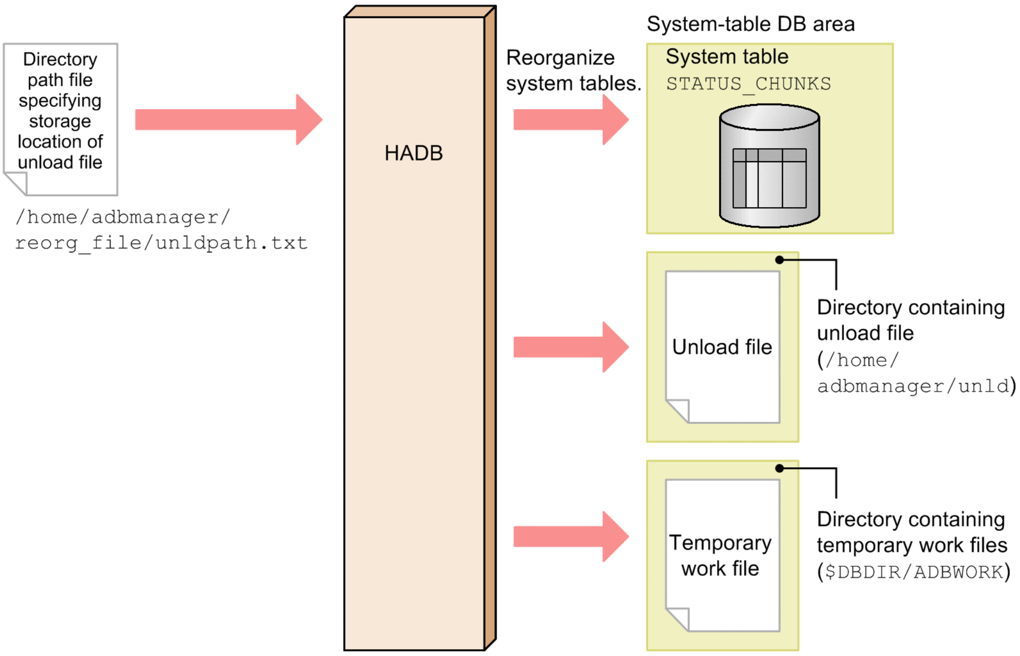
- `STATUS_TABLES`
- `STATUS_COLUMNS`
- `STATUS_INDEXES`
- `STATUS_CHUNKS`
- `STATUS_SYNONYM_DICTIONARIES`

37.3 Examples

This section uses examples to explain how to reorganize data by using the adbreorgsystemdata command.

37.3.1 Reorganizing system tables

In this example, the system table STATUS_CHUNKS stored in the system-table DB area will be reorganized.



Directory path file specifying storage location of unload file
/home/adbmanager/reorg_file/unldpath.txt
HADB
Reorganize system tables.
System-table DB area
System table
STATUS_CHUNKS
Unload file
Directory containing unload file
(/home/adbmanager/unld)
Temporary work file
Directory containing temporary work files
($DBDIR/ADBWORK)

This example uses the following procedure to execute the adbreorgsystemdata command.

Procedure

1. Create a directory path file that specifies storage locations for unload files.
2. Execute the adbreorgsystemdata command.

Using examples, the following subsections explain each of these steps.

(1) Creating a directory path file that specifies storage locations for unload files

Create a directory path file (/home/adbmanager/reorg_file/unldpath.txt) that specifies the storage directory for unload files, and then specify the storage directory for unload files.

Example specification of the directory path file that specifies the storage locations of unload files

```
/home/adbmanager/unld
```

(2) Executing the adbreorgsystemdata command

```
adbreorgsystemdata --timeout 300                                 ...1
                   -c table                                      ...2
```

```
                     -f /home/adbmanager/reorg_file/unldpath.txt    ...3
                     -n STATUS_CHUNKS                               ...4
```

Explanation:

1. Specify 300 seconds (5 minutes) as the amount of time until the `adbreorgsystemdata` command times out.
2. Specify the target table to undergo reorganization processing.
3. Specify the name of the directory path file that specifies storage locations for unload files, which you created in (1) Creating a directory path file that specifies storage locations for unload files.
4. Specify the system table `STATUS_CHUNKS` as the table to be processed.

37.4 Return codes

The following table lists and describes the return codes of the adbreorgsystemdata command.

Table 37-1: Return codes of the adbreorgsystemdata command

Return code	Description
0	The adbreorgsystemdata command terminated normally.
4	The adbreorgsystemdata command terminated normally, but a warning message was issued. Check the displayed message or the message output to a message log file and take appropriate action.
8	An error occurred during execution of the adbreorgsystemdata command. For the cause of the error, see the displayed error message or the message that has been output to a message log file. At this time, the table to be processed is in its state before the adbreorgsystemdata command was executed. After eliminating the cause of the command terminating abnormally, re-execute the adbreorgsystemdata command.
12	An error occurred during execution of the adbreorgsystemdata command. For the cause of the error, see the displayed error message or the message that has been output to a message log file. At this time, both the pre-reorganization data and the post-reorganization data exist in the table to be processed. After eliminating the cause of the command terminating abnormally, re-execute the adbreorgsystemdata command.
16	Communication with the HADB server was lost during execution of the adbreorgsystemdata command. For the cause of the loss of communication with the HADB server, see the displayed error message or the message that has been output to a message log file. Note that if the adbreorgsystemdata command times out, the KFAA96609-E message is output. For details about corrective action to take if this return code is output, see the return codes output to the KFAA80204-I message (0, 4, 8, or 12). However, if the HADB server process is forcibly terminated by using the adbstop --force command during execution of the adbreorgsystemdata command, the KFAA80204-I message might not be output. If this happens, start the HADB server, and then re-execute the adbreorgsystemdata command.

37.5 Handling abnormal termination of the adbreorgsystemdata command

Measures to take when the `adbreorgsystemdata` command terminates abnormally differ depending on the command's return code.

For details about the measures to take, see 37.4 Return codes.

37.6 Notes

1. The `adbreorgsystemdata` command uses three processing real threads. If the `adbreorgsystemdata` command is executed while another command or application program is running, HADB might not be able to allocate the required number of processing real threads. If this happens, HADB waits until the required number of processing real threads can be allocated before executing the `adbreorgsystemdata` command. If allocation of processing real threads is placed in wait status, see *Points to consider about the number of processing real threads to be used during command execution* in the *HADB Setup and Operation Guide* and take appropriate action.

   In addition, the number of processing real threads that can be used depends on whether the client-group facility is applied. For details, see *Points to consider when specifying the number of processing real threads for a group* in the *HADB Setup and Operation Guide*.

2. Do not perform read or write operations on unload files by using other applications while the `adbreorgsystemdata` command is being executed.

3. Use `X'0A'` (`LF`) or `X'0D0A'` (`CRLF`) for the line break code in any file specified in the options of the `adbreorgsystemdata` command.

4. Use the `adbcancel` command to cancel execution of the `adbreorgsystemdata` command. For details about how to use the `adbcancel` command to suspend an executing command, see 1.6 When suspending (forcibly terminating) an executing command.

5. System table reorganization processing is executed in server processes. Therefore, using the **Ctrl**+**C** keys to forcibly terminate the `adbreorgsystemdata` command process will not interrupt reorganization processing of system tables that is being performed by server processes (such processing continues). When the reorganization processing is complete, the `KFAA81001-I` message is output to the message log file.

6. Work files (temporary work files) are created temporarily during reorganization processing of system tables. If the `adbreorgsystemdata` command terminates normally (the return code is `0` or `4`), and if deletion of pre-reorganization data fails (the return code is `12`), temporary work files are deleted automatically. However, if the `adbreorgsystemdata` command does not terminate normally, temporary work files might not be deleted.

   At this time, if you re-execute the `adbreorgsystemdata` command without changing the table to be processed or the `-w` option, the remaining temporary work files are deleted automatically when the command terminates normally. Therefore, it is not a problem if temporary work files remain. However, if you want to increase free disk space before re-executing the `adbreorgsystemdata` command, delete the temporary work files.

7. During the reorganization processing of system tables, files (unload files) are created temporarily to store the data of system tables with the invalid row data removed. If the `adbreorgsystemdata` command terminates normally (the return code is `0` or `4`), and deletion of the pre-reorganization data fails (the return code is `12`), the unload files are deleted automatically. However, if the `adbreorgsystemdata` command does not terminate normally, unload files might not be deleted.

   At this time, if you re-execute the `adbreorgsystemdata` command without changing the table to be processed or the `-f` option, the remaining unload files are deleted automatically when the command terminates normally. Therefore, it is not a problem if unload files remain.

   If you change the `-f` option before re-executing the `adbreorgsystemdata` command, check the storage directory of the unload files from before the change. If any unload files remain, delete them.

   Note that one unload file is output per table to be processed. Names of unload files are written as follows.

   Unload file name

   ```
   reorganize-unload-[table-ID]-00001
   ```

   table-ID: Table ID of the table to be processed, converted to a hexadecimal number expressed as an 8-digit string

38 adbsql (Execute SQL Statements)

This chapter explains the functionality provided by the adbsql command and how to use the command.

38.1 Description

The adbsql command enables you to execute SQL statements interactively. You can enter an SQL statement from the standard input and then output the execution results to the standard output.

The adbsql command establishes a connection to the HADB server as an HADB client.

The adbsql command can execute the following SQL statements.

Definition SQL

- ALTER TABLE
- ALTER USER
- ALTER VIEW
- CREATE AUDIT
- CREATE INDEX
- CREATE SCHEMA
- CREATE TABLE
- CREATE USER
- CREATE VIEW
- DROP AUDIT
- DROP INDEX
- DROP SCHEMA
- DROP TABLE
- DROP USER
- DROP VIEW
- GRANT
- REVOKE

Data manipulation SQL

- DELETE
- INSERT
- SELECT
- UPDATE
- TRUNCATE TABLE
- PURGE CHUNK

You can specify dynamic parameters in these SQL statements. For details, see 38.6 Operating method when dynamic parameters are specified in an SQL statement.

Control SQL

- COMMIT
- ROLLBACK

You can also execute `adbsql` subcommands. The `adbsql` subcommands are used for operations such as disconnecting from HADB. For details about the `adbsql` subcommands, see 38.5 adbsql subcommands.

Prerequisites for using the adbsql command

The `adbsql` command uses the client definition to connect to the HADB server. Therefore, you must set up an HADB client environment and specify environment variables. For details about setting up an HADB client environment and specifying environment variables, see *Setting Up an Environment for an HADB Client (If the ODBC Driver and CLI Functions Are Used)* in the *HADB Application Development Guide*.

To connect to the HADB server, you must use the authorization identifier of an HADB user with the `CONNECT` privilege.

38.2 Specification format for the adbsql command

This section explains the specification format for the `adbsql` command and its options.

All OS users can execute the `adbsql` command.

38.2.1 Explanation of the specification format and options

(1) Specification format

```
adbsql
  [-u authorization-identifier [-p password]]
  [-s [-b]]
  [-V]
  [-d]
```

(2) Explanation of options

- `-u` *authorization-identifier*

  ~<character string>((1 to 100 bytes))

  Specifies the authorization identifier to be used to connect to the HADB server. Specify the authorization identifier of an HADB user who has the `CONNECT` privilege.

  For this option, specify an authorization identifier in the range from 1 to 100 bytes. Note that the byte count (1 to 100 bytes) does not include the double quotation marks used to enclose the authorization identifier.

  > **Important**
  >
  > If the character string used as the authorization identifier includes any lowercase letter or a backslash (`\`), make sure that you check the rules for specifying authorization identifiers. For the rules for specifying authorization identifiers, see 1.4.2 Rules for specifying authorization identifiers and passwords.

- `-p` *password*

  ~<character string>((1 to 255 bytes))

  Specifies the password for the authorization identifier that is specified in the `-u` option.

  > **Important**
  >
  > If the password includes a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), make sure that you check the rules for specifying passwords. For the rules for specifying passwords, see 1.4.2 Rules for specifying authorization identifiers and passwords.

- `-s`

  Specifies that the search results of a `SELECT` statement or of the `#GETDATA` subcommand of the `adbsql` command are to be output to standard output in CSV format. If you redirect the search results to a file, you can create an input data file that can be used by the `adbimport` command.

  The following items are output:

  - Search results of `SELECT` statements and `#GETDATA` subcommands of the `adbsql` command
  - Error messages of `SELECT` statements and `#GETDATA` subcommands of the `adbsql` command

  The following rules and notes apply:

  - When you specify the `-s` option, you must also specify the `-u` and `-p` options.
  - The `-s` and `-V` options are mutually exclusive.
  - When the `adbsql` command is started with the `-s` option specified, `#SET DISPLAY` of the `adbsql` subcommand cannot be executed.
  - The double quotation mark (`"`) is output as the enclosing character and the comma (`,`) is output as the delimiter.
  - If a character string has a length of 0, only the enclosing characters are output.
  - No enclosing characters are output for the null value.
  - No SQL statements other than the `SELECT` statement can be specified.

- `-b`

  When you specify the `-s` option, specify this option as well if you want to output binary data in binary format.

- `-V`

  Specifies that not only the execution results of the SQL statement but also the contents of the executed SQL statement are to be output. If an `adbsql` subcommand is executed, not only the execution results of the `adbsql` subcommand but also the contents of the executed `adbsql` subcommand are to be output.

  The following shows the difference between when this option is specified and when it is not specified.

  Executing an SQL statement from a file (`infile`) by redirecting

  Example 1:

  ```
  adbsql < infile
  ```

  In this example, only the execution results of the SQL statement are output to the standard output.

  Example 2:

  ```
  adbsql -V < infile
  ```

  In this example, the execution results of the SQL statement and the contents of the executed SQL statement are output to the standard output.

  Outputting the execution results of an SQL statement to a file (`outfile`) by redirecting

  Example 3:

  ```
  adbsql > outfile
  ```

  In this example, only the execution results of the SQL statement are output to the file.

  Example 4:

  ```
  adbsql -V > outfile
  ```

  In this example, the execution results of the SQL statement and the contents of the executed SQL statement are output to the file.

- `-d`

  When scaling of decimal data is 0, specify this option if you want to output execution results with the decimal point omitted.

  Example: Output example if the execution result is 10.

  - If the `-d` option is not specified: `10.`

- If the `-d` option is specified: `10`

38.3 Examples

This section explains by way of examples how to use the adbsql command to execute SQL statements.

Important

When you use the adbsql command to execute an SQL statement, a semicolon (;) must be specified at the end of the SQL statement. Always make sure that the semicolon is specified.

38.3.1 Executing an SQL statement from the standard input

This example executes a SELECT statement to search table T1.

1. Start the adbsql command.

   ```
   adbsql
   ```

   If the -u option (authorization identifier) is specified in the command, step 2 is skipped and step 3 is performed. If the -u option (authorization identifier) and the -p option (password) are both specified in the command, steps 2 and 3 are skipped and step 4 is performed.

2. Enter the authorization identifier as the response to the input request.

   ```
   USER-ID ?
   ADBUSER01
   ```

   Specify the authorization identifier of an HADB user who has the CONNECT privilege and who has the SELECT privilege for table T1, and then press the **Enter** key. The authorization identifier entered here is used to connect to the HADB server.

3. Enter the password.

   ```
   PASSWORD ?
   ```

   Specify the password for the authorization identifier of the HADB user that connects to the HADB server, and then press the **Enter** key. The entered characters are not displayed.

4. Execute the SELECT statement.

   ```
   COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
   SELECT 'C1 = ',"C1" FROM "T1";                          ...1
    EXP0001_NO_NAME C1                                     ...2
    --------------- ----------------
              C1 =   -123456789.1234
              C1 =  ****************
   KFAA96404-I 2 rows were selected.
   ```

   Explanation:

   1. This is the SELECT statement that is to be executed. Specify a semicolon (;) at the end of the SQL statement.
   2. This line and below are the execution results of the SELECT statement. For the display format of SQL statement execution results, see 38.7 Output format of SQL statement execution results.

5. Terminate the adbsql command.

   ```
   COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
   #EXIT;
   ```

Execute the `adbsql` command's `#EXIT` subcommand to terminate the `adbsql` command.

About the prompts that are displayed

The prompts explained in the following table are displayed as you execute the `adbsql` command.

Table 38-1: Prompts that are displayed for the SQL statement or adbsql subcommand entry window

No.	Prompt that is displayed	Description
1	`USER-ID ?`	Waiting for entry of the authorization identifier that is needed to connect to the HADB server. This prompt is displayed after the `adbsql` command or the `adbsql` command's `#CONNECT` subcommand has executed. Note that if the `-u` option is specified in the `adbsql` command that is executed, this prompt is not displayed. For the authorization identifier, you can specify a character string of 1 to 100 bytes. Note that the byte count (1 to 100 bytes) does not include the double quotation marks used to enclose the authorization identifier. If no authorization identifier is entered when its entry is requested, `ADBUSER` is assumed to be the authorization identifier.
2	`PASSWORD ?`	Waiting for entry of the password. If the `-u` and `-p` options are specified in the `adbsql` command that is executed, this prompt is not displayed. The character string for the entered password is not displayed. For the password, you can specify a character string of 1 to 255 bytes. The password cannot be omitted.
3	`COMMAND ?`	Normal entry wait status.
4	`NEXT ?`	Waiting for entry of a continuation line. If the **Enter** key is pressed before entry of a semicolon (`;`), indicating the end of the SQL statement or `adbsql` subcommand, that line is treated as a line to be continued on a continuation line (that is, the entry of the SQL statement or `adbsql` subcommand has not been completed). When this prompt is displayed, enter the remainder of the SQL statement or of the `adbsql` subcommand.
5	`DATA(`*n*`) ?`	Waiting for the data for a dynamic parameter. When `COMMAND ?` is displayed after this prompt, enter the data for the dynamic parameter.

38.3.2 Executing an SQL statement specified in a file

This example uses the `CREATE TABLE` statement to define table `T1`.

1. Specify the `CREATE TABLE` statement in the file (`infile`).

```
ADBUSER01                                        ...1
#HelloHADB_01                                    ...2
CREATE FIX TABLE "T1"                            ...3
  ("C1" INT,"C2" CHAR(10),"C3" CHAR(7))
  IN DBAREA01;
```

Explanation:

1. Specifies on the file's first line the authorization identifier of an HADB user with the `CONNECT` privilege and the schema definition privilege.

2. Specifies on the second line the password for the authorization identifier of the HADB user.

3. Specifies the `CREATE TABLE` statement. Make sure that you enter a semicolon (`;`) at the end of the SQL statement.

2. Start the `adbsql` command.

```
adbsql -V < infile
```

This example specifies the -V option to display the SQL statement that is executed.

3. Check the execution results.

```
  USER-ID ?
ADBUSER01

  PASSWORD ?
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
CREATE FIX TABLE "T1"                     ...1
NEXT ?        +----2----+----3----+----4----+----5----+----6----+----7----+
  ("C1" INT,"C2" CHAR(10),"C3" CHAR(7))   ...1
NEXT ?        +----2----+----3----+----4----+----5----+----6----+----7----+
  IN DBAREA01;                            ...1
KFAA96403-I SQL processing completed. ...2
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
```

Explanation:

1. Displays the executed SQL statement because the -V option was specified.
2. A message indicating that the CREATE TABLE statement has executed successfully is displayed.

Important points

- Make sure that the text in the file is encoded in the character encoding specified in the ADBCLTLANG environment variable.
- For the line-ending code, use X'0A' (LF) or X'0D0A' (CRLF).
- #EXIT is executed automatically even if you omit #EXIT and #DISCONNECT from the file.
- If the adbsql command's #DISCONNECT and #EXIT subcommands are preceded by an EOF control character, the DISCONNECT processing is performed at that point.
- For the password, 1 to 255 bytes of characters can be specified.

38.3.3 Outputting the SQL statement's execution results to a file

This example uses the SELECT statement to search table USERSLIST, and outputs the execution results to a file (outfile).

1. Start the adbsql command.

```
adbsql -V > outfile
```

This example specifies the -V option to output the SQL statement to be executed to a file (outfile).

2. Specifies the authorization identifier.

```
adbsql -V > outfile
ADBUSER01 ← Specify the authorization identifier of the HADB user, and then press
the Enter key.
```

Specify the authorization identifier of an HADB user who has the CONNECT privilege and who has the SELECT privilege for table USERSLIST. This HADB user authorization identifier is used to connect to the HADB server.

3. Enter the password.

```
adbsql -V > outfile
ADBUSER01
         ← Specify the password for the authorization identifier, and then press t
he Enter key.
```

Specify the password for the authorization identifier of the HADB user that connects to the HADB server. The entered characters are not displayed.

4. Execute the SELECT statement

```
adbsql -V > outfile
ADBUSER01
SELECT "USERID","NAME","SEX" FROM "USERSLIST";
```

5. Terminate the adbsql command.

```
adbsql -V > outfile
ADBUSER01
SELECT "USERID","NAME","SEX" FROM "USERSLIST";
#EXIT;
```

This example executes the adbsql command's #EXIT subcommand to terminate the adbsql command.

6. The SELECT statement execution results are output to the file (outfile).

```
  USER-ID ?
ADBUSER01

  PASSWORD ?
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
SELECT "USERID","NAME","SEX" FROM "USERSLIST";        ...1
 USERID NAME                           SEX            ...2
 ------ ------------------------------ ---
 U00555 Mike Johnson                     M
 U00358 Nancy White                      F
 U00212 Maria Gomez                      F
 U00687 Taro Tanaka                      M
 U00869 Bob Clinton                      M
KFAA96404-I 5 rows were selected. COMMAND ?     +----2----+----3----+----4----+---
-5----+----6----+----7----+
#EXIT;
```

Explanation:

1. This is the executed SELECT statement.
2. The lines beginning with this line are the execution results of the SELECT statement. For the display format of SQL statement execution results, see 38.7 Output format of SQL statement execution results.

The text output to the file is encoded in the character encoding specified in the ADBCLTLANG environment variable.

The entry history of adbsql subcommands is also output to the file.

38.4 Return codes

The following table lists and describes the return codes of the `adbsql` command.

Table 38-2: Return codes of the adbsql command

Return code	Description
`0`	The `adbsql` command terminated normally.
`4`	At least one SQL statement error or `adbsql` subcommand error occurred.
`8`	Execution of the `adbsql` command failed for a reason such as an invalid option.

38.5 adbsql subcommands

This section explains the functions and input formats of the `adbsql` subcommands.

38.5.1 #CHUNKS (displays chunk information)

Function

This subcommand displays chunk information for the specified table.

Format

```
#CHUNKS [schema-name.]table-identifier [SHOW {ALL|COMMENT}];
```

`[`*schema-name*`.]`*table-identifier*

Specifies the name of a table for which chunk information is to be displayed, in the format *schema-name*`.`*table-identifier*. If the schema name is omitted, the authorization identifier of the HADB user who executed this subcommand is assumed.

For details about the specification rules that apply when a schema name or table identifier includes any lowercase letter or backslash (\), see 1.4.3 Table name specification rules.

`[SHOW {ALL|COMMENT}]`

If `SHOW ALL` is specified

All columns in the `STATUS_CHUNKS` system table are displayed. For details about the displayed information, see *Content of STATUS_CHUNKS* in the *HADB Setup and Operation Guide*.

If `SHOW COMMENT` is specified

Information for all items shown in the following table is displayed.

If this option is omitted (neither `SHOW ALL` nor `SHOW COMMENT` is specified)

Information shown in the following table is displayed. However, the `CHUNK_COMMENT` column is not displayed.

Table 38-3: Information displayed by #CHUNKS (if SHOW COMMENT is specified or this option is omitted)

No.	Column name	Description
1	`CHUNK_ID`	Chunk ID
2	`IS_CURRENT_CHUNK`	Current chunk • `Y`: Current chunk • `N`: Not the current chunk
3	`CHUNK_STATUS`	Chunk status For details, see `CHUNK_STATUS` in *Content of STATUS_CHUNKS* in the *HADB Setup and Operation Guide*.
4	`IS_ARCHIVE`	Whether the chunk is an archived chunk • `Y`: Is archived chunk • `N`: Is not archived chunk
5	`STORAGE_FORMAT`	Chunk-data storage format • '`COLUMN`' Column store format • '`ROW`' Row store format

No.	Column name	Description
6	CREATE_TIME	Date and time the chunk was created For details, see CREATE_TIME in *Content of STATUS_CHUNKS* in the *HADB Setup and Operation Guide*.
7	CHUNK_COMMENT	Comment specified in a chunk The null value is stored if no comment is set. This is displayed only when the SHOW COMMENT option is specified.

Important points

- The scope of chunk information in tables displayed by executing #CHUNKS varies depending on the privileges HADB users have. The following describes the scope of chunk information in tables to be displayed.

  When the authorization identifier of an HADB user with the DBA privilege is used to connect to the HADB server

  Chunks in tables defined by all HADB users

  When the authorization identifier of an HADB user without the preceding privilege is used to connect to the HADB server

  - Chunks in the tables defined by this HADB user (with the authorization identifier that was used for the current connection to the HADB server)
  - Chunks in tables for which access privileges have been granted by other HADB users
  - Chunks in tables permitted in the PUBLIC specification

  The scope of chunk information in tables displayed by executing #CHUNKS is the same as the scope of information in the STATUS_CHUNKS system table that can be referenced. For details, see *Scope of information in system tables that can be referenced by HADB users* in *System table overview* in the *HADB Setup and Operation Guide*.

- When #CHUNKS is executed, the system table is locked. Execute the COMMIT statement immediately after you execute #CHUNKS. If you do not execute the COMMIT statement, the lock on the system table will not be released. Note that when you terminate the adbsql command after executing #CHUNKS, the lock on the system table is released, so you do not need to execute the COMMIT statement.
- If the specified schema name or table identifier does not exist, the execution result of #CHUNKS contains zero rows (no error is generated).
- The execution results of #CHUNKS are output even if OFF (do not output search results) is specified in #SET DISPLAY. For details about #SET DISPLAY of the adbsql subcommand, see 38.5.12 #SET DISPLAY (outputs the retrieval results).
- The #CHUNKS execution time is not output even if ON (output SQL statement execution time) is specified in #SET EXECTIME REPORT. For details about #SET EXECTIME REPORT of the adbsql subcommand, see 38.5.14 #SET EXECTIME REPORT (outputs SQL statement execution time).
- Even if ON (display index, work table, and access path information) is specified in #SET OPT REPORT, index information, work table information, and access path information are not output when #CHUNKS is executed. For details about #SET OPT REPORT of the adbsql subcommand, see 38.5.16 #SET OPT REPORT (displays index, work table, and access path information).
- The execution results of #CHUNKS are output even if EXEC=PREPARE (do not execute the SQL statement) is specified in #SET OPT REPORT.

38.5.2 #COLUMNS (displays column information)

Function

This subcommand displays column information for the specified table.

Format

```
#COLUMNS [schema-name.]table-identifier [SHOW ALL];
```

[*schema-name*.]*table-identifier*

Specifies the name of a table for which column information is to be displayed, in the format *schema-name*.*table-identifier*. If the schema name is omitted, the authorization identifier of the HADB user who executed this subcommand is assumed.

For details about the specification rules that apply when a schema name or table identifier includes any lowercase letter or backslash (\), see 1.4.3 Table name specification rules.

[SHOW ALL]

If this option is specified

All columns in the SQL_COLUMNS dictionary table are displayed. For details about the displayed information, see *Content of SQL_COLUMNS* in the *HADB Setup and Operation Guide*.

When this option is omitted

Information shown in the following table is displayed.

Table 38-4: Information displayed by #COLUMNS (if the option is omitted)

No.	Column name	Description
1	COLUMN_NAME	Column name
2	DATA_TYPE	Data type
3	DATA_LENGTH	Column definition length For details, see DATA_LENGTH in *Content of SQL_COLUMNS* in the *HADB Setup and Operation Guide*.
4	IS_NULLABLE	Whether the NOT NULL constraint is specified for the column For details, see IS_NULLABLE in *Content of SQL_COLUMNS* in the *HADB Setup and Operation Guide*.
5	IS_DEFAULT_COLUMN	Whether the DEFAULT clause is specified • Y Column for which the DEFAULT clause was specified when a base table was defined • N Column for which the DEFAULT clause is not specified when a base table is defined

Important points

- The scope of column information in tables displayed by executing #COLUMNS varies depending on the privileges HADB users have. The following describes the scope of column information in tables to be displayed.

  When the authorization identifier of an HADB user with the DBA privilege is used to connect to the HADB server

  - Columns in tables defined by all HADB users
  - Columns in tables whose schema name is MASTER

When the authorization identifier of an HADB user without the preceding privilege is used to connect to the HADB server

- Columns in the tables defined by this HADB user (with the authorization identifier that was used for the current connection to the HADB server)
- Columns in tables for which access privileges have been granted by other HADB users
- Columns in tables permitted in the PUBLIC specification
- Columns in tables whose schema name is MASTER

The scope of column information in tables displayed by executing #COLUMNS is the same as the scope of information in the SQL_COLUMNS dictionary table that can be referenced. For details, see *Scope of information in dictionary tables that can be referenced by HADB users* in *Dictionary table overview* in the *HADB Setup and Operation Guide*.

- When #COLUMNS is executed, the dictionary table is locked. Execute the COMMIT statement immediately after you execute #COLUMNS. If you do not execute the COMMIT statement, the lock on the dictionary table will not be released. Note that when you terminate the adbsql command after executing #COLUMNS, the lock on the dictionary table is released, so you do not need to execute the COMMIT statement.
- If the specified schema name or table identifier does not exist, the execution result of #COLUMNS contains zero rows (no error is generated).
- The execution results of #COLUMNS are output even if OFF (do not output search results) is specified in #SET DISPLAY. For details about #SET DISPLAY of the adbsql subcommand, see 38.5.12 #SET DISPLAY (outputs the retrieval results).
- The #COLUMNS execution time is not output even if ON (output SQL statement execution time) is specified in #SET EXECTIME REPORT. For details about #SET EXECTIME REPORT of the adbsql subcommand, see 38.5.14 #SET EXECTIME REPORT (outputs SQL statement execution time).
- Even if ON (display index, work table, and access path information) is specified in #SET OPT REPORT, index information, work table information, and access path information are not output when #COLUMNS is executed. For details about #SET OPT REPORT of the adbsql subcommand, see 38.5.16 #SET OPT REPORT (displays index, work table, and access path information).
- The execution results of #COLUMNS are output even if EXEC=PREPARE (do not execute the SQL statement) is specified in #SET OPT REPORT.

38.5.3 #COMMENT (specifies a comment line)

Function

This subcommand handles the specified character string up to the semicolon as a comment.

Format

```
#COMMENT comment;
```

38.5.4 #CONNECT (connects to the HADB server)

Function

This subcommand connects to the HADB server.

You use the #CONNECT subcommand to re-connect to the HADB server after you have disconnected from the server by executing the #DISCONNECT subcommand.

Format

```
#CONNECT;
```

Important points

- Once you execute #CONNECT, entry of the authorization identifier and password will be requested.
- The password cannot be omitted.
- To connect to the HADB server using a different authorization identifier from the one used for the current connection to the HADB server, disconnect from the HADB server with the #DISCONNECT command and then execute #CONNECT.

38.5.5 #DISCONNECT (disconnects from the HADB server)

Function

This subcommand terminates the executing transaction normally and commits it. The subcommand then disconnects from the HADB server.

Format

```
#DISCONNECT;
```

Important points

Execution of #DISCONNECT disconnects from the HADB server, but the adbsql command does not terminate. To terminate the adbsql command, execute #EXIT. To re-connect to the HADB server, execute #CONNECT.

38.5.6 #EXIT (terminates the adbsql command)

Function

This subcommand terminates the adbsql command.

Format

```
#EXIT;
```

Important points

If #EXIT is executed while the HADB server is connected, the HADB server is disconnected (#DISCONNECT) and then the adbsql command is terminated.

38.5.7 #GETCOUNT (gets the number of data items stored in a chunk)

Function

This subcommand gets the number of data items stored in the specified chunk.

#GETCOUNT can be executed on multi-chunk tables.

Format

```
#GETCOUNT table-name CHUNKID=chunk-ID;
```

table-name

Specifies the table name from which to get the number of data items. For table-name, you can specify only a multi-chunk table for which the HADB user (authorization identifier) connected to the HADB server has the `SELECT` privilege.

chunk-ID

Specifies the chunk ID for which to get the number of data items. Specifies an existing chunk ID. For details about how to find out chunk IDs, see *Searching system tables* in the *HADB Setup and Operation Guide*.

> **Note**
>
> You can use this subcommand to acquire the number of data items stored in the chunk of the specified chunk ID even if that chunk is in wait status.

Important points

- The output format of `#GETCOUNT` is the same as that of the `SELECT` statement for searching a table specified for table name. For details about the output format of the `SELECT` statement, see 38.7.2 Data manipulation SQL statements (SELECT statement).
- `#GETCOUNT` results are output even if `#SET DISPLAY` is set to `OFF` (do not output search results). For details about `#SET DISPLAY` of the `adbsql` subcommand, see 38.5.12 #SET DISPLAY (outputs the retrieval results).
- If `#GETCOUNT` is specified, the `#SET EXECTIME REPORT` and `#SET OPT REPORT` specifications are not enabled (always treated as `OFF`).

38.5.8 #GETDATA (gets data stored in a chunk)

Function

This subcommand acquires data from a specified chunk and outputs it to the standard output.

`#GETDATA` can be executed on multi-chunk tables.

Format

```
#GETDATA table-name CHUNKID=chunk-ID;
```

table-name

Specifies the name of the table from which data is to be acquired. HADB users (authorization identifiers) connected to the HADB server can only specify the name of a multi-chunk table for which they have the `SELECT` privilege.

chunk-ID

Specifies the chunk ID of a chunk from which data is to be acquired. Specify an existing chunk ID. For details about how to check chunk IDs, see *Searching system tables* in the *HADB Setup and Operation Guide*.

> **Note**
>
> You can use this subcommand to output data stored in the chunk of the specified chunk ID even if that chunk is in wait status.

Example

This example outputs data from a chunk (chunk ID is 1) of table T1 to a file (outfile) in CSV format.

```
adbsql -u ADBUSER01 -p '#HelloHADB_01' -s > outfile          ...1
#GETDATA "T1" CHUNKID=1;                                     ...2
#EXIT;                                                       ...3
```

Explanation:

1. Starts the adbsql command with the -u, -p and -s options specified.
2. Executes the #GETDATA subcommand.
   Data in a chunk (chunk ID 1) of table T1 is output to a file (outfile) in CSV format.
3. Terminates the adbsql command.

Notes

- The output format of #GETDATA is the same as that of the SELECT statement for searching a table specified by name. For details about the output format of the SELECT statement, see 38.7.2 Data manipulation SQL statements (SELECT statement).
- The results of #GETDATA are output even when OFF (do not output the retrieval results) is specified in #SET DISPLAY. For details about #SET DISPLAY of the adbsql subcommand, see 38.5.12 #SET DISPLAY (outputs the retrieval results).
- If #GETDATA is specified, the #SET EXECTIME REPORT and #SET OPT REPORT specifications are not enabled (always treated as OFF).

38.5.9 #GETUSER (displays the authorization identifier)

Function

This subcommand displays the authorization identifier of the HADB user who executes this subcommand.

Format

```
#GETUSER;
```

Important points

- The execution results of #GETUSER are output even if OFF (do not output search results) is specified in #SET DISPLAY. For details about #SET DISPLAY of the adbsql subcommand, see 38.5.12 #SET DISPLAY (outputs the retrieval results).
- The #GETUSER execution time is not output even if ON (output SQL statement execution time) is specified in #SET EXECTIME REPORT. For details about #SET EXECTIME REPORT of the adbsql subcommand, see 38.5.14 #SET EXECTIME REPORT (outputs SQL statement execution time).
- Even if ON (display index, work table, and access path information) is specified in #SET OPT REPORT, index information, work table information, or access path information is not output when #GETUSER is executed. For details about #SET OPT REPORT of the adbsql subcommand, see 38.5.16 #SET OPT REPORT (displays index, work table, and access path information).
- The execution results of #GETUSER are output even if EXEC=PREPARE (do not execute the SQL statement) is specified in #SET OPT REPORT.

38.5.10 #INDEXES (displays index information)

Function

This subcommand displays index information for the specified table.

Format

```
#INDEXES [[schema-name.]table-identifier] [SHOW ALL];
```

[[*schema-name*.]*table-identifier*]

Specifies the name of a table for which index information is to be displayed, in the format *schema-name*.*table-identifier*. If the schema name is omitted, the authorization identifier of the HADB user who executed this subcommand is assumed.

For details about the specification rules that apply when a schema name or table identifier includes any lowercase letter or backslash (\), see 1.4.3 Table name specification rules.

If the schema name and table identifier are omitted (a table name is not specified), index information for all tables is displayed. However, the tables whose information will be displayed vary depending on the privileges HADB users have. See Important points.

[SHOW ALL]

If this option is specified

All columns in the SQL_INDEXES dictionary table are displayed. For details about the displayed information, see *Content of SQL_INDEXES* in the *HADB Setup and Operation Guide*.

When this option is omitted

Information shown in the following table is displayed.

Table 38-5: Information displayed by #INDEXES (if the option is omitted)

No.	Column name	Description
1	TABLE_SCHEMA	Schema name
2	TABLE_NAME	Table identifier of the table for which the index is defined
3	INDEX_NAME	Index identifier In the case of an index that corresponds to the primary key, an index identifier having the same name as the constraint name is stored.
4	INDEX_TYPE	Index type • B-TREE: B-tree index • RANGE: Range index • TEXT: Text index
5	COLUMN_COUNT	Number of indexed columns
6	UNIQUE_TYPE	B-tree index type (whether the index is unique) • Y: Unique index • N: Non-unique index

Important points

- The scope of index information displayed by executing #INDEXES varies depending on the privileges HADB users have. The following describes the scope of index information to be displayed.

  When the authorization identifier of an HADB user with the DBA privilege or audit admin privilege is used to connect to the HADB server

  - Indexes in tables defined by all HADB users
  - Indexes in tables whose schema name is MASTER

When the authorization identifier of an HADB user without the preceding privilege is used to connect to the HADB server

- Indexes in the tables defined by this HADB user (with the authorization identifier that was used for the current connection to the HADB server)
- Indexes in tables for which access privileges have been granted by other HADB users
- Indexes in tables permitted in the PUBLIC specification
- Indexes in tables whose schema name is MASTER

The scope of index information displayed by executing #INDEXES is the same as the scope of information in the SQL_INDEXES dictionary table that can be referenced. For details, see *Scope of information in dictionary tables that can be referenced by HADB users* in *Dictionary table overview* in the *HADB Setup and Operation Guide*.

- When #INDEXES is executed, the dictionary table is locked. Execute the COMMIT statement immediately after you execute #INDEXES. If you do not execute the COMMIT statement, the lock on the dictionary table will not be released. Note that when you terminate the adbsql command after executing #INDEXES, the lock on the dictionary table is released, so you do not need to execute the COMMIT statement.
- If the specified schema name or table identifier does not exist, the execution result of #INDEXES contains zero rows (no error is generated).
- The execution results of #INDEXES are output even if OFF (do not output search results) is specified in #SET DISPLAY. For details about #SET DISPLAY of the adbsql subcommand, see 38.5.12 #SET DISPLAY (outputs the retrieval results).
- The #INDEXES execution time is not output even if #SET EXECTIME REPORT is set to ON (output SQL statement execution time). For details about #SET EXECTIME REPORT of the adbsql subcommand, see 38.5.14 #SET EXECTIME REPORT (outputs SQL statement execution time).
- Even if #SET OPT REPORT is set to ON (display index, work table, and access path information), index information, work table information, and access path information are not output when #INDEXES is executed. For details about #SET OPT REPORT of the adbsql subcommand, see 38.5.16 #SET OPT REPORT (displays index, work table, and access path information).
- The execution results of #INDEXES are output even if EXEC=PREPARE (do not execute the SQL statement) is specified in #SET OPT REPORT.

38.5.11 #SET BINARY_DISP_MODE (specifies the binary data output format)

Function

This subcommand specifies whether binary data obtained as retrieval results is to be displayed in hexadecimal or binary format.

If the -b option was not specified when the adbsql command was started, the initial value is HEX (binary data is displayed in hexadecimal format). If the -b option was specified when the adbsql command was started, the initial value is BIN (binary data is displayed in binary format).

Format

```
#SET BINARY_DISP_MODE {HEX|BIN};
```

HEX: Displays binary data in hexadecimal format.

BIN: Displays binary data in binary format.

Notes

When ON (displays retrieval results in hexadecimal) is specified in #SET DUMPMODE, specification of BIN in this subcommand will be ignored.

38.5.12 #SET DISPLAY (outputs the retrieval results)

Function

Specifies whether to output the retrieval results.

The initial value is ON (output the retrieval results).

Format

```
#SET DISPLAY {ON|OFF};
```

ON: Outputs the retrieval results.

OFF: Does not output the retrieval results.

Example

This example displays SQL statement execution time, excluding the time required for processing and outputting the retrieval results.

```
adbsql -u ADBUSER01 -p '#HelloHADB_01'                       ...1
#SET EXECTIME REPORT ON;                                     ...2
#SET DISPLAY OFF;                                            ...3
SELECT * FROM "T1";                                          ...4
KFAA96404-I 100000000 rows were selected.
KFAA96415-I SQL processing time (Fetch) : 18.863501 sec.     ...5
#EXIT;                                                       ...6
```

Explanation:

1. Starts the adbsql command with the -u and -p options specified.
2. Specifies ON (output SQL statement execution time) in #SET EXECTIME REPORT.
3. Specifies OFF in #SET DISPLAY (do not output the retrieval results).
4. Executes the SQL statement.
5. Displays the SQL statement execution time, excluding the time required for processing and outputting the retrieval results.
6. Terminates the adbsql command.

Notes

If you specify the -s option when starting the adbsql command, you cannot execute #SET DISPLAY.

38.5.13 #SET DUMPMODE (displays the search results in hexadecimal)

Function

This subcommand specifies whether the search results are to be displayed in hexadecimal or character format.

The initial value is OFF (search results are displayed in character format).

Format

```
#SET DUMPMODE {ON|OFF};
```

`ON`: Display the search results in hexadecimal.

`OFF`: Display the search results in character format.

38.5.14 #SET EXECTIME REPORT (outputs SQL statement execution time)

Function

Specifies whether to output the execution time of the following SQL statements:

- Definition SQL statements
- Data manipulation SQL statements
- Control SQL statements

The following times are output for SQL statement execution time.

- For definition SQL statements

  The time between the point preceding issuance of `a_rdb_SQLAllocStmt()` and completion of `a_rdb_SQLFreeStmt()`

- For the `SELECT` statement

  The time from fetching the first row (the first `a_rdb_SQLFetch()`) to fetching the last row (the last `a_rdb_SQLFetch()` that is issued)

- For data manipulation SQL statements, excluding the `SELECT` statement

  The time between the point preceding issuance of `a_rdb_SQLExecute()` and completion of `a_rdb_SQLExecute()`

- For control SQL statements

  The time between the point preceding issuance of `a_rdb_SQLEndTran()` and completion of `a_rdb_SQLEndTran()`

The initial value is `OFF` (do not output SQL statement execution time).

Format

```
#SET EXECTIME REPORT {ON|OFF};
```

`ON`: Outputs SQL statement execution time.

`OFF`: Does not output SQL statement execution time.

The following shows the output format for SQL statement execution time when `ON` is specified:

```
KFAA96415-I SQL processing time (aa....aa) : ssssssssssss.pppppp sec.
```

Legend:

aa....aa: Displays the action that was executed. If row fetching was performed, `Fetch` is displayed. If an SQL statement was executed, `Execute` is displayed.

ssssssssssss: Displays a number of seconds. The effective digits are displayed, left-justified.

pppppp: Displays fractional seconds.

Notes

If an error occurs, SQL statement execution time is not output.

38.5.15 #SET NULL (specifies a null value display format)

(1) Function

Specifies the character string to be displayed when the retrieval result is a null value.

The default value of the character string to be displayed when the retrieval result is a null value is the value specified in the `ADBSQLNULLCHAR` environment variable. If the `ADBSQLNULLCHAR` environment variable is not specified, the default value is an asterisk (`*`).

(2) Specification format

```
#SET NULL {[character-string-to-be-displayed-for-null-value]
            |REPEAT fill-character-to-be-displayed-for-null-value
            |DELETE};

  character-string-to-be-displayed-for-null-value::=character-string-lateral
  fill-character-to-be-displayed-for-null-value::=character-string-lateral
```

(3) Explanation of options

character-string-to-be-displayed-for-null-value:

Specifies the character string to be displayed when the retrieval result is a null value.

The following rules apply when specifying this option:

- Specify a character string enclosed in single quotation marks (`'`).

  Example: `'NULL'`
- A character string (0 to 32 bytes) can be specified.
- If this option is omitted or if a 0-byte character string is specified for it, spaces are displayed when the retrieval result is a null value.
- If a specified character string includes one or more single quotation marks (`'`), use two single quotation marks ('') to express each single quotation mark ('). For example, if you want the character string `N'ULL` displayed for a null value, specify as follows:

  `'N''ULL'`

`REPEAT` *fill-character-to-be-displayed-for-null-value*:

Specifies the character to be repeated (fill character) to be displayed when the retrieval result is a null value. A character string with the specified fill character, repeated up to the maximum width of the retrieval results that can be displayed, is displayed when the retrieval result is a null value.

The following rules apply when specifying this option:

- Specify a fill character enclosed in single quotation marks (`'`).

  Example: `':'`
- Set the length of the fill character to be specified to 1 byte.
- To use a single quotation mark (`'`) as a fill character, specify as follows.

```
    ''''
```

`DELETE`

Returns the character string to be displayed to its default value when the retrieval result is a null value.

(4) Execution examples

Displaying the default value for a null value

```
SELECT CAST(NULL AS CHAR(4)) FROM "T1";     ...1

 EXP0001_NO_NAME
 ---------------
            ****                            ...2
```

Explanation:

1. Executes an SQL statement without executing `#SET NULL`.
2. If the retrieval result is a null value, the value specified in the `ADBSQLNULLCHAR` environment variable is displayed. If the `ADBSQLNULLCHAR` environment variable is not specified, asterisks (`*`) are displayed.

Displaying a character string for a null value

```
#SET NULL 'NULL';                            ...1
SELECT CAST(NULL AS CHAR(4)) FROM "T1";      ...2

 EXP0001_NO_NAME
 ---------------
            NULL                             ...3
```

Explanation:

1. Specifies `NULL` for the character string to be displayed when the retrieval result is a null value, and executes `#SET NULL`.
2. Executes an SQL statement.
3. If the retrieval result is a null value, `NULL` is displayed.

Displaying spaces for a null value

```
#SET NULL '';                               ...1
SELECT CAST(NULL AS CHAR(4)) FROM "T1";     ...2

 EXP0001_NO_NAME
 ---------------
                                            ...3
```

Explanation:

1. Specifies a 0-byte character string for the character string to be displayed when the retrieval result is a null value, and executes `#SET NULL`.
2. Executes an SQL statement.
3. If the retrieval result is a null value, spaces are displayed.

Displaying colons (:) for a null value (with the fill character specified)

```
#SET NULL REPEAT ':';                       ...1
SELECT CAST(NULL AS CHAR(4)) FROM "T1";     ...2

 EXP0001_NO_NAME
 ---------------
            ::::                            ...3
```

Explanation:

1. Specifies the colon (`:`) for the fill character to be displayed when the retrieval result is a null value, and executes `#SET NULL`.
2. Executes an SQL statement.
3. When the retrieval result is a null value, the colon (`:`) is repeatedly displayed up to the maximum width of the retrieval results that can be displayed.

(5) Notes

- If multi-byte characters are specified in the character string to be displayed when the retrieval result is a null value, the character string might be displayed correctly.
- If the character string to be displayed when the retrieval result is a null value exceeds the width that can be displayed for retrieval results, only the portion of the retrieval result that can be displayed out of the specified character string is displayed. If multi-byte characters are specified in the character string to be displayed when the retrieval result is a null value, and if the end of the width that can be displayed in the retrieval result falls in the middle of a multi-byte character, only the part of the character string up to the preceding character is displayed.

38.5.16 #SET OPT REPORT (displays index, work table, and access path information)

(1) Function

This subcommand specifies whether index, work table, and access path information for any of the following SQL statements is to be displayed when the SQL statement is to execute:

- `DELETE` statement
- `INSERT` statement

  In the following cases, index information, work table information, and access path information are displayed:

  - A query expression body is specified in the `INSERT` statement.
  - `VALUES` is specified in the `INSERT` statement, and a subquery is also specified
- `PURGE CHUNK` statement

  If a subquery is specified in the `PURGE CHUNK` statement, index information, work table information, and access path information are displayed.
- `SELECT` statement
- `UPDATE` statement

If the executed SQL statement does not create a work table, no work table information is displayed.

If an index specification is specified for a table reference, whether that index specification was applied is displayed as the index information. For details about the index specification, see *Specification format and rules for index specifications* in the manual *HADB SQL Reference*.

For details about the access path information, see *How to use access paths (how to use SQL statement execution plans)* in *Tuning Application Programs* in the *HADB Application Development Guide*.

The initial value is `OFF` (index, work table, and access path information is not displayed).

(2) Specification format

```
#SET OPT REPORT {ON
                   [TYPE = {INDEX|PATH|ALL}]
                   [EXEC = {NORMAL|PREPARE}]
                |OFF};
```

(3) Explanation of options

`ON`:

Displays index, work table, and access path information.

`[TYPE = {INDEX|PATH|ALL}]`:

Specifies the type of information that is to be displayed:

`INDEX`: Display index and work table information.

`PATH`: Display only access path information.

`ALL`: Display index, work table, and access path information.

`[EXEC = {NORMAL|PREPARE}]`:

Specifies whether the SQL statement is to be executed after the information has been displayed:

`NORMAL`: Execute the SQL statement after the information has been displayed.

`PREPARE`: For the following SQL statements, display the information, but do not execute the SQL statement:

- `DELETE` statement
- `INSERT` statement
- `PURGE CHUNK` statement
- `SELECT` statement
- `UPDATE` statement

Any other SQL statement will be executed, even though `EXEC = PREPARE` is specified.

`OFF`:

Does not display index, work table, or access path information.

(4) Examples

Executing an SQL statement after displaying index and work table information

```
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
#SET OPT REPORT ON;                                          ...1
KFAA96499-I OPT REPORT changed  ( OFF -> ON )
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
SELECT * FROM "T1" WHERE "C1"=100 ORDER BY "C2";             ...2
```

```
KFAA96405-I The B-tree index "INDX01" will be used.
 (table name="ADBUSER01"."T1" as "***")                      ...2
KFAA96406-I Executing this SQL will create 1 work tables.   ...3
 retrieval-results                                           ...4
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
```

Explanation:

1. Executes `#SET OPT REPORT` specifying that index and work table information is to be displayed.
2. Executes the SQL statement.
3. Displays the index and work table information as messages. The displayed messages indicate the following:
   - B-tree index `INDX01` is used when this SQL statement executes.
   - One work table is created when this SQL statement executes.
4. Displays the execution results of the SQL statement.

Executing an SQL statement after displaying all information

```
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
#SET OPT REPORT ON TYPE = ALL EXEC = NORMAL;                 ...1
KFAA96499-I OPT REPORT changed  ( OFF -> ON )
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
SELECT * FROM "T1" WHERE "C1"=100 ORDER BY "C2";             ...2
KFAA96405-I The B-tree index "INDX01" will be used.
 (table name="ADBUSER01"."T1" as "***")                      ...2
KFAA96406-I Executing this SQL will create 1 work tables.   ...3
 access-path-information                                     ...4
KFAA96412-I The access path was output.                      ...5
 retrieval-results                                           ...6
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
```

Explanation:

1. Executes `#SET OPT REPORT` specifying that index, work table, and access path information is to be displayed.
2. Executes the SQL statement.
3. Displays the index and work table information as messages. The displayed messages indicate the following:
   - B-tree index `INDX01` is used when this SQL statement executes.
   - One work table is created when this SQL statement executes.
4. Displays the access path information. For details about how to interpret the displayed access path information, see *Examples of access paths* in *How to use access paths (how to use SQL statement execution plans)* in *Tuning Application Programs* in the *HADB Application Development Guide*.
5. Displays a message indicating that the access path information has been displayed.
6. Displays the execution results of the SQL statement.

Displaying only access path information without executing an SQL statement

```
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
#SET OPT REPORT ON TYPE = PATH EXEC = PREPARE;               ...1
KFAA96499-I OPT REPORT changed  ( OFF -> ON )
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
SELECT * FROM "T1" WHERE "C1"=100 ORDER BY "C2";             ...2
 access-path-information                                     ...3
KFAA96412-I The access path was output.                      ...4
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
```

Explanation:

1. Executes `#SET OPT REPORT` specifying that access path information is to be displayed.
2. Executes the SQL statement.
3. Displays the access path information. For details about how to interpret the displayed access path information, see *Examples of access paths* in *How to use access paths (how to use SQL statement execution plans)* in *Tuning Application Programs* in the *HADB Application Development Guide*.
4. Displays a message indicating that the access path information has been displayed.

(5) Messages displayed as index and work table information

This subsection explains the messages that are displayed as index and work table information.

Messages that are output as index information

The following table lists and describes the messages that are output as index information. If no index is used, no messages are output.

Table 38-6: List of messages that are output as index information

No.	Message that is output	Contents of index information
1	`KFAA96405-I`	• If the index specification is not specified B-tree indexes are used for retrieval. • If the index specification (`WITH INDEX`) is specified Because the index specification is invalid,# B-tree indexes that are not specified are used for retrieval.
2	`KFAA96407-I`	No indexes (B-tree indexes, text indexes, or range indexes) are used for retrieval.
3	`KFAA96408-I`	Range indexes are used for retrieval.
4	`KFAA96409-I`	The index specification (`WITH INDEX`) is applied and the specified B-tree indexes are used for retrieval.
5	`KFAA96410-I`	The index specification (`WITHOUT INDEX`) is applied, and use of the specified B-tree or text indexes is suppressed.
6	`KFAA96411-I`	The index specification is invalid.
7	`KFAA96413-I`	• If the index specification is not specified: Text indexes are used for retrieval. • If the index specification (`WITH INDEX`) is specified: Because the index specification is invalid,# text indexes that are not specified are used for retrieval.
8	`KFAA96414-I`	The index specification (`WITH INDEX`) is applied, and the specified text indexes are used for retrieval.

#

The index specification is ignored in the following cases:

- A nonexistent index name was specified.
- A B-tree index with the null-value exclusion specification is specified, but the specified condition contains a null value within the index retrieval range.
- HADB cannot effectively use text indexes, such as when text indexes are specified but no `LIKE` predicate that can be evaluated in the specified text indexes is specified.

Message that is output as work table information

If a work table is created, the KFAA96406-I message is output as work table information. If no work table is created, this message is not output.

(6) Notes

The same index information might be displayed multiple times for a table that was specified in FULL OUTER JOIN for a joined table. For details about the index information, check the access path information.

38.5.17 #SET TRAN_ISOLV (specifies the transaction isolation level)

Function

This subcommand specifies the transaction isolation level.

When #SET TRAN_ISOLV is executed, the initial value is the value of the adb_clt_trn_iso_lv operand in the client definition.

Format

```
#SET TRAN_ISOLV{READ_COMMITTED|REPEATABLE_READ};
```

READ_COMMITTED: Sets the transaction isolation level to READ COMMITTED.

REPEATABLE_READ: Sets the transaction isolation level to REPEATABLE READ.

For details about the transaction isolation levels, see *Transaction isolation levels supported by HADB* in the *HADB Setup and Operation Guide*.

38.5.18 #TABLES (displays table information)

Function

This subcommand displays table information.

Format

```
#TABLES [schema-name] [SHOW ALL];
```

[*schema-name*]

Specifies a schema name for which table information is to be displayed.

If the schema name is omitted, table information for all tables is displayed. However, the tables whose information will be displayed vary depending on the privileges HADB users have. See Important points.

[SHOW ALL]

If this option is specified

All columns in the SQL_TABLES dictionary table are displayed. For details about the displayed information, see *Content of SQL_TABLES* in the *HADB Setup and Operation Guide*.

When this option is omitted

Information shown in the following table is displayed.

Table 38-7: Information displayed by #TABLES (if the option is omitted)

No.	Column name	Description
1	TABLE_SCHEMA	Schema name
2	TABLE_NAME	Table identifier
3	STORAGE_FORMAT	Table-data storage format • 'COLUMN' Column store format • 'ROW' Row store format The null value is displayed for a viewed table.
4	N_COLS	Number of columns in the table
5	N_INDEX	Number of indexes defined for the table (sum of all indexes)
6	IS_CHUNK	Specification of CHUNK chunk specification • Y: Specified (multi-chunk table) • N: Not specified (single chunk table)
7	N_CHUNK	Number of existing chunks If N is output for column name IS_CHUNK, a null value is displayed.
8	N_CHUNK_RESERVED	Maximum number of chunks to be created
9	CREATE_TIME	Time stamp when the table was defined

Important points

- The scope of table information displayed by executing #TABLES varies depending on the privileges HADB users have. The following describes the scope of table information to be displayed.

  When the authorization identifier of an HADB user with the DBA privilege or audit admin privilege is used to connect to the HADB server

  - Tables defined by all HADB users
  - Tables whose schema name is MASTER

  When the authorization identifier of an HADB user without the preceding privilege is used to connect to the HADB server

  - Tables defined by this HADB user (with the authorization identifier that was used for the current connection to the HADB server)
  - Tables for which access privileges have been granted by other HADB users
  - Tables permitted in the PUBLIC specification
  - Tables whose schema name is MASTER

  The scope of table information displayed by executing #TABLES is the same as the scope of information in the SQL_TABLES dictionary table that can be referenced. For details, see *Scope of information in dictionary tables that can be referenced by HADB users* in *Dictionary table overview* in the *HADB Setup and Operation Guide*.

- When #TABLES is executed, the dictionary table is locked. Execute the COMMIT statement immediately after you execute #TABLES. If you do not execute the COMMIT statement, the lock on the dictionary table will not be released. Note that when you terminate the adbsql command after executing #TABLES, the lock on the dictionary table is released, so you do not need to execute the COMMIT statement.

- If the specified schema name does not exist, the execution result of #TABLES contains zero rows (no error is generated).

- The execution results of #TABLES are output even if OFF (do not output search results) is specified in #SET DISPLAY. For details about #SET DISPLAY of the adbsql subcommand, see 38.5.12 #SET DISPLAY (outputs the retrieval results).
- The #TABLES execution time is not output even if ON (output SQL statement execution time) is specified in #SET EXECTIME REPORT. For details about #SET EXECTIME REPORT of the adbsql subcommand, see 38.5.14 #SET EXECTIME REPORT (outputs SQL statement execution time).
- Even if ON (display index, work table, and access path information) is specified in #SET OPT REPORT, index information, work table information, or access path information is not output when #TABLES is executed. For details about #SET OPT REPORT of the adbsql subcommand, see 38.5.16 #SET OPT REPORT (displays index, work table, and access path information).
- The execution results of #TABLES are output even if EXEC=PREPARE (do not execute the SQL statement) is specified in #SET OPT REPORT.

38.6 Operating method when dynamic parameters are specified in an SQL statement

This section explains by way of an example the operating method when dynamic parameters are specified in an SQL statement.

38.6.1 Example

This example executes the INSERT statement which uses dynamic parameters to add a row.

(1) Adding rows with the VALUES specification

```
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
INSERT INTO T1 VALUES(?,?,?);                     ...1
 DATA (    1) ? : SMALLINT                        ...2
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
25000;                                            ...3
 DATA (    2) ? : VARCHAR (   10)                 ...2
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
'HITACHI';                                        ...3
 DATA (    3) ? : TIMESTAMP (    7)               ...2
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
TIMESTAMP'2011-02-23 12:00:00';                   ...3
KFAA96404-I 1 rows were inserted. COMMAND ?     +----2----+----3----+----4----+----5-
---+----6----+----7----+
COMMIT;                                           ...4
KFAA96403-I SQL processing completed.
```

Explanation:

1. Executes the INSERT statement using dynamic parameters.
2. Displays a request for entry of data for a dynamic parameter in the following format:
   DATA (*aa...aa*) ?:*bb...bb*
   aa...aa: Sequence number of the dynamic parameter from the beginning of the SQL statement.
   bb...bb: Data type assumed by the dynamic parameter. One of the following is displayed:
   - INTEGER
   - SMALLINT
   - DECIMAL (*precision*, *scaling*)
   - DOUBLE PRECISION
   - CHAR (*length*)
   - VARCHAR (*length*)
   - DATE (*length*)
   - TIME (*length*)
   - TIMESTAMP (*length*)
   - BINARY (*length*)
   - VARBINARY (*length*)
3. Specifies a literal, which must be consistent with the data type assumed by the dynamic parameter. For the rules for input data, see 38.6.2 Input data specification rules.

4. Executes COMMIT to finalize addition of the row by the INSERT statement.

(2) Adding data by row (ROW specification)

```
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
INSERT INTO "T1" (ROW) VALUES(?);                 ...1
 DATA (    1) ? : ROW (   10)                     ...2
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
X'0A00000000000000003132';                        ...3
KFAA96404-I 1 rows were inserted. COMMAND ?     +----2----+----3----+----4----+----5-
---+----6----+----7----+
COMMIT;                                           ...4
KFAA96403-I SQL processing completed.
```

Explanation:

1. Executes the INSERT statement with a dynamic parameter specified.
2. Displays a message in the following format requesting entry of input data for a dynamic parameter:

   DATA (*aa...aa*) ?:ROW (*length*)

   aa...aa: Sequence number of the dynamic parameter specified in the SQL statement.
3. Entry of a literal using the correct data type that is assumed by the dynamic parameter. For the rules for input data, see 38.6.2 Input data specification rules.
4. Executes COMMIT and finalizes addition of the row by the INSERT statement.

38.6.2 Input data specification rules

1. Specify literals for the input data. For details about the description format of literals, see *Description format of literals* in the manual *HADB SQL Reference*.
2. An input data entry must end with a semicolon (;).
3. To assign the null value, specify NULL.
4. The maximum length of an input data entry is 64,000 bytes.
5. The type of literal that can be specified for an input data entry is determined by the data type assumed by the dynamic parameter. The following table shows the combinations that can be specified.

Table 38-8: Combinations of the data type assumed by the dynamic parameter and the type of literal that can be specified for an input data entry

Data type assumed by the dynamic parameter		Type of literal					
		Numeric literal	Character string literal	Date literal	Time literal	Time stamp literal	Binary literal[#1]
Numeric data		Y	N	N	N	N	Y
Character string data		N	Y	N	N	N	Y
Datetime data	DATE	N	Y[#2]	Y	N	Y	Y
	TIME			N	Y	N	Y
	TIMESTAMP			Y	N	Y	Y
Binary data	BINARY	N	N	N	N	Y	Y

Data type assumed by the dynamic parameter		Type of literal					
		Numeric literal	Character string literal	Date literal	Time literal	Time stamp literal	Binary literal#1
	VARBINARY	N	N	N	N	Y	Y
ROW		N	N	N	N	N	Y

Legend:

Y: Can be specified.

N: Cannot be specified.

#1

Either of the following can be specified for a binary literal:

- Hexadecimal character string literal
- Binary character string literal

The specification format for a hexadecimal character string literal is X'*hexadecimal-character-string*', and the specification format for a binary character string literal is B'*binary-character-string*'. When specifying the format, keep in mind the endian of the OS. The following shows specification examples:

Example of a hexadecimal character string literal: X'A000000000000000';

Example of a binary character string literal: B'01011100';

#2

A character string literal that is specified in a predefined input representation can be specified. For details about the predefined input representations, see the topic *Predefined character-string representations* in the manual *HADB SQL Reference*.

38.6.3 Important points

- If the data length of an input data entry is greater than length set for the dynamic parameter, an error results. If the fractional seconds precision for the input data is longer than that for the dynamic parameter, the input data will be truncated.
- If the data length of an input data entry is less than the length set for the dynamic parameter, the data is initialized to the value shown in the following table and then the data is assigned.

Table 38-9: Data type of input data and the value after initialization

No.	Data type of input data	Value after initialization
1	CHAR	Space (X'20')
2	INTEGER	0
3	SMALLINT	0
4	DECIMAL	0
5	DOUBLE PRECISION	0
6	DATE	0
7	TIME	0
8	TIMESTAMP	0
9	BINARY	0

No.	Data type of input data	Value after initialization
10	ROW	0

38.7 Output format of SQL statement execution results

This section explains the output format of SQL statement execution results.

38.7.1 Definition SQL statements

If a definition SQL statement is executed, a completion message (KFAA96403-I) is displayed in the following format:

```
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
CREATE INDEX "INDX01" ON "T2"("C1") IN DBAREA01 EMPTY;
KFAA96403-I SQL processing completed.
KFAA96415-I SQL processing time (Execute) : 0.853400 sec.
```

The KFAA96415-I message is issued when ON (output SQL statement execution time) is specified in #SET EXECTIME REPORT.

38.7.2 Data manipulation SQL statements (SELECT statement)

If the SELECT statement is executed, the search results are displayed in the following format:

```
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
SELECT 'C1 = ',"C1" FROM "T1";
 EXP0001_NO_NAME C1                                      ...1
 --------------- ----------------                        ...2
           C1 =   -123456789.1234                        ...3
           C1 =  ****************                        ...4
KFAA96404-I 2 rows were selected.                        ...5
KFAA96415-I SQL processing time (Fetch) : 0.477001 sec.  ...6
```

Explanation:

1. Displays the result of the query expression and the column names for the derived table. For details about the column names in derived tables (retrieval item column names), see *Rules* in *Specification format and rules for the SELECT statement* in *SELECT (retrieve rows)* in *Data Manipulation SQL* in the manual *HADB SQL Reference*.
2. Multiple hyphens (-) are successively displayed for each selection expression. If ON (display the retrieval results in hexadecimal format) is specified in #SET DUMPMODE, equal signs (=) are successively displayed.

   The number of hyphens (-) or equal signs (=) indicates the width that can be displayed in the retrieval results. Hyphens (-) or equal signs (=) are displayed such that one of the following lengths, whichever is longer, is reached:

   - Maximum length when column data is expressed in character format
   - Length of the retrieval item column name
3. Displays the search results. If a data conversion scalar function is specified in the selection expression, the search results might not be displayed correctly due to truncation of trailing bytes of multi-byte characters. Note also that if the CHR scalar function's target data is in the range from 127 to 159, the search results might not be displayed correctly. For details about the data conversion functions, see *Data conversion functions* in *Scalar Functions* in the manual *HADB SQL Reference*.

4. When the retrieval result is a null value, the character string to be displayed for a null value, specified in `#SET NULL`, is displayed. However, if the character string to be displayed for a null value exceeds the maximum data length for retrieval results, the character string is truncated in the middle.

   The maximum length of the character string to be displayed for a null value is the same as the length of the hyphens (-) or equal signs (=) that are displayed in explanation 2. above.

5. Displays the number of rows in the search results. If the number of rows exceeds the maximum value, three asterisks (`***`) are displayed.

6. If `ON` (output SQL statement execution time) is specified in `#SET EXECTIME REPORT`, SQL statement execution time is output.

The following table describes the search results display formats.

Table 38-10: Search results display formats

No.	Data type of search results	Search results display format[#1]
1	`INTEGER`	Right-aligned, shortest literal format for integer data. For positive numbers, a plus sign (+) is not displayed. For negative numbers, a minus sign (-) is displayed immediately before the valid data.
2	`SMALLINT`	
3	`DECIMAL`[#3]	For the fraction portion, the scaling of the results of the derived column is displayed right-aligned. If the scaling of the results of the derived column is 0 and the `-d` option is specified, the decimal point is not output. The integer portion is displayed in the shortest format. For positive numbers, a plus sign (+) is not displayed. For negative numbers, a minus sign (-) is displayed immediately before the valid data.
4	`DOUBLE PRECISION`	The exponent portion is displayed in the shortest format. The mantissa portion is always displayed as 17 digits. The entire data is displayed right-aligned. For positive numbers, a plus sign (+) is not displayed. For negative numbers, a minus sign (-) is displayed immediately before the valid data.
5	`CHARACTER`[#5]	Literal format of each data type without the single quotation marks (`'`). A colon (`:`) is displayed for a character that cannot be displayed ((`X'00'` to `X'1F'`).
6	`VARCHAR`[#6]	
7	`DATE`[#5]	Predefined output representation for date data. For details about the predefined output representation for date data, see *Predefined character-string representations* in the manual *HADB SQL Reference*.
8	`TIME`[#5]	Predefined output representation for time data. For details about the predefined output representation for time data, see *Predefined character-string representations* in the manual *HADB SQL Reference*.
9	`TIMESTAMP`[#5]	Predefined output representation for time stamp data. For details about the predefined output representation for time stamp data, see *Predefined character-string representations* in the manual *HADB SQL Reference*.
10	`BINARY`[#4, #5]	If `BINARY` is specified in a selection expression, the value is displayed in either of the following formats: • Hexadecimal format • Binary format
11	`VARBINARY`[#4, #6]	If `VARBINARY` is specified in a selection expression, the value is displayed in either of the following formats: • Hexadecimal format • Binary format
12	`ROW`[#2, #5]	Hexadecimal format if `ROW` is specified in the selection expression

#1

If the data cannot be converted to character format, as many question marks (?) as matches the maximum length of the data in the character format are displayed.

#2

If ROW is specified in the selection expression, this is the data type.

#3

For the DECIMAL type, if the precision and scaling are the same, leading zeros are added. However, if the -s option was specified and the precision and scaling are both 38, no zeros are added.

#4

The display format to be used is determined by the -b option of the adbsql command or specification of #SET BINARY_DISP_MODE. If ON is specified in #SET DUMPMODE (displays retrieval results in hexadecimal format), the hexadecimal format is used regardless of the specification of #SET BINARY_DISP_MODE.

#5

If the maximum length of data that is represented in string format is shorter than the maximum column width, the data displayed in the column is right-aligned. In other cases, the displayed data is left-aligned.

#6

If the maximum length of data that is represented in string format is shorter than the maximum column width, the difference between the maximum string data length and column width is filled with spaces to the left. In other cases, the displayed data is left-aligned.

38.7.3 Data manipulation SQL statements (DELETE statement, INSERT statement, and UPDATE statement)

If the DELETE, INSERT, or UPDATE statement is executed, the number of manipulated rows is displayed in the format shown below. If the number of manipulated rows results in an overflow, three asterisks (***) are displayed.

```
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
DELETE FROM "T1" WHERE "C1"=0;
KFAA96404-I            1 rows were deleted.
KFAA96415-I SQL processing time (Execute) : 40.011020 sec.

COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
INSERT INTO "T1"("C1","C2") VALUES(+1520.34,'abc');
KFAA96404-I            1 rows were inserted.
KFAA96415-I SQL processing time (Execute) : 0.501823 sec.

COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
UPDATE "T1" SET "C1"=0 WHERE "C1"<0;
KFAA96404-I            2 rows were updated.
KFAA96415-I SQL processing time (Execute) : 120.991024 sec.
```

The KFAA96415-I message is issued when ON (output SQL statement execution time) is specified in #SET EXECTIME REPORT.

38.7.4 Data manipulation SQL statements (PURGE CHUNK and TRUNCATE TABLE statements)

When the `PURGE CHUNK` or `TRUNCATE TABLE` statement is executed, a completion message is displayed in the following format after execution of the SQL statement:

```
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
PURGE CHUNK "T1" WHERE CHUNKID=1;
KFAA96403-I SQL processing completed.
KFAA96415-I SQL processing time (Execute) : 0.901023 sec.

COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
TRUNCATE TABLE "T1";
KFAA96403-I SQL processing completed.
KFAA96415-I SQL processing time (Execute) : 0.000865 sec.
```

The `KFAA96415-I` message is issued when `ON` (output SQL statement execution time) is specified in `#SET EXECTIME REPORT`.

38.7.5 Control SQL statements

If a control SQL statement or the `adbsql` command's `#DISCONNECT` subcommand is executed, a completion message is displayed in the following format:

```
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+
COMMIT;
KFAA96403-I SQL processing completed.
KFAA96415-I SQL processing time (Execute) : 0.000034 sec.
```

The `KFAA96415-I` message is issued when `ON` (output SQL statement execution time) is specified in `#SET EXECTIME REPORT`.

38.8 Notes

1. Codes X'00' to X'1F' are converted to spaces, if specified.

2. You must specify a semicolon (;) at the end of each SQL statement and adbsql subcommand. Any text following a semicolon is treated as a comment. If there is no semicolon, the line is regarded as being continued (the SQL statement or adbsql subcommand continues). The following shows an example.

   Example

   ```
   CREATE TABLE "T1" ("C1" INT,      ← Continued line
   "C2" CHAR(10),                    ← Continued line
   "C3" CHAR(10)) IN DBAREA01;       ← End of SQL statement
   ```

   When multiple lines are to be concatenated, the line break code is replaced with the space, and then the lines are concatenated.

3. A string of characters in an SQL statement between /* and the first */ that follows the /* is treated as a comment. For details about comments, see *Rules for separators* in the manual *HADB SQL Reference*.

   A semicolon (;) between /* and */ is also treated as part of the comment.

   The following shows an example of a comment specification.

   Example

   ```
   /* COMMENT */ SELECT * FROM "T1";
   ```

4. If a memory shortage error is returned from the HADB server (error message in the range from KFAA30000 to KFAA39999 is issued), evaluate the maximum size of the memory. If the KFAA96457-E message is issued, re-execute the adbsql command after some other process has terminated.

39 adbstart (Start the HADB Server)

This chapter explains the functionality provided by the `adbstart` command and how to use the command.

39.1 Description

The adbstart command starts the HADB server. If the multi-node function is used, executing the adbstart command on a node that has been disconnected from the multi-node configuration returns that node to the running multi-node configuration.

The HADB server's startup mode is as follows:

- Normal start if the HADB server's previous termination mode was normal termination
- Restart if the HADB server's previous termination mode was forced termination or abnormal termination. When the HADB server is restarted, recovery processing is performed on any transaction whose processing was underway at the time of the termination.

This command can also be used to specify the HADB server operation mode as an option. The four HADB server operation modes are as follows:

- Normal mode
- Quiescence mode
- Offline mode
- Maintenance mode

For details about the HADB server operation modes, see *HADB server operation modes* in *Starting and terminating the HADB server and its operation modes* in *Scheduled Operations* in the *HADB Setup and Operation Guide*.

The SQL statements and commands that can be executed depend on the HADB server operation mode. For details, see 39.3 SQL statements and commands that can be executed in each HADB server operation mode.

Multi-node function

- If you are using the multi-node function, you must execute the adbstart command on each node.
- For details about returning a node to the multi-node configuration, see *Returning a node to the multi-node configuration* in *Operations when a node failure occurs* in *Operations When Using the Multi-Node Function* in the *HADB Setup and Operation Guide*.

Centralized management of client definitions

If the client management definition file ($ADBDIR/conf/adbclientdefmang.def) exists, you can execute the adbstart command to apply the function for centrally managing client definitions.

39.2 Specification format for the adbstart command

This section explains the specification format and options for the `adbstart` command.

The only OS users who can execute the `adbstart` command are HADB administrators.

39.2.1 Explanation of the specification format and options

(1) Specification format

```
adbstart [{--normal|--quiescence|--offline|--maintenance}]
```

(2) Explanation of options

Specifies the HADB server operation mode.

If the HADB server is started with no option specified, the previous HADB server operation mode in effect when the HADB server last terminated (normal, quiescence, offline, or maintenance) is inherited. For example, if the HADB server operation mode was quiescence the last time the HADB server terminated, the HADB server operation mode will also be quiescence the next time the HADB server is started. Note that if the HADB server is started for the first time with this option omitted, normal mode is assumed to be the HADB server operation mode.

When the multi-node function is being used, the specified HADB server operation mode takes effect when the HADB server on the master node is started. However, whenever the HADB server on a slave node is started, that HADB server is started in the quiescence mode and the specified HADB server operation mode is ignored.

- `--normal`

  The HADB server is to be started with the HADB server operation mode in the normal mode.

- `--quiescence`

  The HADB server is to be started with the HADB server operation mode in the quiescence mode.

- `--offline`

  The HADB server is to be started with the HADB server operation mode in the offline mode.

- `--maintenance`

  The HADB server is to be started with the HADB server operation mode in the maintenance mode.

39.3 SQL statements and commands that can be executed in each HADB server operation mode

The SQL statements and commands that can be executed depend on the HADB server operation mode. The following table lists and explains the SQL statements and commands that can be executed in each HADB server operation mode.

Table 39-1: SQL statements and commands that can be executed in each HADB server operation mode

No.	SQL statement or command	Normal mode	Quiescence mode	Offline mode		Maintenance mode
				Execution on the server machine#1	Execution from a machine other than the server machine#2	
1	Retrieval SQL statement	Y	Y	Y	N	N
2	Update SQL statement	Y	N	Y	N	N
3	Definition SQL statement	Y	N	Y	N	N
4	Control SQL statement	Y	Y	Y	N	N
5	`adbarchivechunk` command	Y	N	Y	--	N
6	`adbaudittrail` command	Y	Y	Y	--	Y
7	`adbcancel` command	Y	Y	Y	--	Y
8	`adbchgchunkcomment` command	Y	N	Y	--	N
9	`adbchgchunkstatus` command	Y	N	Y	--	N
10	`adbchgnodetype` command	Y	Y	Y	--	Y
11	`adbchgsqltrc` command	Y	Y	Y	--	Y
12	`adbchgsrvmode` command	Y	Y	Y	--	Y
13	`adbclientdefmang` command	Y	Y	Y	--	Y
14	`adbcolumnize` command	Y	N	Y	--	Y
15	`adbconvertaudittrailfile` command	Y	Y	Y	--	Y
16	`adbdbstatus` command	Y	Y	Y	--	N
17	`adbexport` command	Y	Y	Y	--	N
18	`adbgetcst` command	Y	N	Y	--	N
19	`adbidxrebuild` command	Y	N	Y	--	N
20	`adbimport` command	Y	N	Y	--	N
21	`adbinfoget` command	Y	Y	Y	--	Y

No.	SQL statement or command	Normal mode	Quiescence mode	Offline mode		Maintenance mode
				Execution on the server machine[#1]	Execution from a machine other than the server machine[#2]	
22	`adbinfosweep` command[#3]	N	N	N	--	N
23	`adbinit` command[#3]	N	N	N	--	N
24	`adbinstall` command[#3]	N	N	N	N	N
25	`adbls` command	Y	Y	Y	--	Y
26	`adbmergechunk` command	Y	N	Y	--	N
27	`adbmodarea` command	Y	N	Y	--	Y
28	`adbmodbuff` command	Y	Y	Y	--	N
29	`adbmonitor` command	Y	Y	Y	--	Y
30	`adbreorgsystemdata` command	Y	N	Y	--	N
31	`adbsql` command[#4]	Y	Y	Y	N	N
32	`adbstart` command[#3]	N	N	N	--	N
33	`adbstat` command	Y	Y	Y	--	Y
34	`adbstop` command	Y	Y	Y	--	Y
35	`adbsyndict` command	Y	N	Y	--	N
36	`adbunarchivechunk` command	Y	N	Y	--	N

Legend:

Y: Can be executed.

N: Cannot be executed (an error results).

--: Not applicable because the command is executed on the server machine on which the HADB server is installed.

#1

Execution on the server machine means that the SQL statement or command is executed on the server machine on which the HADB server is installed.

#2

Execution from a machine other than the server machine means that the SQL statement or command is executed on a machine other than the server machine on which the HADB server is installed.

#3

The command cannot be executed while the HADB server is running.

#4

For details about the SQL statements that can be executed, see Nos. 1 through 4.

39.4 Example

This example starts the HADB server (in this example, the operation mode of the HADB server the last time it terminated is normal mode. Therefore, the HADB server starts in normal mode).

```
adbstart

KFAA90000-I adbstart processing started.
KFAA91105-I The HADB system was started normally. (HADB server operation mode = "NORM
AL")
KFAA90001-I adbstart processing ended. (return code = 0)
```

39.5 Return codes

The following table lists and describes the return codes of the `adbstart` command.

Table 39-2: Return codes of the adbstart command

Return code	Description
`0`	The `adbstart` command terminated normally.
`4`	The `adbstart` command issued a warning message or error message, and then terminated. The HADB server has already started. Check the displayed message or the message output to a message log file and take appropriate action.
`8`	An error occurred during execution of the `adbstart` command. Check the displayed message or the message output to a message log file and take appropriate action.

39.6 Notes

1. If the previous termination mode was abnormal termination, the command displays the KFAA91104-Q message when the HADB server is started. This message asks if the existing troubleshooting information can be deleted. If you have already used the adbinfoget command to collect troubleshooting information, or if there is no need to collect troubleshooting information, enter Y as the response to the KFAA91104-Q message. When Y is entered, the command discards the troubleshooting information and starts the HADB server. If you have not collected troubleshooting information with the adbinfoget command, enter N as the response to the KFAA91104-Q message. When N (or anything other than Y) is entered, the adbstart command's processing is canceled.

2. If you execute the adbstart command after the HADB server has been upgraded, the KFAA91107-Q message asking whether the database is to be upgraded will be output. If it is safe to upgrade the database, enter Y. If the adbstart command terminates normally, the HADB server has been upgraded. If upgrading might cause a problem, enter N (or any character other than Y), and the HADB server upgrade processing will be canceled and the adbstart command will terminate.

   For details about how to upgrade HADB servers, see *Upgrading the HADB server version* in *Building a System* in the *HADB Setup and Operation Guide*.

3. If the audit trail facility was enabled the last time the HADB server terminated, the audit trail facility remains in effect the next time the HADB server is started. The specification of the --write-error option of the adbaudittrail command is also inherited. For details about the adbaudittrail command, see 3. adbaudittrail (Manage the Audit Trail Facility).

4. If the audit trail facility is enabled, a new current audit trail file is created when the HADB server is started. Note that when the HADB server is restarted, the current audit trail file used the last time the HADB server terminated is renamed. Then, a new current audit trail file is created.

5. If the audit trail facility is enabled, HADB server cannot be started unless all the following conditions are satisfied. If the conditions are not satisfied, the adbstart command results in an error.

   - An existing directory is specified for the adb_audit_log_path operand in the server definition.
   - Read, write, and execution privileges are set so that the HADB administrator can access the directory specified for the adb_audit_log_path operand in the server definition.
   - The number of audit trail files stored under the directory specified for the adb_audit_log_path operand in the server definition is equal to or smaller than the value specified for the adb_audit_log_max_num operand in the server definition.

6. To return a node that has been disconnected from the multi-node configuration to the running multi-node configuration while the multi-node function is being used, perform the following:

   - Make sure there are no ongoing transactions. For example, use the adbchgsrvmode command to change the operation mode of the HADB server on the master node to maintenance mode. For details about the adbchgsrvmode command, see 9. adbchgsrvmode (Change the HADB Server Operation Mode).
   - Check the status of HADB servers on other nodes in advance. If an HADB server on any other node is engaged in termination processing, the disconnected node cannot be returned to the multi-node configuration.

     To check the status of the HADB servers, execute the adbls -d srv command on each node. On the STATUS column in the output results, you can check the status of the HADB server on the node on which the command was executed. For details about the adbls -d srv command, see 30. adbls -d srv (Display the HADB Server Status).
   - If you want to return multiple nodes to the multi-node configuration, return them one by one. You cannot return multiple nodes at the same time.

For details about returning a node to the multi-node configuration, see *Returning a node to the multi-node configuration* in *Operations when a node failure occurs* in *Operations When Using the Multi-Node Function* in the *HADB Setup and Operation Guide*.

40 adbstat (Perform Statistical Analysis of the HADB Server)

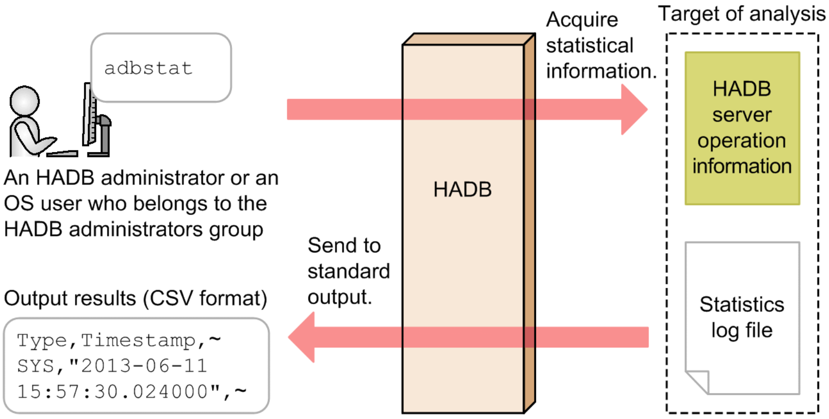
This chapter explains the functionality provided by the `adbstat` command and how to use the command.

Note that the term *tables* in this chapter means base tables.

40.1 Description

When the `adbstat` command is executed, HADB server operation information is output. The following figure provides an overview of the `adbstat` command.

Figure 40-1: Overview of the adbstat command



adbstat
An HADB administrator or an OS user who belongs to the HADB administrators group
Acquire statistical information.
Target of analysis
HADB server operation information
HADB
Send to standard output.
Statistics log file
Output results (CSV format)
Type,Timestamp,~
SYS,"2013-06-11 15:57:30.024000",~

Explanation:

When the `adbstat` command is executed, HADB server operation information is output to standard output in CSV format. You can redirect these output results to a file and forward the file to a spreadsheet program so that you can check the HADB server operation information in tabular format.

The statistics log file is used to store some of the HADB server operation information. For details about the statistics log file, see the topic *Using the statistics log files* in *Performing statistical analysis (checking HADB server operation information)* in *Scheduled Operations* in the *HADB Setup and Operation Guide*.

The `adbstat` command outputs four types of information:

- **HADB server statistical information**

  HADB server operational status information is output. Each execution of the `adbstat` command outputs only one line of HADB server statistical information.

- **Connection operation information**

  Connection operational status information is output. This information can be obtained and output for each connection that has been established with an application program.

- **Global buffer statistical information**

  Global buffer operational status information is output. This information can be obtained and output for each global buffer.

- **SQL statement statistical information**

  SQL statement operational status information is output. When the `adbstat` command is executed, information about the number of DB areas accessed by each SQL statement is output from the statistical information that has been output to the statistics log file. If an SQL statement did not access any DB areas, one line of SQL statement information is output.

  In the case of SQL statements to which batch transfer of dynamic parameter values was applied, statistical information is output collectively for all those SQL statements as if the statistical information were for a single SQL statement. For details about batch transfer of dynamic parameter values, see the topic *Batch transfer of dynamic parameter values* in *Designs Related to Improvement of Application Program Performance* in the *HADB Application Development Guide*.

Important

Access path statistical information is not output by the `adbstat` command. For details about access path statistical information, see the topic *Examples of output of and output items for access path statistical information* in *Running SQL tracing* in *Scheduled Operations* in the *HADB Setup and Operation Guide*.

40.2 Specification format for the adbstat command

This section explains the specification format and options for the `adbstat` command.

The following OS users can execute the `adbstat` command:

- HADB administrator
- OS users who belong to the HADB administrators group

40.2.1 Explanation of the specification format and options

(1) Specification format

```
adbstat
  [-c {sys|cnct|buf|sql}]
  [-n {application-identifier|global-buffer-name|connection-information}]
  [-m SQL-statement-statistical-information-output-date-and-time-specification]
  [-q SQL-statement-sequence-number]
  [-t]
  [-k enclosing-character]
  [-s {delimiter|tab}]
```

(2) Relationship between information that is output and options

The relationship between the information that is output by the `adbstat` command and the options is explained in the two tables below. For details about the information that is output, see 40.5 Items that are output.

■ Combination of options that are specified (part 1)

The table below explains the relationship between the information that is output and the options when the `adbstat` command is used to output the following information:

- HADB server statistical information
- Connection operation information
- Global buffer statistical information

Table 40-1: Relationship between information that is output by the adbstat command and the options (part 1)

No.	Information that is output	Option specifications	
		-c option	-n option
1	HADB server statistical information	`sys` or the option is omitted.	--#
2	Connection operation information for all valid connections	`cnct`	--
3	Connection operation information for a specific connection	`cnct`	*application-identifier*
4	Statistical information for all global buffers	`buf`	--
5	Statistical information for a specific global buffer	`buf`	*global-buffer-name*

Legend:

--: The `-n` option is omitted.

#

When `sys` is specified in the `-c` option or the `-c` option is omitted, specifying the `-n` option results in an error.

■ Combination of options that are specified (part 2)

The table below explains the relationship between the information that is output and the options when the `adbstat` command is used to output SQL statement statistical information.

Note

This subsection explains the recommended combinations of options for outputting SQL statement statistical information.

The amount of SQL statement statistical information can be large. By combining the options as shown in the following table to narrow down the output target, you can more easily find the information you need.

Table 40-2: Relationship between information that is output by the adbstat command and options (part 2)

No.	Information that is output	Option specifications			
		-c option	-n option	-m option	-q option
1	In the SQL statement statistical information for a specific SQL statement, the information that was output to the statistics log file during a specified period	`sql`	*connection-information*	*SQL-statement-statistical-information-output-date-and-time-specification*	*SQL-statement-sequence-number*
2	In the SQL statement statistical information for an SQL statement that was executed with a specified connection, information that was output to the statistics log file during a specified period	`sql`	*connection-information*	*SQL-statement-statistical-information-output-date-and-time-specification*	--
3	SQL statement statistical information that was output to the statistics log file during a specified period	`sql`	--	*SQL-statement-statistical-information-output-date-and-time-specification*	--
4	SQL statement statistical information for a specified SQL statement	`sql`	*connection-information*	--	*SQL-statement-sequence-number*
5	SQL statement statistical information for an SQL statement executed with a specified connection	`sql`	*connection-information*	--	--
6	The most recent five minutes of SQL statement statistical information The target range is five minutes in the past from the point when the last information was output to the statistics log file.	`sql`	--	--	--

Legend:

--: The option is omitted.

(3) Explanation of options

● `-c {sys|cnct|buf|sql}`

Specifies the type of information to be output:

`sys`: HADB server statistical information

`cnct`: Connection operation information

`buf`: Global buffer statistical information

`sql`: SQL statement statistical information

• `-n {`*application-identifier*`|`*global-buffer-name*`|`*connection-information*`}`

~<character string>

Specifies that the information for a specific connection or global buffer is to be output.

application-identifier:

Specifies the application identifier for which statistical information is to be output. You can specify this option when you specify `cnct` in the `-c` option.

global-buffer-name:

Specifies the name of the global buffer for which statistical information is to be output. You can specify this option when you specify `buf` in the `-c` option.

connection-information:

Specifies the connection for which information is to be output. You can specify this option when you specify `sql` in the `-c` option.

If you want to narrow down the SQL statement statistical information to be output, specify this option. This specification enables you to limit the output to the information for the specified connection information.

> **Note**
>
> If you want to narrow down the output target even further, specify connection information in this option, an output date and time specification in the `-m` option, and an SQL statement sequence number in the `-q` option.

If a non-existent application identifier, global buffer name, or connection information is specified in this option, no information will be output (only the title lines will be output). When the `-t` option is specified, neither information nor title lines will be output.

The following rules apply to specifying this information:

- The character string specified as an application identifier or a global buffer name is treated as being in all uppercase letters.
- If the application identifier or global buffer name includes a lowercase letter, enclose the entire application identifier or global buffer name in double quotation marks (`"`) and then enclose it in single quotation marks (`'`). An application identifier or global buffer name enclosed in both double quotation marks and single quotation marks is treated as being case-sensitive.

  Example: `-n '"ap001"'`
- The connection information is not case sensitive.
- Specify one of the following for the connection information:

    - Connection information displayed in the `KFAA81000-I` message that was output to the message log file when the connection was established
    - Connection ID that was checked in the `CID` column in the `adbls -d cnct` command's execution results

    For details about the `adbls -d cnct` command, see 23. adbls -d cnct (Display the Connection Status).

- `-m` *SQL-statement-statistical-information-output-date-and-time-specification*

  Specifies a range of SQL statement statistical information to be output in terms of date and time information that was output to the statistics log file.

  If no information within the specified range was output to the statistics log file, no information will be output (only the title lines will be output). When the `-t` option is specified, neither information nor title lines will be output.

  Specify this option in the following format:

  ```
  output-start-date-and-time[,output-end-date-and-time]
  ```

  The following rules apply to specifying this information:

  - You can specify this option when you specify `sql` in the `-c` option.
  - Specify *output-start-date-and-time* and *output-end-date-and-time* in one of the following formats:
    - *yyyy*/*mm*/*dd*Δ*hh*:*mm*:*ss*
    - *yyyy*-*mm*-*dd*Δ*hh*:*mm*:*ss*

    Legend: Δ: Space
  - Because *output-start-date-and-time* and *output-end-date-and-time* each contains a space, you must use one of the following methods to specify this information:

    Example 1: If you specify only an output start date and time, enclose the entire output start date and time in single quotation marks (`'`).

    `-m '2014/06/06 12:00:00'`

    Example 2: If you specify both an output start date and time and an output end date and time, enclose each of them in single quotation marks (`'`).

    `-m '2014/06/06 12:00:00','2014/06/06 13:00:00'`

    Example 3: If you specify both an output start date and time and an output end date and time, enclose the entire date and time specification in single quotation marks (`'`).

    `-m '2014/06/06 12:00:00,2014/06/06 13:00:00'`
  - Do not specify any spaces before or after the comma (`,`) that separates the output start date and time and the output end date and time.
  - An error will occur if a non-existent date and time is specified.
  - An error will occur if the specified output end date and time precedes the specified output start date and time.
  - If the output end date and time is omitted, all the statistical information in the statistics log file starting from the specified output start date and time is output.
  - The dates and times over which the statistical information to be output is output to the statistics log file is from the output start date and time to the output end date and time specified in this option. The command discards any fractions of a second in date and time information that was output to the statistics log file and then compares the resulting values with the specified output start date and time and output end date and time.

- `-q` *SQL-statement-sequence-number*

  ~<integer>((1 to 18,446,744,073,709,551,615))

  Specifies the sequence number of an SQL statement for which SQL statement statistical information is to be output.

  If a non-existent SQL statement sequence number is specified in this option, no information will be output (only the title lines will be output). If the `-t` option is specified, neither information nor title lines will be output.

The following rules apply to specifying this information:

- You can specify this option when you specify `sql` in the `-c` option.
- Specify in this option an *SQL-statement-sequence-number-since-connection-was-established* from the `KFAA51000-I` messages that have been output to the message log file.

> **Important**
>
> To output the `KFAA51000-I` messages to the message log file, you must have specified in advance the `adb_sql_text_out` operand in the server definition so that SQL statements will be output to the message log file.
>
> For details about the `adb_sql_text_out` operand in the server definition, see the topic *Operands related to SQL statements (set format)* in *Detailed descriptions of the server definition operands* in *Designing the Server Definition* in the *HADB Setup and Operation Guide*.

- `-t`

  Specifies that title lines (column headers) are not to be output.

- `-k` *enclosing-character*

  ~<character string>((1 byte))<<">>

  Specifies an enclosing character to be used for the character strings in CSV format. If this option is omitted, the double quotation mark (`"`) is assumed to be the enclosing character.

  A single-byte character must be specified as the enclosing character (a character string consisting of two or more bytes cannot be specified). None of the following characters are permitted to be the enclosing character:

  - Space, tab, asterisk (`*`)
  - The character that is used as the delimiter (the value of or default value for the `-s` option)

> **Important**
>
> - If you specify a single quotation mark (`'`) as the enclosing character, enclose it in double quotation marks (`"`).
>
>   Example: If you specify a single quotation mark (`'`) as the enclosing character:
>
>   `adbstat ... -k "'" ...`
>
> - If you specify as the enclosing character a special character used by the shell, such as the vertical bar (`|`), enclose it in either double quotation marks (`"`) or single quotation marks (`'`).
>
>   Example: If you specify the vertical bar (`|`) as the enclosing character:
>
>   `adbstat ... -k "|" ...`
>
> - Do not use as the enclosing character any character that is used in the column titles and contents output by the `adbstat` command. For details about the columns that are output by the `adbstat` command, see 40.5 Items that are output.

- `-s {`*delimiter*`|tab}`

  ~<character string>((1 byte))<<,>>

  Specifies the delimiter used in CSV format. To specify the tab character as the delimiter, specify `tab`. If this option is omitted, the comma (`,`) is assumed to be the delimiter.

  Only `tab` or a single-byte character can be specified as the delimiter (except for `tab`, a character string consisting of two or more bytes cannot be specified). None of the following characters can be specified as the delimiter:

- Uppercase letters (A to Z), lowercase letters (a to z), numeric characters (0 to 9), underscore (_), double quotation mark ("), asterisk (*)
- The character that is used as the enclosing character (the value of or default value for the -k option)

Important

- If you specify a single quotation mark (') as the delimiter, enclose it in double quotation marks (").

  Example: If you specify a single quotation mark (') as the delimiter:

  ```
  adbstat ... -s "'" ...
  ```

- If you specify as the delimiter a special character used by the shell, such as the vertical bar (|), enclose it in either double quotation marks (") or single quotation marks (').

  Example: If you specify the vertical bar (|) as the delimiter:

  ```
  adbstat ... -s "|" ...
  ```

- Do not use as the delimiter any character that is used in the column titles and contents output by the adbstat command. For details about the columns that are output by the adbstat command, see 40.5 Items that are output.

40.3 Examples

This section explains by way of examples how to use the `adbstat` command to check HADB server operation information.

For details about using the `adbstat` command for tuning, see *Tuning* in the *HADB Setup and Operation Guide*.

40.3.1 Checking the operational status of the entire system

This example outputs HADB server statistical information to enable checking of the operational status of the entire system.

■ Command to be executed

```
adbstat
```

Explanation:

Executes the `adbstat` command.

40.3.2 Checking the operational status of a specific application program

This example outputs connection operation information to enable checking of the operational status of a specific application program.

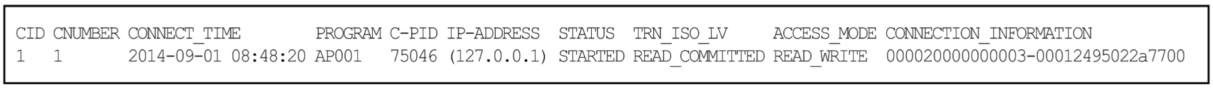
(1) Checking application identifiers

This example executes the following command to obtain the application identifier of the application program whose operation information is to be checked.

■ Command to be executed

```
adbls -d cnct
```

■ Example of execution results



```
CID CNUMBER CONNECT_TIME        PROGRAM C-PID IP-ADDRESS  STATUS  TRN_ISO_LV     ACCESS_MODE CONNECTION_INFORMATION
1   1       2014-09-01 08:48:20 AP001   75046 (127.0.0.1) STARTED READ_COMMITTED READ_WRITE  000020000000003-00012495022a7700
```

In the execution results, the `PROGRAM` column displays the application identifier of the application program that is running currently.

(2) Executing the adbstat command

■ Command to be executed

```
adbstat -c cnct              ...1
        -n AP001             ...2
```

Explanation:

1. Specifies that connection operation information is to be output.

2. Specifies the application identifier (`AP001`) of the application program for which statistical information is to be output.

40.3.3 Checking the operational status of a specific global buffer

This example outputs global buffer statistical information to enable checking of the operational status of a specified global buffer.

(1) Checking global buffer names

This example checks for the name of a global buffer to verify statistical information.

- Execute the `adbls -d gbuf` command to check the names of the global buffers. For details about the `adbls -d gbuf` command, see 24. adbls -d gbuf (Display Global Buffer Information).
- Check the message log file when the HADB server starts and obtain the global buffer names from the server definition information in the `KFAA50027-I` message that is output.
- Check the server definition file (`$ADBDIR/conf/server.def`) used to start the HADB server and obtain the global buffer names.

(2) Executing the adbstat command

■ **Command to be executed**

```
adbstat -c buf                  ...1
        -n BUF001               ...2
```

Explanation:

1. Specifies that global buffer statistical information is to be output.
2. Specifies the name of the global buffer (`BUF001`) for which statistical information is to be output.

40.3.4 Checking the SQL statement statistical information in a specific date and time range

This example outputs SQL statement statistical information to enable checking of SQL statement execution status within a specified range of dates and times.

(1) Checking SQL statement sequence numbers

This example checks the `KFAA51000-I` messages output to the message log file to obtain the SQL statement sequence number of the SQL statement for which statistical information is to be output.

> **Important**
>
> To output the `KFAA51000-I` messages to the message log file, you must have specified in advance the `adb_sql_text_out` operand in the server definition so that SQL statements will be output to the message log file.

(2) Executing the adbstat command

■ Command to be executed

```
adbstat -c sql                                            ...1
        -m '2014/09/28 13:00:00','2014/09/28 15:00:00'    ...2
        -q 4                                              ...3
```

Explanation:

1. Specifies that SQL statement statistical information is to be output.
2. Specifies the date and time range of the statistical information that is to be output (from 2014-09-28 at 13:00:00 to 2014-09-28 at 15:00:00 as the range of date and time statistical information was output to the statistics log file.
3. Specifies the SQL statement sequence number (`4`) of the SQL statement for which statistical information is to be output.

40.4 Return codes

The following table lists and explains the return codes of the adbstat command.

Table 40-3: Return codes of the adbstat command

Return code	Description
0	The adbstat command terminated normally.
8	An error occurred while the adbstat command was executing. For the cause of the error, see the displayed message or the message log file.

40.5 Items that are output

This section explains the items that are output by the `adbstat` command.

40.5.1 Items that are output as HADB server statistical information

The following table lists the titles of the items (columns) that are output as HADB server statistical information and the content that is output. The columns are output in the order listed in the table.

Table 40-4: Columns output as HADB server statistical information

No.	Column title	Information that is output	Enclosing character
1	`Type`	Type of information that is output: `SYS`: HADB server statistical information	N
2	`Timestamp`	Date and time the HADB server statistical information was output The date and time the `adbstat` command was executed (and HADB server statistical information was output) are displayed in the following format: `"`*YYYY*`-`*MM*`-`*DD*Δ*hh*`:`*mm*`:`*ss*`.`*nnnnnn*`"`#1	Y
3	`Last_execute_time`	Date and time of the previous output The last date and time that HADB server statistical information was output are displayed in the following format: `"`*YYYY*`-`*MM*`-`*DD*Δ*hh*`:`*mm*`:`*ss*`.`*nnnnnn*`"`#1 If this is the first time HADB server statistical information has been output since the HADB server started, the date and time the HADB server started are displayed.	Y
4	`Total_memory_max_size`	Maximum amount of the entire memory that has been used (bytes) This is the highest amount of memory usage between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer	N
5	`Share_memory_size`	Size of the area allocated for shared memory management and global buffer pages (bytes) This is the area size allocated during execution of the `adbstat` command. Output format: • Unsigned 8-byte integer	N
6	`Proc_memory_max_size`	Maximum amount of process common memory that has been used (bytes) This is the highest amount of memory usage between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer	N
7	`Rthd_memory_max_size`	Sum of the maximum usage amounts of all real thread private memory (bytes) This is the highest amount of memory usage between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer	N

No.	Column title	Information that is output	Enclosing character
8	`Tran_start_cnt`	Transaction start count[#2] This is the number of times transactions were started since the last time HADB server statistical information was output. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	N
9	`Tran_commit_cnt`	Commit count[#3] This is the number of times transactions were committed since the last time HADB server statistical information was output. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	N
10	`Tran_rollback_cnt`	Rollback count[#3] This is the number of times transactions were rolled back since the last time HADB server statistical information was output. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	N
11	`SQL_execute_wait_total_time`	Total wait time for SQL statement execution (seconds) This is the total amount of time that execution of SQL statements was in wait status since the last time HADB server statistical information was output. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615 seconds.	N
12	`SQL_execute_wait_cnt`	SQL statement execution wait count This is the number of times that execution of SQL statements was placed in wait status since the last time HADB server statistical information was output. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	N
13	`Wrktbl_page_use_max`	Maximum number of work table pages allocated This is the maximum number of pages allocated between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
14	`Log_usrbuf_out_cnt`	Flush count due to full user log buffer This is the number of times flushing occurred since the last time HADB server statistical information was output. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N

No.	Column title	Information that is output	Enclosing character
15	`Log_usrfile_max_size`	Maximum usage of user log file (bytes) This is the highest amount of memory usage between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer	N
16	`DBarea_extension_cnt`	DB area automatic extension count This is the number of times automatic extension of DB area files (regular files) has occurred since the last time HADB server statistical information was output.#4 Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
17	`Log_usrfile_write_cnt`	Number of times user log files have been written This is the number of times user log files have been written since the last time the HADB server statistical information was output. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
18	`Log_usrfile_reduced_cnt`	Number of times user log files were reduced to the initialization size This is the number of times user log files were reduced to the initialization size since the last time the HADB server statistical information was output. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
19	`Log_mstfile_max_size`	This value is used by the system. Output format: • Unsigned 8-byte integer	N
20	`Log_mstfile_reduced_cnt`	This value is used by the system. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
21	`Audit_file_write_cnt`	Number of times audit trail files have been written This is the number of times audit trail files have been written since the last time the HADB server statistical information was output. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
22	`Audit_file_write_size`	Data size written to the audit trail file (bytes) This is the size of data that has been written to the audit trail file since the last time the HADB server statistical information was output. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
23	`Audit_file_swap_cnt`	Number of times audit trail file has been swapped This is the number of times the audit trail file has been swapped since the last time the HADB server statistical information was output.	N

No.	Column title	Information that is output	Enclosing character
		Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	
24	`Tbldef_req_cnt`	Number of times acquisition of table-definition information was requested This is the number of times acquisition of table-definition information was requested since the last time HADB server statistical information was output. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
25	`Tbldef_access_cnt`	Number of times table-definition information was acquired from the dictionary table This is the number of times table-definition information was acquired from the dictionary table since the last time HADB server statistical information was output. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
26	`Tbldef_cache_access_cnt`	Number of times table-definition information was acquired from the table-definition pool This is the number of times table-definition information was acquired from the table-definition pool since the last time HADB server statistical information was output. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
27	`Tbldef_cache_register_cnt`	Number of times table-definition information was registered to the table-definition pool This is the number of times table-definition information was registered since the last time HADB server statistical information was output. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
28	`Tbldef_cache_sweep_cnt`	Number of times table-definition information was purged from the table-definition pool This is the number of times table-definition information was purged since the last time HADB server statistical information was output. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N

Legend:

Y: The output value is enclosed in a pair of enclosing characters.

N: The output value is not enclosed in the enclosing character.

#1

The year (*YYYY*) is output in 4 digits. The month (*MM*), date (*DD*), hours (*hh*), minutes (*mm*), and seconds (*ss*) are each output in 2 digits. The microsecond (*nnnnnn*) is output as six digits. If there are not enough digits, leading zeros are added. Δ indicates a single-byte space.

#2

This includes both the number of times transactions were started explicitly by application programs and the number of times transactions were started implicitly by the system at the time the HADB server started and during command execution.

#3

This includes both the number of times transactions were terminated explicitly from application programs by issuing the COMMIT or ROLLBACK statement and the following values:

- Number of times transactions were committed implicitly when definition SQL statements were executed
- Number of times transactions were rolled back implicitly due to an SQL error
- Number of times transactions were terminated implicitly by the system at the time the HADB server started or during command execution

#4

This does not include automatic extensions for the work table DB area. To determine whether automatic extension has occurred for the work table DB area, check the Wrktbl_page_use_max value in No. 13 in Table 40-4: Columns output as HADB server statistical information in 40.5.1 Items that are output as HADB server statistical information.

40.5.2 List of items that are output as connection operation information

The table below lists the titles of the items (columns) that are output as connection operation information and the content that is output. The columns are output in the order listed in the table.

Table 40-5: Columns output as connection operation information

No.	Column title	Information that is output	Enclosing character
1	Type	Type of information that is output: CNCT: Connection operation information	N
2	Timestamp	Date and time the connection operation information was output The date and time when the adbstat command was executed (and when the connection operation information was output) are displayed in the following format: "*YYYY*-*MM*-*DD*Δ*hh*:*mm*:*ss*.*nnnnnn*"#1	Y
3	Last_execute_time	Date and time of the previous output The last date and time connection operation information was output for the corresponding connection are displayed in the following format: "*YYYY*-*MM*-*DD*Δ*hh*:*mm*:*ss*.*nnnnnn*"#1 If this is the first time activity information has been output since the corresponding connection was established, the date and time that connection was established are displayed.# 2	Y
4	AP_name	Application identifier • For a connection from an application program The application identifier specified in the adb_clt_ap_name operand in the client definition is output. If the adb_clt_ap_name option is omitted, ******** is output. • For a connection from a command The command name is output.	Y

No.	Column title	Information that is output	Enclosing character
		In the case of the `adbsql` command, the application identifier specified in `adb_clt_ap_name` in the client definition is output.	
5	`Connection_num`	Connection sequence number This is the connection sequence number assigned since the HADB server started. Output format: • Signed 4-byte integer	N
6	`Connect_time`	Date and time the connection was established The date and time the corresponding connection was established is output in the following format: "*YYYY*-*MM*-*DD*Δ*hh*:*mm*:*ss*.*nnnnnn*"#1	Y
7	`Tran_start_cnt`	Transaction start count#3 This is the number of times transactions were started since the last time the corresponding connection operation information was output. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	N
8	`Tran_commit_cnt`	Commit count#4 This is the number of times transactions were committed since the last time the corresponding connection operation information was output. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	N
9	`Tran_rollback_cnt`	Rollback count#4 This is the number of times transactions were rolled back since the last time the corresponding connection operation information was output. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	N
10	`SQL_execute_wait_total_time`	Total wait time for SQL statement execution (seconds) This is the total amount of time that execution of SQL statements was in wait status since the last time the corresponding connection operation information was output. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615 seconds.	N
11	`SQL_execute_wait_cnt`	SQL statement execution wait count This is the number of times execution of SQL statements was placed in wait status since the last time the corresponding connection operation information was output. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	N

No.	Column title	Information that is output	Enclosing character
12	Multi_stmt_insufficient_thd_cnt	Number of times a shortage of processing threads occurred when multiple statement handles were executed concurrently This is the number of times a shortage of threads occurred since the last time connection operation information on the corresponding connection was output. Output format: • Unsigned 8-byte integer • The count is reset to 0 once it reaches 18,446,744,073,709,551,615.	N
13	SELECT_cnt	Number of times the SELECT statement was executed This is the number of times the SELECT statement was executed since the last time the corresponding connection operation information was output. An execution of the SELECT statement is counted when the cursor open processing has terminated normally. Output format: • Unsigned 8-byte integer • The count is reset to 0 once it reaches 18,446,744,073,709,551,615.	N
14	SELECT_total_time	SELECT statement execution time (seconds) This is the amount of time spent on execution of SELECT statements since the last time the corresponding connection operation information was output. This value includes the execution time of SELECT statements that terminated abnormally. Output format: • Unsigned 8-byte integer • The count is reset to 0 once it reaches 18,446,744,073,709,551,615 seconds.	N
15	Fetch_row_cnt	Number of fetched rows#5 This is the number of rows fetched since the last time the corresponding connection operation information was output. Output format: • Unsigned 8-byte integer • The count is reset to 0 once it reaches 18,446,744,073,709,551,615 rows.	N
16	INSERT_cnt	Number of times the INSERT statement was executed This is the number of times the INSERT statement was executed since the last time the corresponding connection operation information was output. An execution of an INSERT statement that terminated abnormally is not included in the count. Output format: • Unsigned 8-byte integer • The count is reset to 0 once it reaches 18,446,744,073,709,551,615.	N
17	INSERT_total_time	INSERT statement execution time (seconds) This is the amount of time spent on execution of INSERT statements since the last time the corresponding connection operation information was output.	N

No.	Column title	Information that is output	Enclosing character
		This value includes the execution time of INSERT statements that terminated abnormally. Output format: - Unsigned 8-byte integer - The count is reset to 0 once it reaches 18,446,744,073,709,551,615 seconds.	
18	UPDATE_cnt	Number of times the UPDATE statement was executed This is the number of times the UPDATE statement was executed since the last time the corresponding connection operation information was output. An execution of an UPDATE statement that terminated abnormally is not included in the count. Output format: - Unsigned 8-byte integer - The count is reset to 0 once it reaches 18,446,744,073,709,551,615.	N
19	UPDATE_total_time	UPDATE statement execution time (seconds) This is the amount of time spent on execution of UPDATE statements since the last time the corresponding connection operation information was output. This value includes the execution time of UPDATE statements that terminated abnormally. Output format: - Unsigned 8-byte integer - The count is reset to 0 once it reaches 18,446,744,073,709,551,615 seconds.	N
20	DELETE_cnt	Number of times the DELETE statement was executed This is the number of times the DELETE statement was executed since the last time the corresponding connection operation information was output. An execution of a DELETE statement that terminated abnormally is not included in the count. Output format: - Unsigned 8-byte integer - The count is reset to 0 once it reaches 18,446,744,073,709,551,615.	N
21	DELETE_total_time	DELETE statement execution time (seconds) This is the amount of time spent on execution of DELETE statements since the last time the corresponding connection operation information was output. This value includes the execution time of DELETE statements that terminated abnormally. Output format: - Unsigned 8-byte integer - The count is reset to 0 once it reaches 18,446,744,073,709,551,615 seconds.	N
22	Update_row_cnt	Number of rows updated#6 This is the number of rows updated since the last time the corresponding connection operation information was output.	N

No.	Column title	Information that is output	Enclosing character
		The SQL statements subject to counting of the number of rows updated include the `INSERT`, `UPDATE`, and `DELETE` statements. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615 rows.	
23	`TRUNCATE_TABLE_cnt`	Number of times the `TRUNCATE TABLE` statement was executed This is the number of times the `TRUNCATE TABLE` statement was executed since the last time the corresponding connection operation information was output. An execution of a `TRUNCATE TABLE` statement that terminated abnormally is not included in the count. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	N
24	`TRUNCATE_TABLE_total_time`	`TRUNCATE TABLE` statement execution time (seconds) This is the amount of time spent on execution of `TRUNCATE TABLE` statements since the last time the corresponding connection operation information was output. This value includes the execution time of `TRUNCATE TABLE` statements that terminated abnormally. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615 seconds.	N
25	`PURGE_CHUNK_cnt`	Number of times the `PURGE CHUNK` statement was executed This is the number of times the `PURGE CHUNK` statement was executed since the last time the corresponding connection operation information was output. An execution of a `PURGE CHUNK` statement that terminated abnormally is not included in the count. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	N
26	`PURGE_CHUNK_total_time`	`PURGE CHUNK` statement execution time (seconds) This is the amount of time spent on execution of `PURGE CHUNK` statements since the last time the corresponding connection operation information was output. This value includes the execution time of `PURGE CHUNK` statements that terminated abnormally. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615 seconds.	N
27	`Hashgrp_area_max_size`	Maximum usage of the hash group area[#7] (bytes) This is the highest amount of usage between the time the corresponding connection was established and the time the `adbstat` command was executed.	N

No.	Column title	Information that is output	Enclosing character
		Output format: - Unsigned 8-byte integer	
28	`Hashgrp_area_get_cnt`	Number of times the hash group area was allocated[#7] This is the number of allocations between the time the corresponding connection was established and the time the `adbstat` command was executed. Output format: - Unsigned 8-byte integer - The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	N
29	`Hashtbl_area_max_size`	Maximum usage of the hash table area (bytes) This is the highest amount of usage between the time the corresponding connection was established and the time the `adbstat` command was executed. Output format: - Unsigned 8-byte integer	N
30	`DBbuff_wrktbl_clt_read_cnt`	Number of times files were read into the local work table buffer This is the number of times files were read since the last time the corresponding connection operation information was output. Output format: - Unsigned 8-byte integer - If overflow occurs, `18446744073709551615` is output.	N
31	`DBbuff_wrktbl_clt_write_cnt`	Number of times data was written from the local work table buffer to files This is the number of times data was written to files since the last time the corresponding connection operation information was output. Output format: - Unsigned 8-byte integer - If overflow occurs, `18446744073709551615` is output.	N
32	`Log_usrbuf_out_cnt`	Flush count due to full user log buffer This is the number of times flushing occurred since the last time the corresponding connection operation information was output. Output format: - Unsigned 8-byte integer - If overflow occurs, `18446744073709551615` is output.	N
33	`Log_usrfile_max_size`	Maximum usage of user log files (bytes) This is the highest amount of usage between the time the corresponding connection was established and the time the `adbstat` command was executed. Output format: - Unsigned 8-byte integer	N
34	`DBarea_extension_cnt`	DB area automatic extension count This is the number of times automatic extension of DB area files (regular files) has occurred since the last time the corresponding connection operation information was output.[# 8] Output format: - Unsigned 8-byte integer - If overflow occurs, `18446744073709551615` is output.	N

No.	Column title	Information that is output	Enclosing character
35	`Log_usrfile_write_cnt`	Number of times user log files have been written This is the number of times user log files have been written since the last time the corresponding connection operation information was output. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
36	`DBbuff_wrktbl_clt_request_fix_cnt`	This value is used by the system.	N
37	`DBbuff_wrktbl_clt_pagein_fix_cnt`	This value is used by the system.	N
38	`DBbuff_wrktbl_clt_pageout_fix_cnt`	This value is used by the system.	N
39	`DBbuff_blk_pagein_cnt`	This value is used by the system.	N
40	`DBbuff_blk_pageout_cnt`	This value is used by the system.	N
41	`DBbuff_blk_read_cnt`	This value is used by the system.	N
42	`DBbuff_blk_write_cnt`	This value is used by the system.	N
43	`Directory_send_num`	Number of times directory information was sent This is the number of times directory information has been sent since the last time the corresponding connection operation information was output. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
44	`Directory_recv_num`	This value is used by the system.	N
45	`Log_send_num`	This value is used by the system.	N
46	`Log_recv_num`	This value is used by the system.	N
47	`DBbuff_send_num`	This value is used by the system.	N
48	`DBbuff_recv_num`	This value is used by the system.	N
49	`Node_com_num`	Number of times communication took place between nodes This is the number of times communication has taken place since the last time the corresponding connection operation information was output. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	N
50	`Node_com_time`	Communication time between nodes (microseconds) This is the time that has been used for communication since the last time the corresponding connection operation information was output. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615 microseconds.	N

No.	Column title	Information that is output	Enclosing character
51	`Log_usrfile_reduced_cnt`	Number of times user log files were reduced to the initialization size This is the number of times user log files were reduced to the initialization size since the last time the corresponding connection operation information was output. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N

Legend:

Y: The output value is enclosed in a pair of enclosing characters.

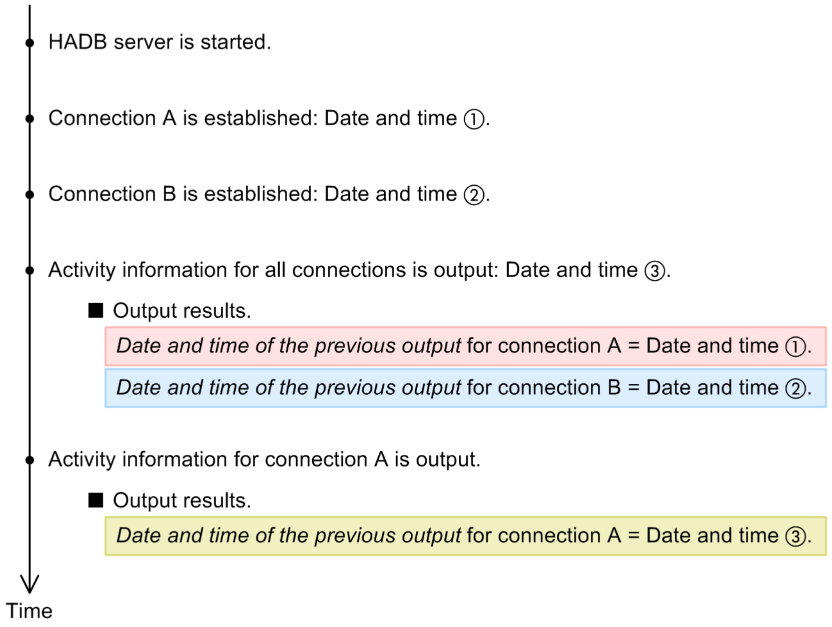
N: The output value is not enclosed in the enclosing character.

#1

The year (*YYYY*) is output as four digits. The month (*MM*), day (*DD*), hour (*hh*), minute (*mm*), and second (*ss*) are each output as two digits. The microsecond (*nnnnnn*) is output as six digits. If there are not enough digits, leading zeros are added. Δ indicates a single-byte space.

#2

The following figure shows the date and time that are output as the date and time of the previous output:


HADB server is started.
Connection A is established: Date and time ①.
Connection B is established: Date and time ②.
Activity information for all connections is output: Date and time ③.
■ Output results.
Date and time of the previous output for connection A = Date and time ①.
Date and time of the previous output for connection B = Date and time ②.
Activity information for connection A is output.
■ Output results.
Date and time of the previous output for connection A = Date and time ③.
Time

When connection operation information is output for the first time since connections A and B were established, what is displayed for the date and time of the previous output is the date and time each connection was established. The next time that connection operation information is output, the date and time of the previous output will be the date and time of the last output of the corresponding connection operation information.

#3

This includes both the number of times transactions were started explicitly by application programs and the number of times transactions were started implicitly by the system at the time the HADB server started and during command execution.

#4

This includes both the number of times transactions were terminated explicitly from application programs by issuing the `COMMIT` or `ROLLBACK` statement and the following values:

- Number of times transactions were committed implicitly when definition SQL statements were executed
- Number of times transactions were rolled back implicitly due to an SQL error
- Number of times transactions were terminated implicitly by the system during command execution

#5

This is the number of rows fetched on the HADB server. If an error occurs in an SQL statement using batch transfer of retrieval results, the displayed value might differ from the actual number of rows fetched by the application program.

The fetched row count does not include the number of rows retrieved when the following commands were executed:

- `adbarchivechunk` command
- `adbexport` command
- `adbgetcst` command
- `adbidxrebuild` command
- `adbmergechunk` command
- `adbreorgsystemdata` command

#6

If an error occurs in an SQL statement using batch transfer of retrieval results, the rows that were updated successfully before the error occurred are counted. These rows are counted as updated rows even if the processing is rolled back implicitly.

#7

If local hash grouping processing is applied, as many hash grouping areas are allocated as the number of SQL processing real threads. If local hash grouping processing is applied within a subquery that contains an external reference column, local hash grouping processing is performed as many times as the number of rows outside the subquery. Therefore, as many hash grouping areas are allocated as the value obtained from the following formula:

```
number-of-rows-in-external-queries × number-of-SQL-processing-real-threads
```

#8

This does not include automatic extensions for the work table DB area. To determine whether automatic extension has occurred for the work table DB area, check the `Wrktbl_page_use_max` value in No. 13 in Table 40-4: Columns output as HADB server statistical information in 40.5.1 Items that are output as HADB server statistical information.

40.5.3 List of items that are output as global buffer statistical information

The table below lists the titles of the items (columns) that are output as global buffer statistical information and the content that is output. The columns are output in the order listed in the table.

Table 40-6: Columns output as global buffer statistical information

No.	Column title	Information that is output	Enclosing character
1	`Type`	Type of information that is output: `BUF`: Global buffer statistical information	N

No.	Column title	Information that is output	Enclosing character
2	`Timestamp`	Date and time the global buffer statistical information was output The date and time the `adbstat` command was executed (and global buffer statistical information was output) are displayed in the following format: `"`*YYYY*`-`*MM*`-`*DD*Δ*hh*`:`*mm*`:`*ss*`.`*nnnnnn*`"`#1	Y
3	`Last_execute_time`	Date and time of the previous output The last date and time the corresponding global buffer's statistical information was output are displayed in the following format: `"`*YYYY*`-`*MM*`-`*DD*Δ*hh*`:`*mm*`:`*ss*`.`*nnnnnn*`"`#1 Note, however, that the time at which the HADB server started is output if the statistical information for the relevant global buffer is output for the first time since the HADB server was started.#2	Y
4	`DBbuff_name`	Global buffer name	Y
5	`DBbuff_page_num`	Total number of global buffer sectors This item is related to the `-p` option of the `adbbuff` operand in the server definition. It is the number of buffer sectors in existence when the `adbstat` command was executed. Output format: • Unsigned 8-byte integer	N
6	`DBbuff_page_use_cnt`	Number of global buffer sectors that have been allocated This item is related to the `-p` option of the `adbbuff` operand in the server definition. It is the number of buffer sectors that have been allocated when the `adbstat` command was executed. Output format: • Unsigned 8-byte integer	N
7	`DBbuff_page_request_cnt`	Number of page requests for the global buffer This item is related to the `-p` option of the `adbbuff` operand in the server definition. It is the number of requests issued between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
8	`DBbuff_page_hit_cnt`	Global buffer page hit count This item is related to the `-p` option of the `adbbuff` operand in the server definition. It indicates the hit count between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
9	`DBbuff_page_hit_rate`	Global buffer page hit rate#3 (%) This item is related to the `-p` option of the `adbbuff` operand in the server definition. It is the hit rate during execution of the `adbstat` command. Output format:	N

No.	Column title	Information that is output	Enclosing character
		• Unsigned 8-byte integer	
10	DBbuff_page_pagein_cnt	Global buffer page-in count This item is related to the -p option of the adbbuff operand in the server definition. It is the page-in count between the time the HADB server started and the time the adbstat command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, 18446744073709551615 is output.	N
11	DBbuff_page_pageout_cnt	Global buffer page-out count This item is related to the -p option of the adbbuff operand in the server definition. It is the page-out count between the time the HADB server started and the time the adbstat command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, 18446744073709551615 is output.	N
12	DBbuff_page_read_cnt	Number of times files were read into the global buffer This item is related to the -p option of the adbbuff operand in the server definition. It is the number of times files were read between the time the HADB server started and the time the adbstat command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, 18446744073709551615 is output.	N
13	DBbuff_page_write_cnt	Number of times data was written from the global buffer to files This item is related to the -p option of the adbbuff operand in the server definition. It is the number of times data was written to files between the time the HADB server started and the time the adbstat command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, 18446744073709551615 is output.	N
14	DBbuff_page_rng_num	Total number of range index buffer sectors This item is related to the -a option of the adbbuff operand in the server definition. It is the number of buffer sectors in existence when the adbstat command was executed. Output format: • Unsigned 8-byte integer	N
15	DBbuff_page_rng_use_cnt	Number of range index buffer sectors that have been allocated This item is related to the -a option of the adbbuff operand in the server definition. It is the number of buffer sectors that have been allocated when the adbstat command was executed. Output format: • Unsigned 8-byte integer	N

No.	Column title	Information that is output	Enclosing character
16	`DBbuff_page_rng_request_cnt`	Number of page requests for range index buffers This item is related to the `-a` option of the `adbbuff` operand in the server definition. It is the number of requests issued between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
17	`DBbuff_page_rng_hit_cnt`	Page hit count in range index buffer This item is related to the `-a` option of the `adbbuff` operand in the server definition. It is the hit count between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
18	`DBbuff_page_rng_hit_rate`	Page hit rate in range index buffer#3 (%) This item is related to the `-a` option of the `adbbuff` operand in the server definition. It is the hit rate during execution of the `adbstat` command. Output format: • Unsigned 8-byte integer	N
19	`DBbuff_page_rng_pagein_cnt`	Range index buffer page-in count This item is related to the `-a` option of the `adbbuff` operand in the server definition. It is the page-in count between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
20	`DBbuff_page_rng_pageout_cnt`	Range index buffer page-out count This item is related to the `-a` option of the `adbbuff` operand in the server definition. It is the page-out count between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
21	`DBbuff_page_rng_read_cnt`	Number of times files were read into range index buffers This item is related to the `-a` option of the `adbbuff` operand in the server definition. It is the number of times files were read between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N

No.	Column title	Information that is output	Enclosing character
22	`DBbuff_page_rng_write_cnt`	Number of times data was written from range index buffers to files This item is related to the `-a` option of the `adbbuff` operand in the server definition. It is the number of times data was written to files between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
23	`DBbuff_tblscan_request_cnt`	Number of page requests for table scan buffers This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. It is the number of requests issued between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
24	`DBbuff_tblscan_hit_cnt`	Table scan buffer page hit count[#4] This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. It is the hit count between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
25	`DBbuff_tblscan_hit_rate`	Table scan buffer page hit rate[#3, #4] (%) This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. It is the hit rate during execution of the `adbstat` command. Output format: • Unsigned 8-byte integer	N
26	`DBbuff_tblscan_pagein_cnt`	Table scan buffer page-in count[#4] This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. It is the page-in count between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
27	`DBbuff_tblscan_read_cnt`	Number of times files were read into table scan buffers[#5] This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. It is the number of times files were read between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N

No.	Column title	Information that is output	Enclosing character
28	`DBbuff_tblscan_failed_cnt`	Number of times a buffer shortage occurred in table scan buffer This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. It is the number of times a buffer shortage occurred between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
29	`DBbuff_tblscan_reuse_cnt`	Number of table scan buffer sectors reused This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. The output number is the number of buffer sectors reused between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
30	`DBbuff_tblscan_read_size`	Size of files read into the table scan buffer (kilobytes) This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. This value is the size of files that were read between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
31	`DBbuff_page_wait_cnt`	This value is used by the system.	N
32	`DBbuff_page_read_request_cnt`	Number of requests to read files into the global buffer This value is the number of requests issued between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
33	`DBbuff_page_write_request_cnt`	Number of requests to write data from the global buffer to files This value is the number of requests issued between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
34	`DBbuff_page_rng_read_request_cnt`	Number of requests to read files into the range index buffer This value is the number of requests issued between the time the HADB server started and the time the `adbstat` command was executed. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
35	`DBbuff_page_rng_write_request_cnt`	Number of requests to write data from the range index buffer to files This value is the number of requests issued between the time the HADB server started and the time the `adbstat` command was executed.	N

No.	Column title	Information that is output	Enclosing character
		Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	

Legend:

Y: The output value is enclosed in a pair of enclosing characters.

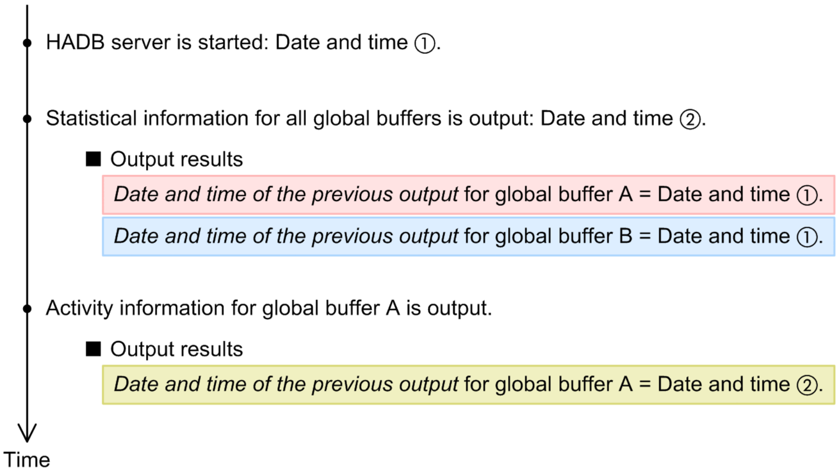
N: The output value is not enclosed in the enclosing character.

#1

The year (*YYYY*) is output as four digits. The month (*MM*), day (*DD*), hour (*hh*), minute (*mm*), and second (*ss*) are each output as two digits. The microsecond (*nnnnnn*) is output as six digits. If there are not enough digits, leading zeros are added. Δ indicates a single-byte space.

#2

The following figure shows the date and time that are output as the date and time of the previous output:



HADB server is started: Date and time ①.

Statistical information for all global buffers is output: Date and time ②.

■ Output results

Date and time of the previous output for global buffer A = Date and time ①.

Date and time of the previous output for global buffer B = Date and time ①.

Activity information for global buffer A is output.

■ Output results

Date and time of the previous output for global buffer A = Date and time ②.

When global buffer statistical information is output for the first time since the HADB server started, what is displayed for the date and time of the previous output for global buffers A and B is the date and time the HADB server was started. The next time that global buffer statistical information is output, the date and time of the previous output will be the date and time of the last output of global buffer statistical information for the corresponding buffer.

#3

Digits to the right of the decimal point are rounded up.

#4

Statistical information on accesses to the table scan buffer in pages.

#5

Statistical information on accesses to the table scan buffer in segments.

40.5.4 Items that are output as SQL statement statistical information

(1) List of items that are output as SQL statement statistical information

The following table lists the titles of the items (columns) that are output as SQL statement statistical information and the content that is output. The columns are output in the order listed in the table.

If one SQL statement accessed multiple global buffers and DB areas, separate lines of information are output under that SQL statement for the individual global buffers and DB areas. For examples of SQL statement statistical information, see (2) Examples of SQL statement statistical information.

Table 40-7: Columns output as SQL statement statistical information

No.	Column title	Information that is output	Enclosing character
1	`Type`	Type of information that is output: `SQL`: SQL statement statistical information If, for a single SQL statement, there is information about more than one DB area, the same value is output on all lines for that SQL statement. For an SQL statement that did not access any DB areas, the regular value is output.	N
2	`HADB_system_version`	Version of the HADB server that output the information to the statistics log file If, for a single SQL statement, there is information about more than one DB area, the same value is output on all lines for that SQL statement. For an SQL statement that did not access any DB areas, the regular value is output.	Y
3	`Timestamp`	Date and time the information was output to the statistics log file The date and time the corresponding information was output to the statistics log file are output in the following format: `"`*YYYY*`-`*MM*`-`*DD*Δ*hh*`:`*mm*`:`*ss*`.`*nnnnnn*`"`#1 If, for a single SQL statement, there is information about more than one DB area, the same value is output on all lines for that SQL statement. For an SQL statement that did not access any DB areas, the regular value is output.	Y
4	`AP_name`	Application identifier • For a connection from an application program The application identifier specified in the `adb_clt_ap_name` operand in the client definition is output. If the `adb_clt_ap_name` option is omitted, `********` is output. • For a connection from a command The command name is output. In the case of the `adbsql` command, the application identifier specified in `adb_clt_ap_name` in the client definition is output. If, for a single SQL statement, there is information about more than one DB area, the same value is output on all lines for that SQL statement. For an SQL statement that did not access any DB areas, the regular value is output.	Y

No.	Column title	Information that is output	Enclosing character
5	`Connection_information`	Connection information The connection information in the `KFAA81000-I` message that is output to the message log file when connection was established is output. If, for a single SQL statement, there is information about more than one DB area, the same value is output on all lines for that SQL statement. For an SQL statement that did not access any DB areas, the regular value is output.	Y
6	`Connect_time`	Date and time the connection was established The date and time the corresponding connection was established is output in the following format: "*YYYY*-*MM*-*DD*Δ*hh*:*mm*:*ss*.*nnnnnn*"[#1] If, for a single SQL statement, there is information about more than one DB area, the same value is output on all lines for that SQL statement. For an SQL statement that did not access any DB areas, the regular value is output.	Y
7	`SQL_serial_number`	SQL statement sequence number since the connection was established The SQL statement sequence number that is output in the `KFAA51000-I`[#2] and `KFAA51005-I`[#3] messages that are output to the message log file. You can identify the target SQL statement from this SQL statement sequence number. If, for a single SQL statement, there is information about more than one DB area, the same value is output on all lines for that SQL statement. For an SQL statement that did not access any DB areas, the regular value is output. Output format: • Unsigned 8-byte integer • Once `18,446,744,073,709,551,615` is reached, the value is set to `0` again.	N
8	`SQL_type`	SQL statement type[#4] The SQL statement type of the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the same value is output on all lines for that SQL statement. For an SQL statement that did not access any DB areas, the regular value is output.	N
9	`SQL_total_time`	SQL statement execution time[#5] (microseconds) Amount of time required to execute the corresponding SQL statement is output. This value is counted toward the execution time even if the SQL statement terminated abnormally. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615 microseconds.	N

No.	Column title	Information that is output	Enclosing character
10	`Fetch_row_cnt`	Number of fetched rows[#6] The number of rows retrieved by the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615 rows.	N
11	`Update_row_cnt`	Number of rows updated[#7] The number of rows updated by the corresponding SQL statement is output. The SQL statements subject to counting of the number of rows updated include the `INSERT`, `UPDATE`, and `DELETE` statements. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615 rows.	N
12	`Hashgrp_area_max_size`	Maximum usage of the hash group area[#8] (bytes) The largest value for the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer	N
13	`Hashgrp_area_get_cnt`	Number of times the hash group area was allocated[#8] The number of times the corresponding SQL statement allocated the hash group area is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	N
14	`Hashtbl_area_max_size`	Maximum usage of the hash table area (bytes) The largest value for the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer	N

No.	Column title	Information that is output	Enclosing character
15	`Lock_dbarea_request_cnt`	Number of times locked resources (DB areas) were allocated The number of times the corresponding SQL statement allocated locked resources (data DB areas and system-table DB areas) is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	N
16	`Lock_dbarea_wait_cnt`	Number of times allocation of locked resources (DB areas) was placed in wait status The number of times the allocation of locked resources (data DB areas and system-table DB areas) by the corresponding SQL statement was placed in wait status is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	N
17	`Lock_dbarea_wait_time`	Wait time for allocation of locked resources (DB areas) (microseconds) The amount of time that allocation of locked resources (data DB areas and system-table DB areas) by the corresponding SQL statement was placed in wait status is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615 microseconds.	N
18	`Lock_table_request_cnt`	Number of times locked resources (tables) were allocated The number of times the corresponding SQL statement allocated locked resources (table to be processed and system tables) is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	N
19	`Lock_table_wait_cnt`	Number of times allocation of locked resources (tables) was placed in wait status The number of times the allocation of locked resources (table to be processed and system tables) by the corresponding SQL statement was placed in wait status is output.	N

No.	Column title	Information that is output	Enclosing character
		If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	
20	`Lock_table_wait_time`	Wait time for allocation of locked resources (tables) (microseconds) The amount of time that allocation of locked resources (table to be processed and system tables) by the corresponding SQL statement was placed in wait status is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615 microseconds.	N
21	`DBbuff_dbarea_information_num`	DB area information output number If there are multiple entries for DB area information, a number starting with 1 is output to identify each entry. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer	N
22	`DBbuff_information_num`	Global buffer information output number If there are multiple entries for global buffer information per DB area, a number starting with 1 is output to identify each entry. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer	N
23	`DBbuff_dbarea_name`	DB area name If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement.	Y
24	`DBbuff_name`	Global buffer name If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement.	Y

No.	Column title	Information that is output	Enclosing character
25	`DBbuff_page_request_cnt`	Number of page requests for the global buffer The number of times a page was requested with the corresponding SQL statement is output (except for range index pages). If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
26	`DBbuff_page_hit_cnt`	Global buffer page hit count The number of times a page was hit with the corresponding SQL statement is output (except for range index pages). If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
27	`DBbuff_page_hit_rate`	Global buffer page hit rate#10 (%) The hit rate for the corresponding SQL statement is output (except for range index pages). If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer	N
28	`DBbuff_page_put_cnt`	Number of times data was written to the global buffer The number of times data was written with the corresponding SQL statement is output (except for range index pages). If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
29	`DBbuff_page_read_cnt`	Number of times files were read into the global buffer The number of times data was read with the corresponding SQL statement is output (except for range index pages). If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N

No.	Column title	Information that is output	Enclosing character
30	`DBbuff_page_write_cnt`	Number of times data was written from the global buffer to files The number of times data was written to files with the corresponding SQL statement is output (except for range index pages). If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
31	`DBbuff_page_rng_request_cnt`	Number of range index page requests for the global buffer The number of requests issued by the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
32	`DBbuff_page_rng_hit_cnt`	Range index page hit count in the global buffer The hit count for the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
33	`DBbuff_page_rng_hit_rate`	Range index page hit rate in the global buffer#10 (%) The hit rate for the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer	N
34	`DBbuff_page_rng_put_cnt`	Number of times data was written to range index pages in the global buffer The number of times data was written with the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N

No.	Column title	Information that is output	Enclosing character
35	`DBbuff_page_rng_read_cnt`	Number of times files were read into range index pages in the global buffer The number of times files were read by the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
36	`DBbuff_page_rng_write_cnt`	Number of times data was written from range index pages in the global buffer to files The number of times data was written to files by the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
37	`DBbuff_tblscan_request_cnt`	Number of page requests for table scan buffers This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. The number of times the page requests were issued by the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
38	`DBbuff_tblscan_hit_cnt`	Table scan buffer page hit count#11 This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. The hit count for the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
39	`DBbuff_tblscan_hit_rate`	Table scan buffer page hit rate#10, #11 (%) This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. The hit rate for the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an	N

No.	Column title	Information that is output	Enclosing character
		SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer	
40	`DBbuff_tblscan_read_cnt`	Number of times files were read into table scan buffers#12 This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. The number of times files were read by the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
41	`DBbuff_tblscan_failed_cnt`	Number of times a buffer shortage occurred in table scan buffer This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. The number of times a buffer shortage occurred for the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
42	`DBbuff_tblscan_insufficient_buff_num`	Number of table scan buffer sectors in which a shortage occurred This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. The number of buffer sectors in which a shortage occurred for the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
43	`DBbuff_tblscan_reset_cnt`	Number of times table scan buffers were reset This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. The number of times table scan buffers were reset by the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer	N

No.	Column title	Information that is output	Enclosing character
		- If overflow occurs, `18446744073709551615` is output.	
44	`DBbuff_wrktbl_clt_request_cnt`	Number of local work table buffer page requests The number of the page requests issued by the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer - If overflow occurs, `18446744073709551615` is output.	N
45	`DBbuff_wrktbl_clt_hit_cnt`	Local work table buffer hit count The hit count for the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer - If overflow occurs, `18446744073709551615` is output.	N
46	`DBbuff_wrktbl_clt_hit_rate`	Local work table buffer hit rate#10 (%) The hit rate for the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer	N
47	`DBbuff_wrktbl_clt_put_cnt`	Number of times data was written into the local work table buffer The number of times data was written by the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer - If overflow occurs, `18446744073709551615` is output.	N
48	`DBbuff_wrktbl_clt_read_cnt`	Number of times files were read into the local work table buffer The number of times files were read by the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer - If overflow occurs, `18446744073709551615` is output.	N
49	`DBbuff_wrktbl_clt_write_cnt`	Number of times data was written from the local work table buffer to files The number of times data was written to files by the corresponding SQL statement is output.	N

No.	Column title	Information that is output	Enclosing character
		If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	
50	`DBbuff_wrktbl_clt_tbl_cnt`	Number of times the local work table buffer was created The number of times the corresponding SQL statement created the local work table buffer is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
51	`DBbuff_wrktbl_clt_sort_merge_cnt`	Maximum number of sorting stages for the local work table The maximum number of sorting stages for the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
52	`DBbuff_wrktbl_clt_sort_page_max_cnt`	Maximum number of work table pages used for sorting the local work table The largest number of work table pages among all values for the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
53	`Log_usrbuf_out_cnt`	Flush count due to full user log buffer The flush count resulting from the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
54	`Log_usrfile_max_size`	Maximum usage of user log file (bytes) The largest value for the corresponding SQL statement is output is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format:	N

No.	Column title	Information that is output	Enclosing character
		- Unsigned 8-byte integer	
55	`DBarea_extension_cnt`	DB area automatic extension count[#13] The number of times automatic extension of DB area files (regular files) has occurred in the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer - If overflow occurs, `18446744073709551615` is output.	N
56	`Log_usrfile_write_cnt`	Number of times user log files were written[#14] The number of times user log files have been written by the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer - If overflow occurs, `18446744073709551615` is output.	N
57	`DBbuff_wrktbl_clt_request_fix_cnt`	This value is used by the system.[#14]	N
58	`DBbuff_wrktbl_clt_pagein_fix_cnt`	This value is used by the system.[#14]	N
59	`DBbuff_wrktbl_clt_pageout_fix_cnt`	This value is used by the system.[#14]	N
60	`DBbuff_wrktbl_clt_max_used_blk_num`	Maximum number of pages used in the local work table buffer[#14] The largest number of pages used by the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer	N
61	`Directory_send_num`	This value is used by the system.[#14]	N
62	`Directory_recv_num`	This value is used by the system.[#14]	N
63	`Log_send_num`	This value is used by the system.[#14]	N
64	`Log_recv_num`	This value is used by the system.[#14]	N
65	`DBbuff_send_num`	This value is used by the system.[#14]	N
66	`DBbuff_recv_num`	This value is used by the system.[#14]	N
67	`Node_com_num`	Number of times communication took place between nodes[#14] The number of times communication was performed by the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows.	N

No.	Column title	Information that is output	Enclosing character
		Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615.	
68	`Node_com_time`	Communication time between nodes#14 (microseconds) The time during which the applicable SQL statement communicated. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • The count is reset to `0` once it reaches 18,446,744,073,709,551,615 microseconds.	N
69	`Bidx_page_split_cnt`	Number of times an index page split occurred in B-tree indexes#14 The number of times an index page split in B-tree indexes occurred in the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • If an overflow occurs, `18446744073709551615` is output.	N
70	`Tidx_page_split_cnt`	Number of times an index page split occurred in text indexes#14 The number of times an index page split in text indexes occurred in the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • If an overflow occurs, `18446744073709551615` is output.	N
71	`Bidx_validation_check_cnt`	Number of times validity of row data was checked in retrievals using B-tree indexes#14 The number of times validity was checked in the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • If an overflow occurs, `18446744073709551615` is output.	N
72	`Ridx_sgmt_skip_cnt`	Number of times table segments were skipped in retrievals using range indexes#14 The number of times segments were skipped in the applicable SQL statement.	N

No.	Column title	Information that is output	Enclosing character
		If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • If an overflow occurs, `18446744073709551615` is output.	
73	`Ridx_chunk_skip_cnt`	Number of times table chunks were skipped in retrievals using range indexes#14 The number of times chunks were skipped in the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • If an overflow occurs, `18446744073709551615` is output.	N
74	`Ridx_chunk_read_cnt`	Number of times table chunks were not skipped in retrievals using range indexes#14 The number of times chunks were not skipped in the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • If an overflow occurs, `18446744073709551615` is output.	N
75	`Ridx_chunk_judge_cnt`	Number of times storage of data values by table chunks in the range satisfying the search condition was checked in retrievals using range indexes#14 The number of times checking occurred in the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • If an overflow occurs, `18446744073709551615` is output.	N
76	`Ridx_sgmt_read_cnt`	Number of times table segments were not skipped in retrievals using range indexes#14 The number of times segments were not skipped in the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer	N

No.	Column title	Information that is output	Enclosing character
		- If an overflow occurs, `18446744073709551615` is output.	
77	`Ridx_sgmt_judge_cnt`	Number of times storage of data values by table segments in the range satisfying the search condition was checked in retrievals using range indexes#14 The number of times checking occurred in the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer - If an overflow occurs, `18446744073709551615` is output.	N
78	`SegmentRel_rthd_max_num`	Maximum number of processing real threads used during segment release#14 The largest number of processing real threads used during segment release in the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer - If an overflow occurs, `18446744073709551615` is output.	N
79	`Csvread_file_cnt`	Number of files opened by the `ADB_CSVREAD` function#14 The number of files opened by the `ADB_CSVREAD` function of the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer - If an overflow occurs, `18446744073709551615` is output.	N
80	`Csvread_file_read_size`	Total size of data that was read by the `ADB_CSVREAD` function#14 (bytes) The total size of data that was read by the `ADB_CSVREAD` function of the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer - If an overflow occurs, `18446744073709551615` is output.	N
81	`Hashgrp_area_shortage`	Occurrence of a hash grouping area shortage#14 Either of the following values is output: - `Y`: A hash grouping area shortage occurred in the applicable SQL statement.	N

No.	Column title	Information that is output	Enclosing character
		- N: A hash grouping area shortage did not occur in the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows.	
82	Hashgrp_area_sufficient_size	Size needed to avoid a hash grouping area shortage#14 (kilobytes) The size needed to prevent a hash grouping area shortage from occurring in the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer - If an overflow occurs, 18446744073709551615 is output.	N
83	Hashtbl_area_shortage	Occurrence of a hash table area shortage#14 Either of the following values is output: - Y: A hash table area shortage occurred in the applicable SQL statement. - N: A hash table area shortage did not occur in the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows.	N
84	Syndict_file_access_time	Total access time to the synonym dictionary file#15 (microseconds) Total amount of time required for the applicable SQL statement to access the synonym dictionary file is output. This value is counted toward the access time even if the SQL statement terminated abnormally. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer - If overflow occurs, 18446744073709551615 is output.	N
85	Auditread_file_cnt	Number of files opened by the ADB_AUDITREAD function#16 The number of files opened by the ADB_AUDITREAD function of the applicable SQL statement. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer - If overflow occurs, 18446744073709551615 is output.	N
86	Auditread_file_read_size	Total size of data that was read by the ADB_AUDITREAD function#16 (bytes) The total size of data that was read by the ADB_AUDITREAD function of the applicable SQL statement.	N

No.	Column title	Information that is output	Enclosing character
		If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	
87	`DBbuff_tblscan_read_size`	Size of files read into the table scan buffer[#17] (kilobytes) This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. The size of files read by the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.[#9] If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
88	`DBbuff_tblscan_rthd_min_size`	Minimum value of memory in the table scan buffer used by each real thread[#17, #18] (bytes) This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. The smallest amount of memory in the table scan buffer used by each real thread of the applicable SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.[#9] If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
89	`DBbuff_tblscan_rthd_max_size`	Maximum value of memory in the table scan buffer used by each real thread[#17, #18] (bytes) This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. The largest amount of memory in the table scan buffer used by each real thread of the applicable SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.[#9] If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
90	`DBbuff_tblscan_use_size`	Total memory usage of the table scan buffer during execution of the SQL statement[#17, #18] (bytes) This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. The total amount of memory that was used in the table scan buffer during execution of the applicable SQL statement is output.	N

No.	Column title	Information that is output	Enclosing character
		If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	
91	`DBbuff_tblscan_insufficient_buff_size`	Total amount of memory requested when there was not enough free space in the table scan buffer during execution of the SQL statement#17, #18 (bytes) This item is related to the `-k` and `-v` options of the `adbbuff` operand in the server definition. The total amount of memory that was requested when there was not enough free space in the table scan buffer during execution of the applicable SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output for each DB area.#9 If an SQL statement did not access any DB areas, no value is output for that SQL statement. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
92	`Hashflt_disabled`	Whether the hash filter is invalidated#19 Either of the following values is output: • `Y`: A hash filter is applicable to the corresponding SQL statement, but the hash filter is invalidated. • `N`: Other than above If any part of the hash filter is invalidated, `Y` is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows.	N
93	`Tbldef_req_cnt`	Number of times acquisition of table-definition information was requested#19 The number of times acquisition of table-definition information was requested by the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer • If overflow occurs, `18446744073709551615` is output.	N
94	`Tbldef_access_cnt`	Number of times table-definition information was acquired from the dictionary table#19 The number of times table-definition information was acquired from the dictionary table by the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer	N

No.	Column title	Information that is output	Enclosing character
		- If overflow occurs, `18446744073709551615` is output.	
95	`Tbldef_cache_access_cnt`	Number of times table-definition information was acquired from the table-definition pool#19 The number of times table-definition information was acquired from the table-definition pool by the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer - If overflow occurs, `18446744073709551615` is output.	N
96	`Tbldef_cache_register_cnt`	Number of times table-definition information was registered to the table-definition pool#19 The number of times table-definition information was registered by the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer - If overflow occurs, `18446744073709551615` is output.	N
97	`Tbldef_cache_sweep_cnt`	Number of times table-definition information was purged from the table-definition pool#19 The number of times table-definition information was purged by the corresponding SQL statement is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer - If overflow occurs, `18446744073709551615` is output.	N
98	`DBbuff_page_wait_cnt`	This value is used by the system.#20	N
99	`Max_sql_rthd_num`	Maximum number of SQL processing real threads#21 The maximum number of SQL processing real threads when the corresponding SQL statement was executed is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer	N
100	`Hashtbl_area_size`	Size of the hash table area#21 (megabytes) The size of the hash table area when the corresponding SQL statement was executed is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: - Unsigned 8-byte integer	N

No.	Column title	Information that is output	Enclosing character
101	`Hashflt_area_size`	Size of the hash filter area#21 (megabytes) The size of the hash filter area when the corresponding SQL statement was executed is output. If, for a single SQL statement, there is information about more than one DB area, the value is output only on the first row for that SQL statement. No value is output in the subsequent rows. Output format: • Unsigned 8-byte integer	N

Legend:

Y: The output value is enclosed in a pair of enclosing characters.

N: The output value is not enclosed in the enclosing character.

#1

The year (*YYYY*) is output in 4 digits. The month (*MM*), date (*DD*), hours (*hh*), minutes (*mm*), and seconds (*ss*) are each output in 2 digits. The microsecond (*nnnnnn*) is output as six digits. If there are not enough digits, leading zeros are added. Δ indicates a single-byte space.

#2

To output the `KFAA51000-I` messages to the message log file, you must have specified in advance the `adb_sql_text_out` operand in the server definition so that SQL statements will be output to the message log file.

#3

The `KFAA51005-I` message is output if the SQL statement results in an error.

#4

The following table lists and describes the SQL statement types that are displayed.

Table 40-8: SQL statement types

No.	SQL statement type	Description
1	`ALTER_TABLE`	Changing table definitions
2	`ALTER_USER`	Changing HADB user information
3	`ALTER_VIEW`	Re-creating viewed tables
4	`CREATE_AUDIT`	Defining audit targets
5	`CREATE_INDEX`	Defining indexes
6	`CREATE_SCHEMA`	Defining schemata
7	`CREATE_TABLE`	Defining tables
8	`CREATE_USER`	Creating HADB users
9	`CREATE_VIEW`	Defining viewed tables
10	`DROP_AUDIT`	Deleting audit target definitions
11	`DROP_INDEX`	Deleting indexes
12	`DROP_SCHEMA`	Deleting schemata
13	`DROP_TABLE`	Deleting tables

No.	SQL statement type	Description
14	DROP_USER	Deleting HADB users
15	DROP_VIEW	Deleting viewed tables
16	GRANT	Granting privileges
17	REVOKE	Revoking privileges
18	DELETE	Deleting rows
19	INSERT	Inserting rows
20	PURGE_CHUNK	Deleting all rows from a chunk
21	SELECT	Retrieving rows
22	TRUNCATE_TABLE	Deleting all rows from a base table
23	UPDATE	Updating rows
24	No value is output.	The SQL statement type is other than one of the above or the SQL statement preprocessing resulted in an error.

#5

The following SQL statements' execution time is counted:

- SELECT statement
- INSERT statement
- UPDATE statement
- DELETE statement
- TRUNCATE TABLE statement
- PURGE CHUNK statement

#6

This is the number of rows fetched on the HADB server. If an error occurs in an SQL statement using batch transfer of retrieval results, the displayed value might differ from the actual number of rows fetched by the application program.

The fetched row count does not include the number of rows retrieved when the following commands were executed:

- adbarchivechunk command
- adbexport command
- adbgetcst command
- adbidxrebuild command
- adbmergechunk command
- adbreorgsystemdata command

#7

If an error occurs in an SQL statement using batch transfer of retrieval results, the rows that were updated successfully before the error occurred are counted. If the SQL statement is rolled back implicitly, the number of updated rows is 0.

#8

If local hash grouping processing is applied, as many hash grouping areas are allocated as the number of SQL processing real threads. If local hash grouping processing is applied within a subquery that contains an external reference column, local hash grouping processing is performed as many times as the number of rows outside the subquery. Therefore, as many hash grouping areas are allocated as the value obtained from the following formula:

```
number-of-rows-in-external-queries × number-of-SQL-processing-real-threads
```

#9

When a global buffer page is requested, global buffer flushing might occur. In such a case, the page subject to flushing might not be the page accessed by the SQL statement that requested the page. Therefore, information about a DB area or global buffer that is not accessed by the target SQL statement might be output as SQL statement statistical information.

#10

Digits to the right of the decimal point are rounded up.

#11

Statistical information on accesses to the table scan buffer in pages.

#12

Statistical information on accesses to the table scan buffer in segments.

#13

This does not include automatic extensions for the work table DB area. To determine whether automatic extension has occurred for the work table DB area, check the value of Wrktbl_page_use_max in No. 13 in Table 40-4: Columns output as HADB server statistical information in 40.5.1 Items that are output as HADB server statistical information.

#14

No value is output if the value of the HADB_system_version column is 03-00.

A value is output if the value of the HADB_system_version column is 03-01 or later.

#15

The access time is counted each time the synonym dictionary file is accessed, but the time less than a microsecond is cut off. Therefore, if every access requires less than 1 microsecond, the total access time is 0.

#16

No value is output if the value of the HADB_system_version column is earlier than 03-06.

A value is output if the value of the HADB_system_version column is 04-00 or later.

#17

No value is output if the value of the HADB_system_version column is earlier than 04-00.

A value is output if the value of the HADB_system_version column is 04-01 or later.

#18

The value of *memory size used for the table scan buffer* specified in the -v option of the adbbuff operand in the server definition includes the area used for managing the table scan buffer. For this reason, you cannot assign the entire memory specified by the -v option to real threads. Therefore, the value output for the output item applicable to this note does not match the value of *memory size used for the table scan buffer* or *maximum memory size that can be used for each real thread* specified in the -v.

#19

No value is output if the value of the HADB_system_version column is earlier than 04-02.

A value is output if the value of the HADB_system_version column is 04-03 or later.

#20

No value is output if the value of the HADB_system_version column is earlier than 04-03.

A value is output if the value of the HADB_system_version column is 05-00 or later.

#21

No value is output if the value of the `HADB_system_version` column is earlier than 05-00.

A value is output if the value of the `HADB_system_version` column is 05-01 or later.

(2) Examples of SQL statement statistical information

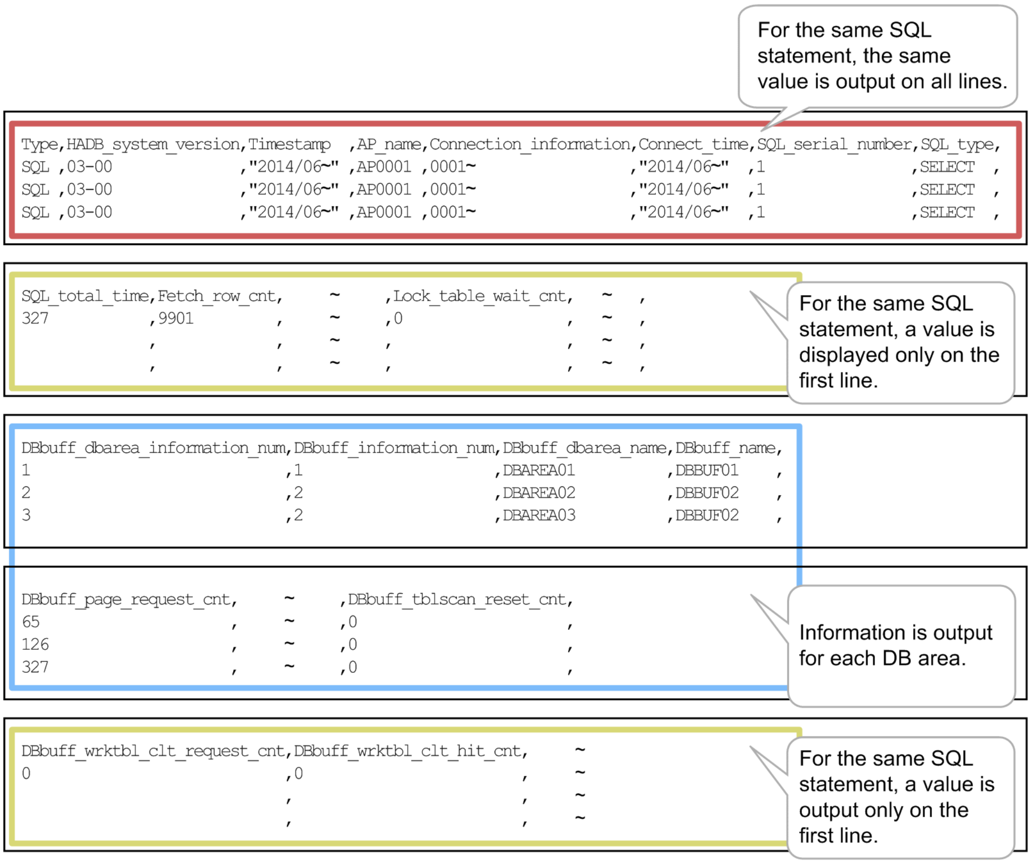
The SQL statement statistical information item values that are output depend on whether the corresponding SQL statement accessed DB areas. This subsection shows examples of SQL statement statistical information when the target SQL statement accessed DB areas and when it did not access any DB areas.

Note

In the examples provided below, spaces are used to align the items for readability. The lines in the actual output results are very long so, in the figure, each line is divided into multiple parts.

The following figure shows an example of SQL statement statistical information when the target SQL statement accessed three DB areas.

Figure 40-2: When the SQL statement accessed three DB areas



For the same SQL statement, the same value is output on all lines.

```
Type,HADB_system_version,Timestamp  ,AP_name,Connection_information,Connect_time,SQL_serial_number,SQL_type,
SQL ,03-00              ,"2014/06~" ,AP0001 ,0001~                 ,"2014/06~"  ,1                ,SELECT  ,
SQL ,03-00              ,"2014/06~" ,AP0001 ,0001~                 ,"2014/06~"  ,1                ,SELECT  ,
SQL ,03-00              ,"2014/06~" ,AP0001 ,0001~                 ,"2014/06~"  ,1                ,SELECT  ,
```

```
SQL_total_time,Fetch_row_cnt,      ~    ,Lock_table_wait_cnt,    ~   ,
327           ,9901         ,      ~    ,0                  ,    ~   ,
              ,             ,      ~    ,                   ,    ~   ,
              ,             ,      ~    ,                   ,    ~   ,
```

For the same SQL statement, a value is displayed only on the first line.

```
DBbuff_dbarea_information_num,DBbuff_information_num,DBbuff_dbarea_name,DBbuff_name,
1                            ,1                     ,DBAREA01          ,DBBUF01    ,
2                            ,2                     ,DBAREA02          ,DBBUF02    ,
3                            ,2                     ,DBAREA03          ,DBBUF02    ,
```

```
DBbuff_page_request_cnt,      ~    ,DBbuff_tblscan_reset_cnt,
65                     ,      ~    ,0                       ,
126                    ,      ~    ,0                       ,
327                    ,      ~    ,0                       ,
```

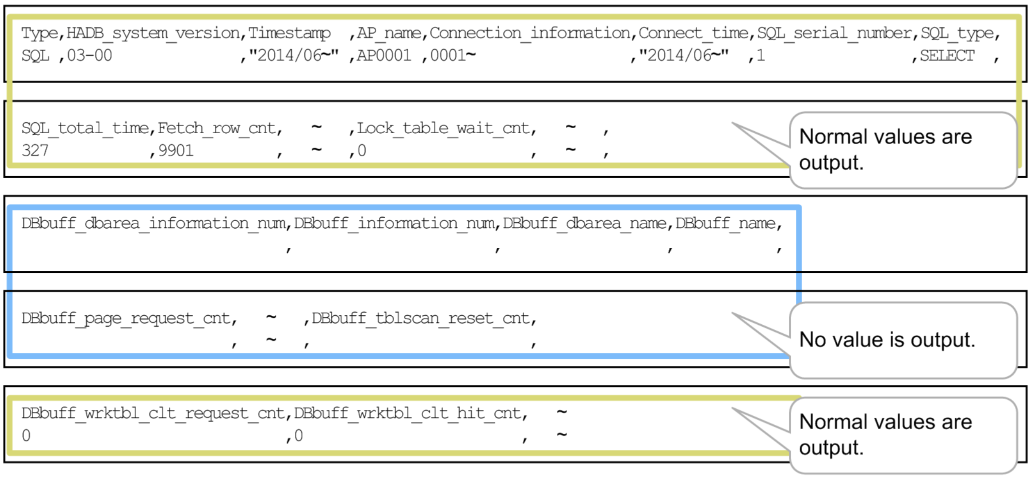
Information is output for each DB area.

```
DBbuff_wrktbl_clt_request_cnt,DBbuff_wrktbl_clt_hit_cnt,      ~
0                            ,0                        ,      ~
                             ,                         ,      ~
                             ,                         ,      ~
```

For the same SQL statement, a value is output only on the first line.

The following figure shows an example of SQL statement statistical information when the target SQL statement did not access any DB areas.

Figure 40-3: When the SQL statement did not access any DB areas



```
Type,HADB_system_version,Timestamp  ,AP_name,Connection_information,Connect_time,SQL_serial_number,SQL_type,
SQL ,03-00                ,"2014/06~" ,AP0001 ,0001~                 ,"2014/06~"  ,1                ,SELECT  ,

SQL_total_time,Fetch_row_cnt,   ~   ,Lock_table_wait_cnt,   ~   ,
327           ,9901         ,   ~   ,0                  ,   ~   ,

DBbuff_dbarea_information_num,DBbuff_information_num,DBbuff_dbarea_name,DBbuff_name,
                             ,                      ,                  ,           ,

DBbuff_page_request_cnt,   ~   ,DBbuff_tblscan_reset_cnt,
                       ,   ~   ,                        ,

DBbuff_wrktbl_clt_request_cnt,DBbuff_wrktbl_clt_hit_cnt,   ~
0                            ,0                        ,   ~
```

40.6 Output rules for the adbstat command

The following are the output rules for the `adbstat` command:

1. Column titles are output on the top rows in the output results. The titles do not use enclosing characters.
   If the `-t` option is specified, no titles are output.
2. When the `adbstat` command is executed, information about SQL statements and application programs is output in addition to information about system operations.
3. The date and time information that is output is based on the time zone set in the `TZ` environment variable at the machine where the HADB server is installed.
4. When output information is a numeric value and overflow occurs, it is possible that no value will be output.
5. The last item output is suffixed with a linefeed (`LF`) code.
6. The characters used in the output results are based on the character encoding set in the `ADBLANG` environment variable at the machine where the HADB server is installed.

40.7 Notes

1. The `adbstat` command does not output a start message or a termination message with a return code. If the `adbstat` has terminated abnormally, check the displayed message or message log file and take appropriate action.
2. The `adbstat` command references the HADB server status without applying a lock so that the HADB server's processing performance is not affected. This means that the command might output inconsistent values depending on the execution timing. In such a case, re-execute the `adbstat` command.
3. Do not use the **Ctrl**+**C** keys or the `kill` OS command to terminate the `adbstat` command forcibly. The next time the `adbstat` command is executed after the command process was terminated forcibly, inconsistent values might be output.
4. Because the `adbstat` command does not change any contents of DB areas, tables, or indexes, there is no need to restore tables or indexes in the event of abnormal termination of the `adbstat` command.

41 adbstop (Terminate the HADB Server)

This chapter explains the functionality provided by the `adbstop` command and how to use the command.

41.1 Description

The adbstop command terminates the HADB server.

The two HADB server termination methods are normal termination and forced termination. You use an option so specify the termination method.

Normal termination

- The command waits until all connected application programs have disconnected and all executing commands have terminated, and then terminates the HADB server.
- The command waits until all active transactions have completed and all executing commands have terminated, and then terminates the HADB server.
- The command cancels all active transactions and all executing commands, and then terminates the HADB server.

> **Note**
>
> If you wish to terminate the HADB server normally when there is no connected application program or executing command, execute the adbstop command without either option specified.

Forced termination

The command terminates the HADB server forcibly without waiting for active transactions to be completed and executing commands to be terminated.

When the multi-node function is being used, execute the adbstop command on the master node. When you execute the adbstop command on the master node, all HADB servers in the multi-node configuration are terminated. To normally terminate only the HADB server on the node where the adbstop command is executed, execute the adbstop command with --node specified.

41.2 Specification format for the adbstop command

This section explains the specification format and options for the `adbstop` command.

The only OS users who can execute the `adbstop` command are HADB administrators.

41.2.1 Explanation of the specification format and options

(1) Specification format

```
adbstop [{--wait {connection|transaction} [-t timeout-value] [--node]
         |--cancel [--node]
         |--force
         |--node}]
```

(2) Explanation of options

Specifies the HADB server termination method.

- `--wait {connection|transaction} [-t` *timeout-value*`]`

  The command is to wait until all application programs have disconnected from the HADB server, all active transactions have completed, and all executing commands have terminated, and then the HADB server is to be terminated normally.

  connection

  Specifies that the command is to wait until all application programs have disconnected from the HADB server and all executing commands have completed, and then the HADB server is to be terminated.

  Note that once the `adbstop` command with `--wait connection` specified is executed, the HADB server will no longer accept new connection establishment requests. The HADB server will still accept new transaction start requests.

  transaction

  Specifies that the command is to wait until all active transactions have completed, all executing commands have terminated, and all applications that are connected to the HADB server have disconnected, and then the HADB server is to be terminated.

  Note that once the `adbstop` command with `--wait transaction` specified is executed, the HADB server will no longer accept new connection establishment requests or new transaction start requests.

  `-t` *timeout-value*

  ~<integer>((1 to 86,400))<<3,600>>(seconds)

  This option can be specified only when `--wait connection` or `--wait transaction` is specified.

  Specifies the amount of time (in seconds) the command is to wait for disconnection of application programs, completion of transactions, and termination of commands. If the amount of time specified in this option expires (resulting in a timeout) while the `adbstop --wait` command is still executing, the HADB server is not terminated and it can resume accepting both new connection establishment requests and new transaction start requests.

- `--cancel`

  The command is to cancel all active transactions and all executing commands, then disconnect all application programs from the HADB server, and then terminate the HADB server.

Note that once the adbstop command with --cancel specified is executed, the HADB server will no longer accept new connection establishment requests or new transaction start requests.

- --force

  The command is to terminate the HADB server forcibly without waiting for active transactions to be completed or executing commands to be terminated. When the HADB server is terminated forcibly, transaction recovery processing will be performed the next time the HADB server is started.

  Note that when the adbstop command with --force specified is executed, the KFAA91153-Q message is output. The KFAA91153-Q message asks you to confirm that you really want to terminate the HADB server forcibly. If you want to terminate the HADB server forcibly, enter Y. If anything other than Y is entered, adbstop command processing is canceled.

  - **If the multi-node function is used:**

    If the adbstop command with --force specified is executed, only the HADB server on the node on which the command is executed is forcibly terminated. If the command is executed on the master node, one of the slave nodes is switched to the master node.

- --node

  This option is related to the multi-node function. This option is specified when terminating an HADB server on a specific node. In the following cases, execute the adbstop command with the --node option specified.

  - Terminating the HADB server on a specific slave node

    On the slave node on which you want to terminate the HADB server, execute the adbstop command with the --node option specified.

  - Switching over the master node (when SCSI reservation for shared disk is in use)

    To switch over the master node when SCSI reservation for shared disk is used, execute the adbstop command with the --node option specified on the master node to terminate the master node's HADB server. For details about how to switch over the master node, see the topic *Switching over the master node by using a command* in the *HADB Setup and Operation Guide*.

    Important

    If host reset is being used rather than SCSI reservation for shared disk, do not execute the adbstop command with the --node option specified on the master node. If host reset is being used and the adbstop command with the --node option specified is accidentally executed on the master node, execute HA Monitor's monswap command to switch over the master node. For details, see *Operations When Using the Multi-Node Function* in the *HADB Setup and Operation Guide*: *Normally terminating the HADB server on the master node only* under *Terminating an HADB server on a specific node* under *Terminating HADB servers in the multi-node configuration* in *Starting and terminating HADB servers in the multi-node configuration*

  Note

  When the multi-node function is not being used, this option is ignored.

  The system processing for when both the --node option and another option are specified is described in the table below.

Table 41-1: System processing for when both the `--node` option and another option are specified

No.	Options specified with the `--node` option	System processing
1	None	If the `adbstop` command is executed on a slave node If an ongoing transaction exists on the node on which the `adbstop` command is executed, an error occurs (the `KFAA50021-E` message is output). If the `adbstop` command is executed on the master node An error occurs (the `KFAA50021-E` message is output) in the following cases. • An ongoing transaction exists on the master node. • A connection has been established with an HADB server in the multi-node configuration, but no transaction has been executed on that connection.
2	`--wait connection`	If the `adbstop` command is executed on a slave node Termination processing waits for the completion of the ongoing transaction on the node on which the `adbstop` command is executed. The termination processing begins when all transactions have been completed. In this case, start requests for new transactions are not accepted by that node. If the `adbstop` command is executed on the master node Termination processing waits for the completion of the ongoing transaction on the master node. If a connection established with an HADB server in the multi-node configuration has not been used to execute transactions, termination processing waits for the completion of that connection. The termination processing begins when all transactions and connections have been completed. In this case, the master node does not accept new transaction start requests or new connection establishment requests.
3	`--wait transaction`	If the `adbstop` command is executed on a slave node Termination processing waits for the completion of the ongoing transaction on the node on which the `adbstop` command is executed. The termination processing begins when all transactions have been completed. In this case, start requests for new transactions are not accepted by that node. If the `adbstop` command is executed on the master node Termination processing waits for the completion of the ongoing transaction on the master node. Even if a connection established with an HADB server in the multi-node configuration has not been used to execute transactions, the termination processing begins upon completion of the ongoing transaction on the master node. (The connection is terminated forcibly). In this case, the master node does not accept new transaction start requests or new connection establishment requests.
4	`--cancel`	If the `adbstop` command is executed on a slave node The ongoing transactions on the node on which the `adbstop` command is executed are canceled, and the connection that is executing that transaction is terminated. Next, the termination processing starts. In this case, start requests for new transactions are not accepted by that node. If the `adbstop` command is executed on the master node The ongoing transaction on the master node is canceled. Then, the connection used for executing that transaction is terminated. If a connection established with an HADB server in the multi-node configuration has not been used to execute transactions, that connection is terminated. The termination processing begins when all connections have been completed. In this case, the master node does not accept new transaction start requests or new connection establishment requests.

If no option is specified, the HADB server's termination mode is set to normal termination. If any application program is connected to the HADB server or a command is executing, the `KFAA91152-E` message will be output and an error will result (the HADB server cannot be terminated). If this happens, see the manual *HADB Messages* and take the appropriate action described for the `KFAA91152-E` message.

41.3 Example

This example terminates the HADB server.

```
adbstop

KFAA90000-I adbstop processing started.
KFAA91154-I The HADB system was terminated normally.
KFAA90001-I adbstop processing ended. (return code = 0)
```

41.4 Return codes

The following table lists and describes the return codes of the adbstop command.

Table 41-2: Return codes of the adbstop command

Return code	Description
0	The adbstop command terminated normally.
4	The adbstop command issued a warning message and then terminated. The HADB server has already been terminated. Check the displayed message or the message output to a message log file and take appropriate action. If, while the HADB server is performing the termination processing started by the adbstop command, the adbstop --force command is executed to forcibly terminate the HADB server, the return code of the preceding adbstop is 4. In addition, the KFAA91154-I message that appears indicates that the HADB server is forcibly terminated.
8	An error occurred during execution of the adbstop command. Check the displayed message or the message output to a message log file and take appropriate action.

41.5 Notes

1. The `adbstop --cancel` command cannot be executed while the `adbstop --wait` command is executing. If you want to cancel termination standby processing (which is in effect while waiting for application programs to be disconnected from the HADB server, active transactions to be completed, and executing commands to be terminated) during execution of the `adbstop --wait` command, use the OS's `kill` command to terminate the `adbstop --wait` command's process.

2. In the following cases, the HADB server can resume acceptance of new connection establishment requests and new transaction start requests (the HADB server returns to the same status it was in before the `adbstop --wait` command executed).

   - A timeout occurs during termination standby processing.
   - Termination standby processing is canceled by the method described in 1.

3. If the multi-node function is used and a node is returning to the multi-node configuration, the `adbstop` command can only be executed with `--force` option specified. (The `adbstop` command without an option specified or with any other option specified cannot be executed.)

4. If the multi-node function is used, the `adbstop --node` command can be executed on the slave node that will be the new master node during master node switchover. At this time, the HADB server on the new master node (on which the `adbstop --node` command was executed) terminates after master node switchover has been completed. In this case, master node switchover occurs again, causing the slave node with the highest priority to be the master node. If a switchover destination node does not exist, the HADB servers in the multi-node configuration terminate.

42 adbsyndict (Manage Synonym Dictionary)

This chapter explains the functionality provided by the `adbsyndict` command and how to use the command.

42.1 Description

This section explains the functionality provided by the adbsyndict command.

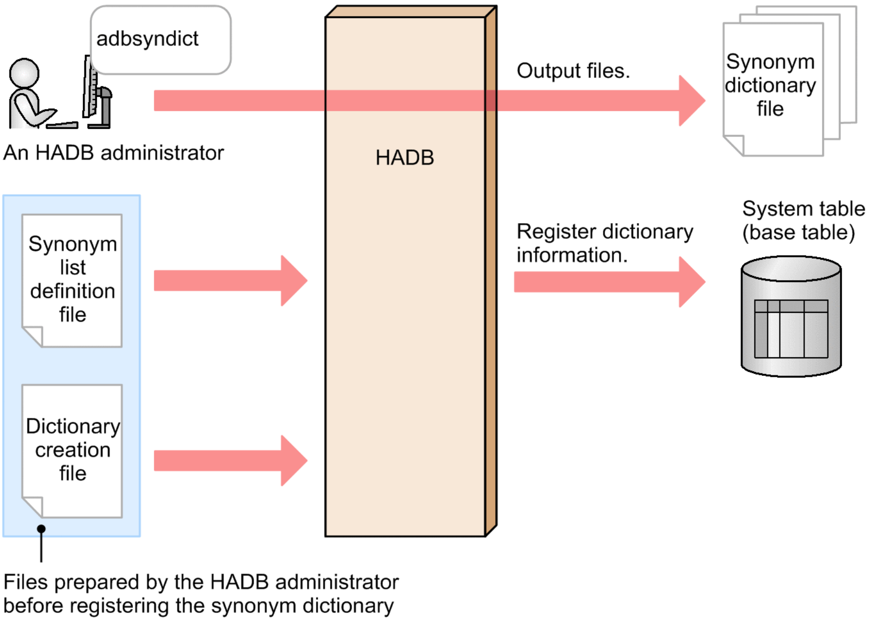
When executed, the adbsyndict command establishes a connection to the HADB server.

42.1.1 Registering a synonym dictionary

Adding a new synonym dictionary is called registering a synonym dictionary. When you execute the adbsyndict command, a synonym dictionary is created based on the specified synonym list definition file and the specified dictionary creation file, and the synonym dictionary is then output to the synonym dictionary file.

The following figure gives an overview of using the adbsyndict command to register a synonym dictionary.

Figure 42-1: Overview of using the adbsyndict command to register a synonym dictionary


adbsyndict
An HADB administrator
HADB
Output files.
Synonym dictionary file
System table (base table)
Register dictionary information.
Synonym list definition file
Dictionary creation file
Files prepared by the HADB administrator before registering the synonym dictionary

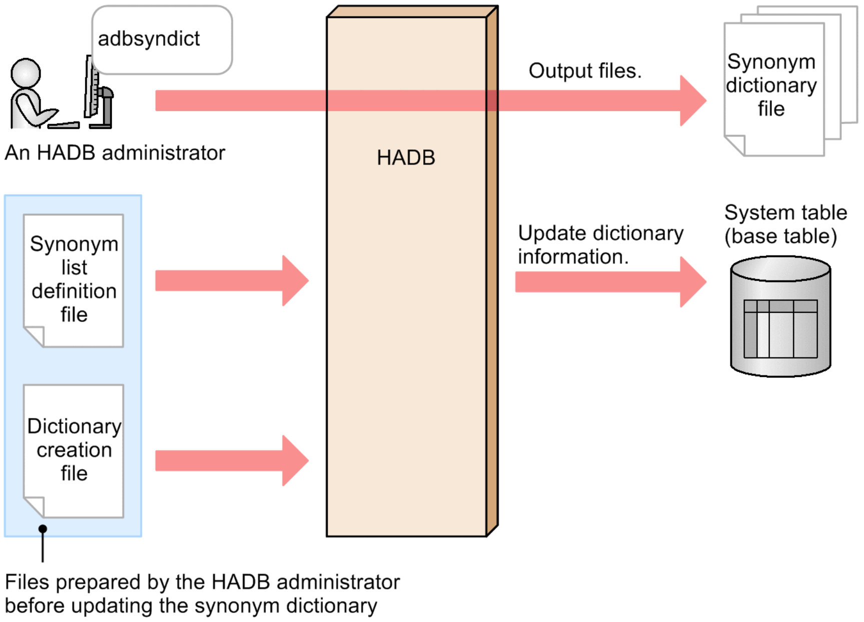
A synonym dictionary is necessary for performing synonym searches. For details about synonym searches, see *Synonym search* in the *HADB Setup and Operation Guide*.

42.1.2 Updating a synonym dictionary

Updating the contents of a registered synonym dictionary is called updating a synonym dictionary. Updating a synonym dictionary includes, for example, adding, deleting, or updating synonyms, or adding or deleting a synonym group. When you execute the adbsyndict command, the synonym dictionary is re-created based on the specified synonym list definition file and dictionary creation file, and then the synonym dictionary is output to the synonym dictionary file.

The following figure gives an overview of using the adbsyndict command to update a synonym dictionary.

Figure 42-2: Overview of using the adbsyndict command to update a synonym dictionary

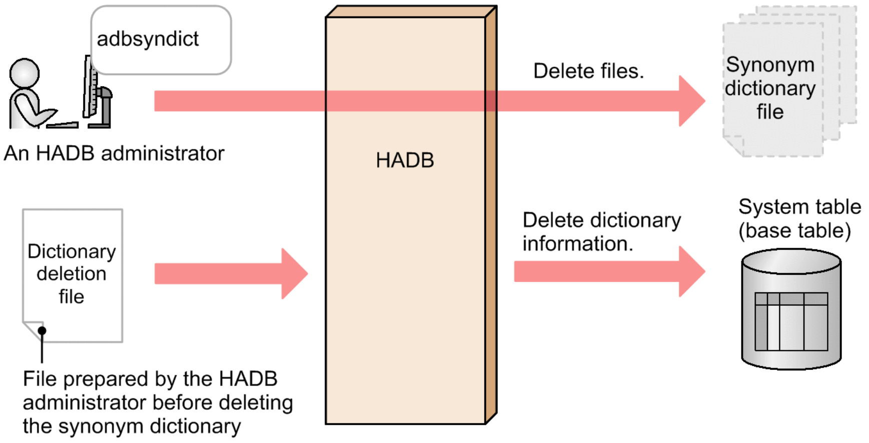

adbsyndict
An HADB administrator
HADB
Output files.
Synonym dictionary file
Update dictionary information.
System table (base table)
Synonym list definition file
Dictionary creation file
Files prepared by the HADB administrator before updating the synonym dictionary

42.1.3 Deleting a synonym dictionary

You can delete synonym dictionaries that are no longer necessary. When you execute the `adbsyndict` command, the synonym dictionary specified in the dictionary deletion file is deleted.

The following figure gives an overview of using the `adbsyndict` command to delete a synonym dictionary.

Figure 42-3: Overview of using the adbsyndict command to delete a synonym dictionary


adbsyndict
An HADB administrator
HADB
Delete files.
Synonym dictionary file
Delete dictionary information.
System table (base table)
Dictionary deletion file
File prepared by the HADB administrator before deleting the synonym dictionary

42.1.4 Synchronizing synonym dictionary files

Copying the synonym dictionary files from the master node to slave nodes to make the synonym dictionary files the same on all nodes is called synchronizing synonym dictionary files. When the `adbsyndict -s` command is executed, the synonym dictionary files under the directory specified for the `adb_syndict_storage_path` operand for the master node are copied to the directory specified for the `adb_syndict_node_storage_path` operand for all nodes.

Note

The HADB server checks whether the latest synonym dictionary file exists under the directory specified for the `adb_syndict_node_storage_path` operand for each node. If the latest file does not exist, the HADB server copies the synonym dictionary file from the master node.

If a synonym search is performed when the multi-node function is used, the synonym dictionary files stored under the directory specified for the `adb_syndict_node_storage_path` operand for each node are used. SQL statement processing for synonym search is not performed on a slave node on which an old synonym dictionary file is stored. If SQL statement processing for synonym search is assigned to such a slave node, the slave node is switched to the master node, where the SQL statement processing is to be performed.

When a synonym dictionary is registered, updated, or deleted, synonym dictionary files are synchronized automatically. Therefore, you need to execute the `adbsyndict -s` command to synchronize synonym dictionary files only in the following cases:

- A node is added.
- A node is returned.
- Synchronization processing of synonym dictionary files has not terminated normally.

42.1.5 Outputting a list of synonyms registered in a synonym dictionary

You can output a list of synonyms registered in a synonym dictionary to a file. The following shows an example of how a list of synonyms is output to a file.

Output example

```
database,data bank,DB↓
application server,AP server↓
```

Legend: ↓: Line break

The list of synonyms is output in the same format as the synonym list definition file. As shown in the above example, one row contains a list of synonyms of one synonym group.

If the synonym list definition file has been lost, you need to re-create it by executing the `adbsyndict` command.

Note

A synonym list definition file is re-created by the `adbsyndict` command according to the following rules:

- Each synonym is enclosed in enclosing characters (double quotation marks).
- The line break code is `X'0A'` (`LF`).

The re-created file might differ from the original synonym list definition file (you created) as follows:

- Even if the same synonym is specified more than once in the synonym group, only one synonym is output.
- The order in which synonyms are specified in the synonym group might be different.

42.1.6 Deleting unnecessary files from under the storage directory for synonym dictionary files

If the adbsyndict command used for registering or updating a synonym dictionary has not terminated normally, unnecessary files might remain in the storage directory for synonym dictionary files. (This directory is specified for the adb_syndict_storage_path operand in the server definition.) Those unnecessary files can be deleted# by executing the adbsyndict command.

#

The following files are deleted:

- Synonym dictionary file that is not used for synonym search (synonym dictionary file that is not registered on the HADB server)
- Temporary work file that was not deleted when, for example, a synonym dictionary was registered

42.2 Specification format for the adbsyndict command

This section explains the specification format and options for the `adbsyndict` command.

The following OS user can execute the `adbsyndict` command:

- HADB administrator

42.2.1 Explanation of the specification format and options

(1) Specification format

```
adbsyndict {-m dictionary-creation-file-name
           |-d dictionary-deletion-file-name
           |-s
           |-n synonym-dictionary-name -o synonym-list-output-file-name
           |--clean}
```

(2) Explanation of options

- `-m` *dictionary-creation-file-name*

  ~<OS path name>((2 to 510 bytes))

  Specify a dictionary creation file name when registering or updating a synonym dictionary. Specify an absolute path for the dictionary creation file name.

  For details about dictionary creation files, see *Creating a dictionary creation file* in *Preparing for synonym search operations* in the *HADB Setup and Operation Guide*.

- `-d` *dictionary-deletion-file-name*

  ~<OS path name>((2 to 510 bytes))

  Specify a dictionary deletion file name when deleting a synonym dictionary. Specify an absolute path for the dictionary deletion file name.

  For details about dictionary deletion files, see *Deleting synonym dictionaries* in the *HADB Setup and Operation Guide*.

- `-s`

  The `adbsyndict` command executed with the `-s` option specified synchronizes the synonym dictionary files on all nodes.

  The `-s` option is specified when the multi-node function is used. If the `-s` option is specified when the multi-node function is not used, the `adbsyndict` command causes an error.

- `-n` *synonym-dictionary-name* `-o` *synonym-list-output-file-name*

  Specify this option to output a list of synonyms registered in a synonym dictionary to a file.

  - `-n` *synonym-dictionary-name*

    ~<character string>((1 to 120 bytes))

    Specify the name of the synonym dictionary from which a list of synonyms is to be output.

  - `-o` *synonym-list-output-file-name*

    ~<OS path name>((2 to 510 bytes))

    Specify the absolute path name of the file to which the synonym list is to be output.

- --clean

  Unnecessary files are deleted from the synonym dictionary file directory specified for the adb_syndict_storage_path operand in the server definition. If a synonym dictionary file or temporary work file failed to be deleted, execute the adbsyndict command with the --clean option specified.

42.3 Examples

This section describes by way of examples how to use the `adbsyndict` command.

42.3.1 Registering a synonym dictionary

In this example, the following two synonym dictionaries are registered:

- Dictionary1
- Dictionary2

This example uses the following procedure to execute the `adbsyndict` command.

Procedure

1. Create a synonym list definition file.
2. Create a dictionary creation file.
3. Execute the `adbsyndict` command.

Using examples, the following subsections explain each of these steps.

(1) Creating a synonym list definition file

A synonym list definition file specifying a list of synonyms is created.

Example of specifying a synonym list definition file (Dictionary1)

```
database,data bank,DB↓
```

Example of specifying a synonym list definition file (Dictionary2)

```
application server,AP server↓
```

Legend: ↓: Line break

In this example, the above two synonym list definition files are saved under the following file names:

- `/home/adbmanager/dictionary1_synonym.txt`
- `/home/adbmanager/dictionary2_synonym.txt`

(2) Creating a dictionary creation file

A dictionary creation file is created, which specifies information about the synonym dictionary to be registered.

Example of specifying a dictionary creation file

```
Dictionary1,/home/adbmanager/dictionary1_synonym.txt,CORRECTIONRULE,database-synon
ym↓
Dictionary2,/home/adbmanager/dictionary2_synonym.txt,CASESENSITIVE,application-ser
ver-synonym↓
```

Legend: ↓: Line break

Explanation:

- 1st column

  Specifies the synonym dictionary name.
- 2nd column

  Specifies the absolute path name of the synonym list definition file created in (1) Creating a synonym list definition file.
- 3rd column

  The notation-correction option is specified. Specify either of the following:

  `CORRECTIONRULE`: Specify this option when creating a synonym dictionary that supports notation-correction searches.

  `CASESENSITIVE`: Specify this option when creating a synonym dictionary that does not support notation-correction searches.
- 4th column

  Specifies the synonym dictionary comment.

In this example, the above dictionary creation file is saved under the following file name:

- `/home/adbmanager/dictionary_information.txt`

(3) Executing the adbsyndict command

```
adbsyndict -m /home/adbmanager/dictionary_information.txt
```

Explanation:

In the `-m` option, specify the absolute path name of the dictionary creation file created in (2) Creating a dictionary creation file.

Note

- For the rules for specifying synonym list definition files, see *Specification rules for a synonym list definition file* in the *HADB Setup and Operation Guide*.
- For details about the rules for specifying dictionary creation files, see *Specification rules for a dictionary creation file* in the *HADB Setup and Operation Guide*.

42.3.2 Updating a synonym dictionary

A synonym is added to the synonym group registered to the synonym dictionary Dictionary1. In this example, *RDB* is added to the synonym group.

This example uses the following procedure to execute the `adbsyndict` command.

Procedure

1. Correct the specification of the stored synonym list definition file.
2. Check the specification of the stored dictionary creation file.
3. Execute the `adbsyndict` command.

Using examples, the following subsections explain each of these steps.

(1) Correcting the specification of the stored synonym list definition file

Example of specifying the synonym list definition file

```
database,data bank,DB,relational database↓
```

Legend: ↓: Line break

The synonym list definition file for the stored Dictionary1 is corrected. In this example, the underlined synonym is added.

In this example, the synonym list definition file for Dictionary1 is saved under the following file name:

- `/home/adbmanager/dictionary1_synonym.txt`

(2) Checking the specification of the stored dictionary creation file

Example of specifying a dictionary creation file

```
Dictionary1,/home/adbmanager/dictionary1_synonym.txt,CORRECTIONRULE,database-synon
ym↓
```

Legend: ↓: Line break

Check the specification of the stored dictionary creation file. Change the comment in view of the added synonym, if necessary.

In this example, the above dictionary creation file is saved under the following file name:

- `/home/adbmanager/dictionary1_information.txt`

(3) Executing the adbsyndict command

```
adbsyndict -m /home/adbmanager/dictionary1_information.txt
```

Explanation:

In the `-m` option, specify the absolute path name of the dictionary creation file you checked in (2) Checking the specification of the stored dictionary creation file.

Note

- For the rules for specifying synonym list definition files, see *Specification rules for a synonym list definition file* in the *HADB Setup and Operation Guide*.
- For details about the rules for specifying dictionary creation files, see *Specification rules for a dictionary creation file* in the *HADB Setup and Operation Guide*.

42.3.3 Deleting a synonym dictionary

In this example, the following two synonym dictionaries are deleted:

- Dictionary1
- Dictionary2

This example uses the following procedure to execute the `adbsyndict` command.

Procedure

1. Create a dictionary deletion file.
2. Execute the `adbsyndict` command.

Using examples, the following subsections explain each of these steps.

(1) Creating a dictionary deletion file

A dictionary deletion file is created, which specifies the name of the synonym dictionary to be deleted.

Example of specifying a dictionary deletion file

```
Dictionary1↓
Dictionary2↓
```

Legend: ↓: Line break

In this example, the above dictionary deletion file is saved under the following file name:

- `/home/adbmanager/dictionary_delete.txt`

(2) Executing the adbsyndict command

```
adbsyndict -d /home/adbmanager/dictionary_delete.txt
```

Explanation:

In the `-d` option, specify the absolute path name of the dictionary deletion file created in (1) Creating a dictionary deletion file.

Note

For the rules for specifying dictionary deletion files, see *Deleting synonym dictionaries* in the *HADB Setup and Operation Guide*.

42.3.4 Synchronizing synonym dictionary files

Synonym dictionary files are synchronized. In this case, execute the following `adbsyndict` command.

```
adbsyndict -s
```

42.3.5 Outputting a list of synonyms registered in a synonym dictionary

A list of synonyms registered in synonym dictionary Dictionary1 is output to /home/adbmanager/dictionary1_output.txt. In this case, execute the following adbsyndict command.

```
adbsyndict -n Dictionary1 -o /home/adbmanager/dictionary1_output.txt
```

Output example of /home/adbmanager/dictionary1_output.txt

```
database,data bank,DB↓
application server,AP server↓
```

Legend: ↓: Line break

The list of synonyms is output in the same format as the synonym list definition file. As shown in the above example, one row contains the synonyms of one synonym group.

42.3.6 Deleting unnecessary files from the storage directory for synonym dictionary files

Unnecessary files are deleted from the synonym dictionary file directory specified for the adb_syndict_storage_path operand in the server definition. In this case, execute the following adbsyndict command.

```
adbsyndict --clean
```

42.4 Return codes

The following table lists and describes the return codes of the adbsyndict command.

Table 42-1: Return codes of the adbsyndict command

Return code	Description
0	The adbsyndict command terminated normally.
4	The adbsyndict command terminated normally, but a warning message was issued. Registration, update, or deletion of the synonym dictionary is complete. Check the displayed message or the message output to a message log file and take appropriate action.
8	One of the following errors occurred: • The specified adbsyndict command did not execute because it was invalid. • An error occurred during execution of the adbsyndict command. For the cause of the error and corrective action to take, see the displayed error message or the message that has been output to a message log file.

■ Action to be taken if return code 4 is returned when a synonym dictionary is registered, updated, or deleted

- **If the KFAA51504-W or KFAA51520-W message is issued**

  Processing to delete a synonym dictionary file or temporary work file that is no longer necessary might have failed. See the displayed warning message, and then take the corrective action described for that warning message.

  If the file that is no longer necessary remains, delete it by using the `adbsyndict --clean` command.

  Then, if you are using the multi-node function, execute the `adbsyndict -s` command to synchronize the synonym dictionary files. Files that are no longer necessary are automatically deleted from the slave node during synchronization processing of synonym dictionary files.

  Note

  Even if an unnecessary synonym dictionary file or temporary work file exists, you can perform a synonym search and execute the adbsyndict command.

- **If the KFAA51518-W message is issued**

  This warning message is issued if an attempt is made to register, update, or delete a synonym dictionary or to synchronize synonym dictionary files while there is a stopped node. The stopped node is not subject to synchronization processing of synonym dictionary files. Therefore, after the node is returned, you need to execute the `adbsyndict -s` command to synchronize the synonym dictionary files.

- **If the KFAA51524-W message is issued**

  This warning message is issued if processing to delete the synonym dictionary file that is no longer necessary is canceled. Use the `adbsyndict --clean` command to delete the synonym dictionary file that is no longer necessary.

  Then, if you are using the multi-node function, execute the `adbsyndict -s` command to synchronize the synonym dictionary files. The synonym dictionary files that are no longer necessary are automatically deleted from the slave node during synchronization processing of synonym dictionary files.

  Note

  Even if an unnecessary synonym dictionary file remains, you can perform a synonym search and execute the adbsyndict command.

■ Action to be taken if return code 4 is returned when synonym dictionary files are synchronized

The possible causes are as follows:

- Synchronization processing of a synonym dictionary file failed.
- There is a node that has not been synchronized.
- Processing to delete a temporary work file failed, or deletion processing was canceled.

See the warning message output to the message log file, and then take the corrective action described for the warning message. Then, execute the `adbsyndict -s` command to synchronize the synonym dictionary files.

■ Action to be taken if return code 4 is returned when a list of synonyms is output

- **If the KFAA51523-W message is issued**

  The synonym dictionary file from which a list of synonyms has been output might be in invalid status. Take the corrective action described for the `KFAA51523-W` message.

■ Action to be taken if the KFAA51504-W or KFAA51520-W message is output when return code 8 is returned

Processing to delete a synonym dictionary file or temporary work file that is no longer necessary might have failed. See the displayed warning message, and then take the corrective action described for that warning message.

If the unnecessary file remains, delete it by using the `adbsyndict --clean` command.

Then, if you are using the multi-node function, execute the `adbsyndict -s` command to synchronize the synonym dictionary files. The synonym dictionary files that are no longer necessary are automatically deleted from the slave node during synchronization processing of synonym dictionary files.

> **Note**
>
> Even if an unnecessary synonym dictionary file or temporary work file exists, you can re-execute the `adbsyndict` command.

42.5 Handling abnormal termination of the adbsyndict command

If the `adbsyndict` command terminates abnormally, the synonym dictionary remains in the same state as before the command was executed. Therefore, after eliminating the cause of the command terminating abnormally, re-execute the `adbsyndict` command.

42.6 Notes

1. Only one `adbsyndict` command can be executed at a time.
2. Use the `adbcancel` command to cancel execution of the `adbsyndict` command. Alternatively, use a method such as the `adbstop --force` command or the OS's `kill` command to forcibly terminate the server process. For details about how to use the `adbcancel` command to suspend an executing command, see 1.6 When suspending (forcibly terminating) an executing command.
3. Synonym dictionary registration and deletion is executed in server processes. Therefore, using the **Ctrl**+**C** keys to forcibly terminate the `adbsyndict` command process will not interrupt synonym dictionary registration, update, or deletion processing that is being performed by server processes (this processing continues). When the synonym dictionary registration, update, or deletion processing is complete, the `KFAA81001-I` message is output to the message log file.
4. Do not store any files other than the synonym dictionary file in the storage directory for the synonym dictionary file.
5. If a synonym dictionary file that is not registered on the HADB server is stored in the storage directory for synonym dictionary files, that file is deleted when the `adbsyndict` command is executed. This does not apply when the command is used for synchronizing synonym dictionary files or outputting a synonym list.

   Note

   If you forcibly terminate the server processes by using commands such as the `adbstop --force` command or OS's `kill` command while a synonym dictionary file is being created, the file might remain in the storage directory for the synonym dictionary file, but will be deleted automatically the next time the `adbsyndict` command is executed.

43 adbunarchivechunk (Unarchive Chunk)

This chapter explains the functionality provided by the `adbunarchivechunk` command and how to use the command.

43.1 Description

This section explains the functionality provided by the `adbunarchivechunk` command and the threads it uses.

The `adbunarchivechunk` command establishes a connection to the HADB server.

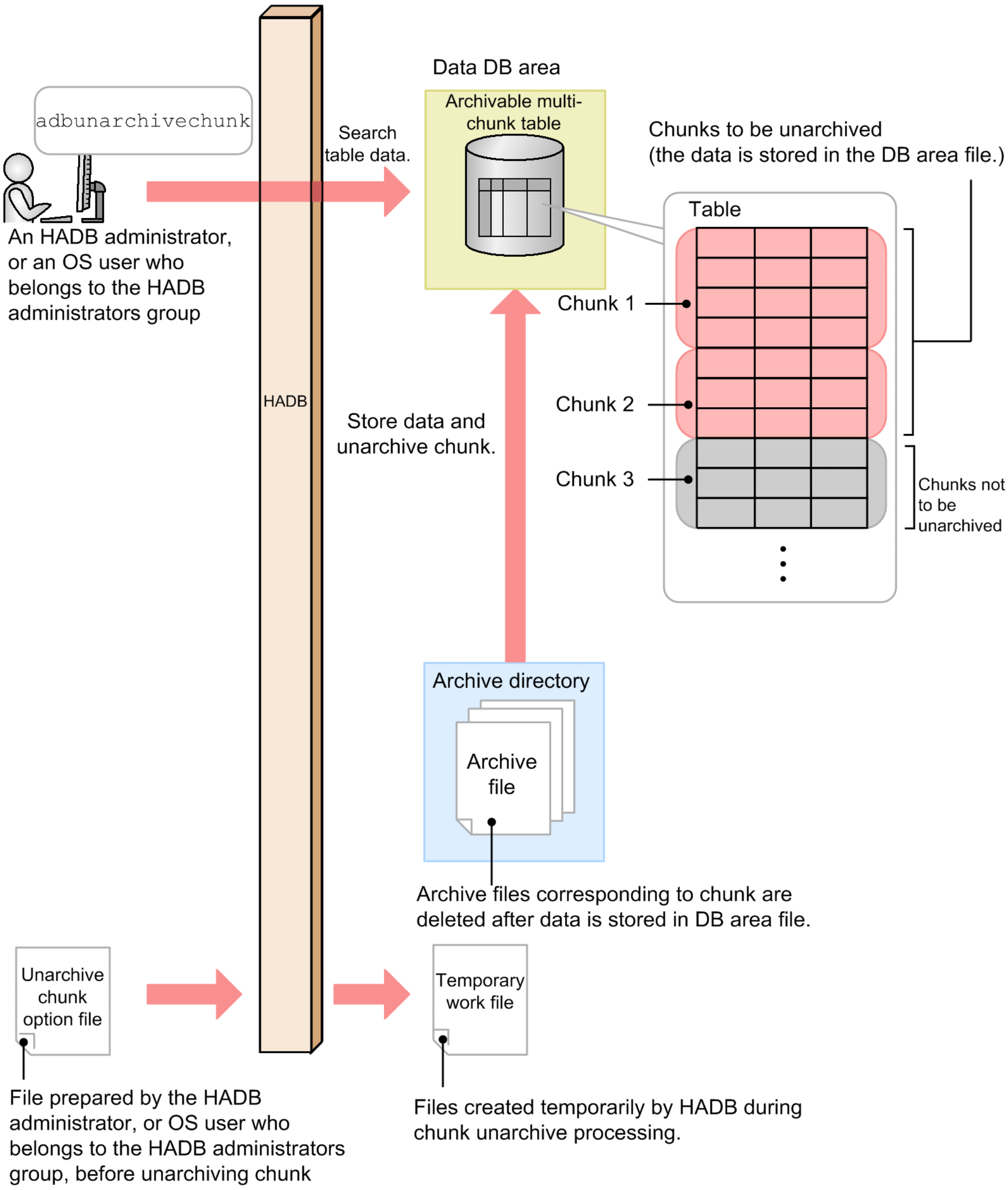
43.1.1 Functional overview

When the `adbunarchivechunk` command is executed, the archived state of chunks can be removed (chunks can be unarchived). You can decompress a compressed chunk by removing its archived state. Removing the archived state of a chunk is called *chunk unarchive processing*.

When chunk unarchive processing is executed, the target archived chunk data (archive file data) is decompressed and stored in a DB area file. This removes the archived state of the chunk.

The following figure provides an overview of chunk unarchive processing.

Figure 43-1: Overview of the adbunarchivechunk command


adbunarchivechunk
An HADB administrator, or an OS user who belongs to the HADB administrators group
HADB
Search table data.
Data DB area
Archivable multi-chunk table
Chunks to be unarchived (the data is stored in the DB area file.)
Table
Chunk 1
Chunk 2
Chunk 3
Chunks not to be unarchived
Store data and unarchive chunk.
Archive directory
Archive file
Archive files corresponding to chunk are deleted after data is stored in DB area file.
Unarchive chunk option file
File prepared by the HADB administrator, or OS user who belongs to the HADB administrators group, before unarchiving chunk
Temporary work file
Files created temporarily by HADB during chunk unarchive processing.

Explanation:

When you execute the adbunarchivechunk command, the target archived chunk data (archive file data) is decompressed and stored in a DB area file. This removes the archived state from the chunk created in the archivable multi-chunk table.

After the storage in the DB area file is processed, the archive file corresponding to the chunk is deleted.

Note that multiple chunks can be unarchived during one execution of the adbunarchivechunk command. When multiple chunks are unarchived, the storage processing and deletion processing is repeated as many times as the number of chunks.

For details about unarchiving chunks, see *Expanding (unarchiving) data* in *Chunk archiving function (compressing data in a chunk)* in the *HADB Setup and Operation Guide*.

43.1.2 Relationships among threads used by the adbunarchivechunk command

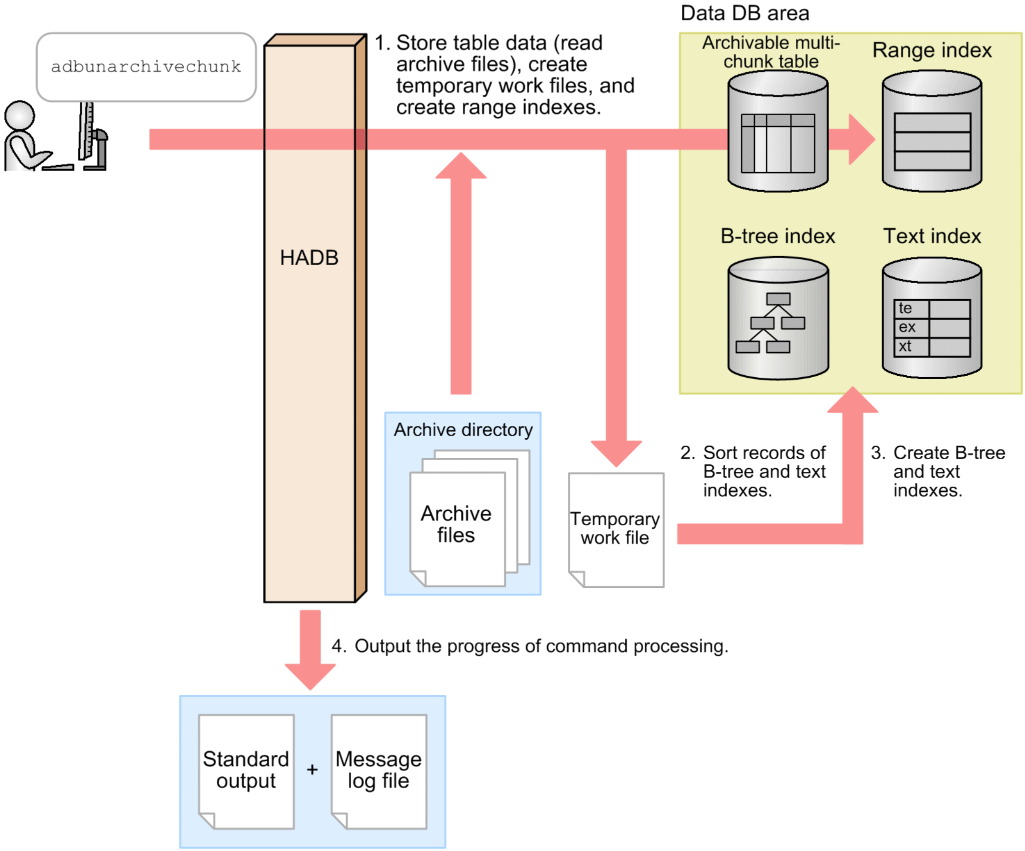
The `adbunarchivechunk` command uses processing real threads in the following processing:

1. Storing table data (reading archive files), creating temporary work files, and creating range indexes
   One or more processing real threads are used.
2. Sorting records of B-tree and text indexes
   One or more processing real threads are used.
3. Creating B-tree and text indexes
   One or more processing real threads are used.
4. Outputting the progress of command processing
   One processing real thread is used.

Steps 1 to 3 are performed in sequence. Step 4 is always performed while steps 1 to 3 are in progress.

The following figure shows the processing of the `adbunarchivechunk` command using processing real threads. The preceding numbers correspond to the numbers in the figure.

Figure 43-2: Processing of the adbunarchivechunk command using processing real threads


adbunarchivechunk
HADB
1. Store table data (read archive files), create temporary work files, and create range indexes.
Data DB area
Archivable multi-chunk table
Range index
B-tree index
Text index
te
ex
xt
Archive directory
Archive files
Temporary work file
2. Sort records of B-tree and text indexes.
3. Create B-tree and text indexes.
4. Output the progress of command processing.
Standard output
+
Message log file

The number of processing real threads used by the `adbunarchivechunk` command for each step is automatically calculated from the *number of processing real threads to be used by the* `adbunarchivechunk` *command*. For details about the *number of processing real threads used by the* `adbunarchivechunk` *command*, see the description of the unarchive chunk option adb_unarcv_rthd_num in (2) Explanation of specification format in 43.2.2 Format of unarchive chunk options.

43.2 Specification format for the adbunarchivechunk command

This section explains the specification format for the `adbunarchivechunk` command and how to specify the unarchive chunk options.

The following OS users can execute the `adbunarchivechunk` command:

- HADB administrator
- OS users who belong to the HADB administrators group

43.2.1 Explanation of the specification format and options

(1) Specification format

```
adbunarchivechunk
  -u authorization-identifier
  [-p password]
  [-g output-interval-for-chunk-archive-processing-progress-messages]
  [-w {name-of-storage-directory-for-temporary-work-files
        |directory-path-file-name-used-to-specify-storage-destination-for-temporary-w
ork-files}]
  [-z unarchive-chunk-option-file-name]
  [-t]
  {-c chunk-ID-specification[,chunk-ID-specification]...|-r range-specification}
  [--force]
  table-to-be-processed
```

Important

You must specify the table to be processed as the last option that is specified.

(2) Explanation of options

● `-u` *authorization-identifier*

~<character string>((1 to 100 bytes))

Specifies the authorization identifier of the HADB user who executes the `adbunarchivechunk` command.

To execute the `adbunarchivechunk` command, the user must have the following two privileges:

- `CONNECT` privilege
- `UNARCHIVE CHUNK` privilege for the table to be processed

For this option, specify an authorization identifier in the range from 1 to 100 bytes. Note that the byte count (1 to 100 bytes) does not include the double quotation marks used to enclose the authorization identifier.

Important

If the character string used as the authorization identifier includes any lowercase letter or a backslash (`\`), make sure that you check the rules for specifying authorization identifiers. For the rules for specifying authorization identifiers, see 1.4.2 Rules for specifying authorization identifiers and passwords.

- `-p` *password*

  ~<character string>((1 to 255 bytes))

  Specifies the password for the authorization identifier that was specified in the `-u` option.

  > **Important**
  >
  > If the password includes a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), make sure that you check the rules for specifying passwords. For the rules for specifying passwords, see 1.4.2 Rules for specifying authorization identifiers and passwords.

  If the `adbunarchivechunk` command is executed with this option omitted, a message prompting the user to enter a password is displayed. In an environment in which a password cannot be entered from the standard input, such as when the command is executed in the background, make sure that you do not omit this option.

- `-g` *output-interval-for-chunk-unarchive-processing-progress-messages*

  ~<integer>((0 to 1,000,000,000))<<0>>(in 1,000 rows)

  Specifies the interval at which to output progress messages for chunk unarchive processing.

  For example, if `2` is specified in this option, the `KFAA80205-I` progress message is output each time 2,000 rows of chunk are unarchived.

  Furthermore, when chunk unarchive processing is performed on multiple chunks, progress messages are output for each chunk. For example, if `2` is specified in this option, and three chunks are undergoing chunk unarchive processing, the `KFAA80205-I` progress message is output each time 2,000 rows of data are unarchived for each of the three chunks. You can tell from the progress message which chunk's processing it refers to.

  If this option is omitted or `0` is specified in this option, no progress message is output.

- `-w` `{`*temporary-work-file-storage-directory-name* `|` *name-of-directory-path-file-specifying-temporary-work-file-storage-locations*`}`

  ~<OS path name>((2 to 518 bytes))

  Specifies a storage location for the temporary work files that are created during chunk unarchive processing, expressed as either of the following:

  1. Absolute path name of the storage directory for temporary work files
  2. Absolute path name of the file (directory path file) that specifies storage directories for temporary work files

  If frequent input/output operations are performed on the temporary work files stored under a single directory, performance might be adversely affected. We recommend that you use method 2 and store temporary work files in directories on multiple disks to distribute the workload of input/output operations on the disk. The following shows an example specification using method 2 (specifying a directory path file).

  Example of directory path file

  ```
  /mnt/diska/wwwww
  /mnt/diskb/xxxxx
  /mnt/diskc/yyyyy
  /mnt/diskd/zzzzz
  ```

  The following is a guideline for the number of directories to be specified:

  Guideline-for-number-of-directories-to-be-specified = *A* × *B*

  A: Number of data storage threads

  Use the following formula to determine the value:

  Value of the unarchive chunk option adb_unarcv_rthd_num -1

  B: Number of B-tree and text indexes that are defined for the target table of chunk unarchiving

If you do not know the number of B-tree and text indexes, check the number of B-tree and text indexes that are defined for the table by using the following SQL statement:

```
SELECT "N_INDEX"-"N_RANGE_INDEX"
    FROM "MASTER"."SQL_TABLES"
        WHERE "TABLE_SCHEMA"='authorization-identifier'
           AND "TABLE_NAME"='name-of-table-to-be-processed'
```

- A maximum of 255 directories can be specified in a directory path file.
- The directories specified in the directory path file are used to store temporary work files, in the order in which they are written in the file. When the last directory specified in the directory path file is used (in this example, `/mnt/diskd/zzzzz`), the first directory (in this example, `/mnt/diska/wwwww`) is then used again.

The following specification rule applies to this option (for both methods 1 and 2):

- Do not specify a forward slash (`/`) at the end of a path name.

For details about how to estimate the sizes of the temporary work files that will be created under the directories specified in the `-w` option, see *Estimating the size of the temporary work file for executing the adbunarchivechunk command* in *Estimating the size of the temporary work file for executing a command* in *Preparing Resources* in the *HADB Setup and Operation Guide*. If this option is omitted, temporary work files are created under `$DBDIR/ADBWORK`.

It is advisable to provide a dedicated directory to store temporary work files. If a directory storing other files is used for temporary work files, data in the existing files might be corrupted. Also make sure that symbolic links and relative paths are specified correctly. For details about the files whose names must differ from the temporary work file storage directory name, see 43.6 Notes.

If the disk that stores temporary work files runs out of free space, specify another disk with a larger capacity as the storage location. For details about the actions to take if the disk that stores temporary work files runs out of free space, see *Steps to take in the event of a shortage of disk space for storing temporary work files during command execution* in *Command-related problems* in *Troubleshooting* in the *HADB Setup and Operation Guide*.

● `-z` *unarchive-chunk-option-file-name*

~<OS path name>((2 to 510 bytes))

Specifies the absolute path name of the unarchive chunk option file that contains the unarchive chunk options. If this option is omitted, all unarchive chunk options are assumed to be their default values because no unarchive chunk option file is specified.

For details about the unarchive chunk options, see 43.2.2 Format of unarchive chunk options.

Make sure that the unarchive chunk option file name is unique, and not already taken by another file. If the unarchive option file has the same name as another file, data in the file might be lost. For details about the files whose names must differ from the unarchive chunk option file name, see 43.6 Notes.

● `-t`

Specify this option if you want to check the chunks to be unarchived before actually executing chunk unarchive processing.

If you specify this option, the chunk IDs of chunks to be unarchived among those corresponding to the range specified in the `-c` option or the `-r` option are output in the `KFAA80245-I` message.

To perform chunk unarchive processing on these chunks, do not change any options other than the `-t` option, and execute the `adbunarchivechunk` command with the `-t` option omitted. To revise which chunks are to be unarchived, correct the specified values in the `-c` option or the `-r` option, and then execute the `adbunarchivechunk` command.

> **Important**
>
> If you specify this option, chunk unarchive processing is not executed.

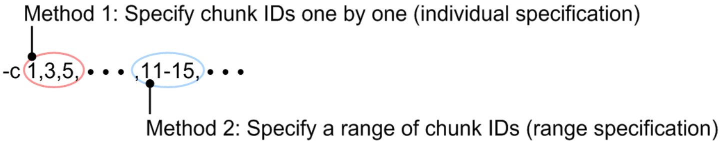
- `-c` *chunk-ID-specification*`[,`*chunk-ID-specification*`]...`

  ~<integer>((1 to 9,223,372,036,854,775,807))

  Specify the chunk IDs of chunks to be unarchived among those in the table to be processed. You can specify one or more chunk IDs.

  You can specify chunk IDs as follows. Note that you can combine multiple methods of specifying chunk IDs.

  Figure 43-3: Example of specifying chunk IDs in the -c option

  
  Method 1: Specify chunk IDs one by one (individual specification)
-c 1,3,5,• • • ,11-15,• • •
Method 2: Specify a range of chunk IDs (range specification)

  - **Specifying individual chunk IDs (individual specification)**

    If you want to specify the IDs of individual chunks to be unarchived, delimit the chunk IDs by using commas (`,`).

    > **Important**
    >
    > There must be no spaces before or after a comma (`,`).

  - **Specifying a range of chunk IDs (range specification)**

    If you want to specify a range of chunk IDs to be unarchived, use the format *smallest-chunk-ID-in-the-range*`-`*largest-chunk-ID-in-the-range* (connect the two chunk IDs with a hyphen (`-`)).

    > **Important**
    >
    > There must be no spaces before or after the hyphen (`-`). The chunk ID specified to the right of the hyphen must be greater than the chunk ID specified to the left of the hyphen.

  Among the chunks whose IDs are specified, the following chunks will not be unarchived (only chunks in the table to be processed that are subject to unarchiving will be unarchived):

  - Chunks that are not in the table to be processed
  - Chunks that are not archived

  The following rules apply to specifying this information:

  - You can specify maximum of 30,000 chunk IDs. The following explains how chunk IDs are counted:

    · For a range specification, all chunk IDs in the specified range are counted.

    • If any duplicate chunk IDs are specified, the duplicate chunk IDs are counted.

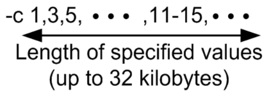
    Examples of specifying chunk IDs and how to count them

    Example 1: `-c 1,3,5`: The command assumes that three chunk IDs are specified (`1`, `3`, and `5`).

    Example 2: `-c 1,3,5-8,10`: The command assumes that seven chunk IDs are specified (`1`, `3`, `5`, `6`, `7`, `8`, and `10`).

    Example 3: `-c 1,3,5,5-8,10`: The command assumes that eight chunk IDs are specified (`1`, `3`, `5`, `5`, `6`, `7`, `8`, and `10`).

  - The maximum permitted length of the entire value specified in the `-c` option is 32 kilobytes. If the length would exceed 32 kilobytes, use range specifications or execute the `adbunarchivechunk` command more than once.



- For details about specifications that result in an error and specification examples, see (3) Errors that can be caused by the -c option.

▪ **Searching for chunk IDs of chunks that are in a table to be processed**

The following explains how to determine the chunk IDs of chunks that are in a table to be processed:

- Searching system tables for `STATUS_CHUNKS`

  Search system tables for `STATUS_CHUNKS`, and check the chunk IDs.

  For details, see *Checking the information about all chunks in a table based on a table name* in *Searching system tables* in *System Tables* in the *HADB Setup and Operation Guide*.

- Executing the `adbdbstatus` command

  Execute the following commands to output the DB area as well as the table and index usage information, and then check the chunk IDs (`Chunk_ID`) (the results are output in CSV format). For details about the items that are output as usage information for DB areas, tables, and indexes, see 13.7.2 List of items that are output as usage information for DB areas, tables, and indexes.

  ```
  adbdbstatus -d used -c table -n name-of-table-to-be-processed
  ```

  Specify the table name in the format *schema-name*.*table-identifier*.

● `-r` *range-specification*

Specify a range of chunks to be unarchived. For details about the `-r` option, see (4) Explanation of the -r option (range specification).

● `--force`

Normally, you will omit this option.

Specify this option in the following situation:

- When a re-executed `adbunarchivechunk` command terminates abnormally, the output error message's solution column clearly states that you must execute the `adbunarchivechunk` command with this option specified

An example of this sort of situation is when you cannot access the command status file needed for re-execution (file in the `$DBDIR/ADBSYS/ADBUTL` directory). If you execute the `adbunarchivechunk` command with this option specified, chunk unarchive processing is forcibly executed.

The procedure for executing the `adbunarchivechunk` command with this option specified is as follows:

1. Execute the `adbunarchivechunk` command with the `--force` option specified.
2. Delete the temporary work files based on the explanation in the following topic in the *HADB Setup and Operation Guide*: *When there are unneeded temporary work files on the disk* in *Steps to take in the event of a shortage of disk space for storing temporary work files during command execution*

● *table-to-be-processed*

Specify the target archivable multi-chunk table for chunk unarchive processing.

The following rules apply to specifying this information:

- Specify in the format *schema-name*.*table-identifier*.

  If the user owns this table, the schema name can be omitted. If another HADB user owns this table, schema name must be specified.

- For details about the specification rules that apply when a schema name or table identifier includes any lowercase letter or backslash (\), see 1.4.3 Table name specification rules.

- Tables other than archivable multi-chunk tables cannot be specified.
- The user can specify only a table for which the user has the `UNARCHIVE CHUNK` privilege.
- Specify a table that can be updated (that is not non-updatable). Specifying a non-updatable table results in an error.
- A viewed table cannot be specified.

(3) Errors that can be caused by the -c option

The following are two possible causes of errors in the `adbunarchivechunk` command that can be caused by the `-c` option's specification:

- The specification format of the `-c` option is invalid.
- An invalid chunk ID is specified in the `-c` option.

In either case, correct the specification, and then re-execute the `adbunarchivechunk` command. For details about the `-c` option, see -c in (2) Explanation of options.

(a) When the specification format of the -c option is invalid

The following table lists the message IDs that are displayed when the specification format of the `-c` option is invalid, and describes the possible errors. If any of these errors occurs, correct the error, and then re-execute the `adbunarchivechunk` command.

Table 43-1: Message IDs that are displayed when the specification format of the -c option is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	`KFAA50225-E`	The number of specified chunk IDs exceeds 30,000.	`-c 1-30001`	`-c 1-30000`
2		The same chunk ID is specified on both sides of a hyphen (-).	`-c 3-3,7`	`-c 3-5,7`
3		The value of the chunk ID on the right side of a hyphen (-) is less than the value of the chunk ID on the left side.	`-c 10-3`	`-c 3-10`
4	• `KFAA90002-E` • `KFAA96840-I`	There is a space between a chunk ID and a comma.	`-c 1,3,Δ10`	`-c 1,3,10`
5		There is a space between a chunk ID and a hyphen (-).	`-c 1,3Δ-Δ10`	`-c 1,3-10`
6	`KFAA90003-E`	The length of the specified value exceeds 32 kilobytes.	-c 1,2, ••• ,xxxxx ↔ 33 kilobytes or more	`-c 1-`*xxxxx* Change from individual specification to range specification (use a hyphen (-), not a comma (,)).

Legend:

Δ: Space

(b) When an invalid chunk ID is specified in the -c option

The following table lists the message IDs that are displayed when an invalid chunk ID is specified in the `-c` option, and describes the possible errors. If an error occurs, correct the error, and then re-execute the `adbunarchivechunk` command.

Table 43-2: Message IDs that are displayed when an invalid chunk ID is specified in the -c option, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification and corrective action to take
1	`KFAA50284-E`	None of the specified chunk IDs is in the table to be processed.	**If only chunk IDs 2 and 6 are in the table to be processed** • `-c 3-5` • `-c 4,5`	**If only chunk IDs 2 and 6 are in the table to be processed** • `-c 2,6` • `-c 2-6` Specify the chunk IDs of archived chunks that are in the table to be processed.

Note:

For details about how to search for the IDs of chunks existing in the table to be processed, see -c in (2) Explanation of options.

(4) Explanation of the -r option (range specification)

Use the `-r` option to specify a range of chunks to be unarchived. If the specified range contains any part of the range of values in the archive range column (the column used to narrow down the search range when searching the archivable multi-chunk table), the corresponding chunks are unarchived. However, among the chunks where data in the specified range is stored, chunks that are not archived are not unarchived.

The range specification is interpreted based on the data type in the archive range column.

The following table lists the types of range specifications and links to their detailed explanations.

Table 43-3: Types of range specifications and their links

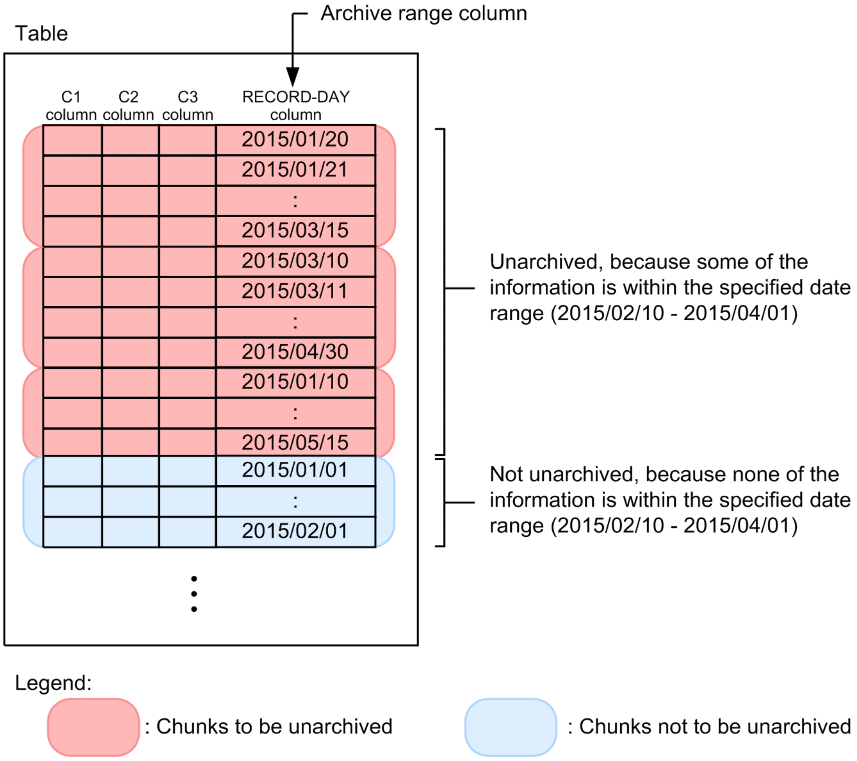
No.	Type of range specification	Link
1	Date range specification	(a) Specifying ranges of dates
2	Time range specification	(b) Specifying ranges of times
3	Time stamp range specification	(c) Specifying ranges of time stamps
4	Fixed-length character string range specification	(d) Specifying ranges of fixed-length character strings
5	Decimal range specification	(e) Specifying ranges of decimal numbers
6	Integer range specification	(f) Specifying ranges of integers
7	Floating point range specification	(g) Specifying ranges of floating point numbers

This section shows which chunks are unarchived, using the following conditions as an example:

- The data type in the archive range column is `DATE`.

- The date range `2015/02/10` - `2015/04/01` is specified in the range specification.

Figure 43-4: Relationship between the content of the date range specification and the chunks to be unarchived


Archive range column
Table
C1 column
C2 column
C3 column
RECORD-DAY column
2015/01/20
2015/01/21
:
2015/03/15
2015/03/10
2015/03/11
:
2015/04/30
2015/01/10
:
2015/05/15
2015/01/01
:
2015/02/01
Unarchived, because some of the information is within the specified date range (2015/02/10 - 2015/04/01)
Not unarchived, because none of the information is within the specified date range (2015/02/10 - 2015/04/01)
Legend:
: Chunks to be unarchived
: Chunks not to be unarchived

The rules for specifying the `-r` option are as follows:

- You can only specify one range. Multiple ranges cannot be specified.
- If 1,024 bytes or more are specified in the range specification, the `KFAA90003-E` message is output.
- In this option, specify the value that results after characters such as shell enclosing characters have been parsed. The specified value and the actual interpreted value might differ depending on how the shell parses the characters. The following is an example of a fixed-length character string range specification.

  Option specification value: `-r 'abc'-'def'`

  Actual interpreted value: `-r abc-def`

  Single quotation marks (`'`) have been omitted from the actual interpreted value.

(a) Specifying ranges of dates

Use dates to specify a range of chunks to be unarchived.

Format

```
-r {YYYY-MM-DD|YYYY/MM/DD}-{YYYY-MM-DD|YYYY/MM/DD}
```

Connect the minimum value (date specification) and maximum value (date specification) by using a hyphen (-). There must be no spaces between the minimum value and the hyphen (-), or between the maximum value and the hyphen (-).

Explanation

YYYY: Year (`0001-9999`)

MM: Month (`01`-`12`)

DD: Day (`01` to the last day of the month specified in *MM*)

Specification example

Specification example 1: `-r 2016/01/01-2016/12/31`

Specification example 2: `-r 2016-01-01-2016-12-31`

Specify each date in either of the following formats: *YYYY*`-`*MM*`-`*DD* or *YYYY*`/`*MM*`/`*DD* (delimit *YYYY*, *MM*, and *DD* using forward slashes (`/`) or hyphens (`-`)).

In the case of specification example 1, `2016/01/01` and `2016/12/31` are treated as date specifications.

In the case of specification example 2, `2016-01-01` and `2016-12-31` are treated as date specifications.

The rules for specifying ranges of dates are as follows:

- If there are not enough digits in the values you want to specify in *YYYY*, *MM*, or *DD*, pad the left side of the value with zeros (`0`).
- The data types in the archive range column for which you can specify a date range are as follows:
  - `DATE`
  - `TIMESTAMP`
- Only dates can be specified in a date range specification. If the data type in the archive range column is `TIMESTAMP`, specified values are treated as if the time has been omitted. Therefore, the start time and end time in a specified value are corrected as shown in the following table. As a result, all chunks containing data in the specified date range are subject to unarchiving.

Table 43-4: Correction of the start time and end time

No.	Number of digits in the fractional seconds defined in the TIMESTAMP-type column	Start time	End time
1	`0`	0 hours 0 minutes 0 seconds	23 hours 59 minutes 59 seconds
2	`3`	0 hours 0 minutes 0.000 seconds	23 hours 59 minutes 59.999 seconds
3	`6`	0 hours 0 minutes 0.000000 seconds	23 hours 59 minutes 59.999999 seconds
4	`9`	0 hours 0 minutes 0.000000000 seconds	23 hours 59 minutes 59.999999999 seconds
5	`12`	0 hours 0 minutes 0.000000000000 seconds	23 hours 59 minutes 59.999999999999 seconds

- The following table lists the message IDs that are displayed when the specified date range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then re-execute the `adbunarchivechunk` command.

Table 43-5: Message IDs that are displayed when the specified date range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	`KFAA50225-E`	• Too many digits in the date • Not enough digits in the date	`-r 2015/7/1-2015/008/031`	`-r 2015/07/01-2015/08/31`

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
2		A date outside the range is specified.	`-r 2015/01/01-2015/06/31`	`-r 2015/01/01-2015/06/30`
3	• `KFAA90002-E` • `KFAA96840-I`	The range specification format is invalid.	**Specification example** `-r 2015/01/11Δ-Δ2015/7/31` **Explanation** Because there is a space between the date specification and the hyphen (`-`), the shell erroneously uses spaces to delimit the option arguments.	`-r 2015/01/11-2015/07/31`

Legend:

Δ: Space

(b) Specifying ranges of times

Use times to specify a range of chunks to be unarchived.

Format

```
-r hh:mm:ss[.[nn...n]]-hh:mm:ss[.[nn...n]]
```

Connect the minimum value (time specification) and maximum value (time specification) by using a hyphen (`-`). There must be no spaces between the minimum value and the hyphen (`-`), or between the maximum value and the hyphen (`-`).

Explanation

hh:: Hours (`00`-`23`)

mm: Minutes (`00`-`59`)

ss: Seconds (`00`-`59`)

`.`: Decimal point

nn...n: Fractional seconds (*n* is `0`-`9`)

Specification example

Specification example 1: `-r 00:00:00-12:59:59`

Specification example 2: `-r 00:00:00.000-12:59:59.999`

The *hh*, *mm*, and *ss* values of each time are delimited using colons (`:`).

When specifying fractional seconds, connect the *ss* and *nn...n* values by using a decimal point (`.`).

In the case of specification example 1, `00:00:00` and `12:59:59` are treated as time specifications.

In the case of specification example 2, `00:00:00.000` and `12:59:59.999` are treated as time specifications.

The rules for specifying ranges of times are as follows:

- If there are not enough digits in the values you want to specify in *hh*, *mm*, or *ss*, pad the left side of the value with zeros (`0`).
- The data type in the archive range column for which you can specify a time range is as follows:
    - `TIME`

- If the fractional seconds have more digits than the fractional seconds defined in the `TIME`-type column, the excess digits are truncated.
- If fractional seconds are omitted, the start time and end time are corrected to reflect the number of digits in the fractional seconds defined in the `TIME`-type column. The following table lists details about how this correction is made.

Table 43-6: Correction of the start time and end time when fractional seconds are omitted

No.	Number of digits in the fractional seconds defined in the TIME-type column	Start time	End time
1	3	.000 seconds	.999 seconds
2	6	.000000 seconds	.999999 seconds
3	9	.000000000 seconds	.999999999 seconds
4	12	.000000000000 seconds	.999999999999 seconds

- For details about comparing the minimum and maximum values for time specifications, see *Comparing datetime data* in *Data types that can be compared* in *Data types that can be converted, assigned, and compared* in the manual *HADB SQL Reference*.
- The following table lists the message IDs that are displayed when the specified time range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then re-execute the `adbunarchivechunk` command.

Table 43-7: Message IDs that are displayed when the specified time range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	`KFAA50225-E`	• Too many digits in the time • Not enough digits in the time	`-r 0:0:0-012:59:59`	`-r 00:00:00-12:59:59`
2		A time outside the range is specified.	`-r 00:00:00-12:60:00`	`-r 00:00:00-12:59:59`
3	• `KFAA90002-E` • `KFAA96840-I`	The range specification format is invalid.	**Specification example** `-r 00:00:00Δ-Δ12:59:59` **Explanation** Because there is a space between the time specification and the hyphen (`-`), the shell erroneously uses spaces to delimit the option arguments.	`-r 00:00:00-12:59:59`

Legend:

Δ: Space

(c) Specifying ranges of time stamps

Use time stamps to specify a range of chunks to be unarchived.

Format

```
-r '{YYYY-MM-DD|YYYY/MM/DD}Δhh:mm:ss[.[nn...n]]'
    -'{YYYY-MM-DD|YYYY/MM/DD}Δhh:mm:ss[.[nn...n]]'
```

Connect the minimum value (time stamp specification) and maximum value (time stamp specification) by using a hyphen (-). There must be no spaces between the minimum value and the hyphen (-), or between the maximum value and the hyphen (-).

Explanation

YYYY: Year (0001-9999)

MM: Month (01-12)

DD: Day (01 to the last day of the month specified in *MM*)

Δ: Halfwidth space or tab

hh:: Hours (00-23)

mm: Minutes (00-59)

ss: Seconds (00-59)

.: Decimal point

nn...n: Fractional seconds (*n* is 0-9)

Specification example

Specification example 1: `-r '2016/01/01 00:00:00'-'2016/12/31 12:59:59'`

Specification example 2: `-r '2016/01/01 00:00:00.000'-'2016/12/31 12:59:59.999'`

Connect the date and time of each specified time stamp by using a halfwidth space, and enclose entire date or time values in single quotation marks ('). When specifying dates, use either of the following formats: *YYYY-MM-DD* or *YYYY/MM/DD* (delimit *YYYY*, *MM*, and *DD* using forward slashes (/) or hyphens (-)). When specifying times, delimit the *hh*, *mm*, and *ss* values by using colons (:). When specifying fractional seconds, connect the *ss* and *nn...n* values by using a decimal point (.).

In the case of specification example 1, 2016/01/01 00:00:00 and 2016/12/31 12:59:59 are treated as time stamp specifications.

In the case of specification example 2, 2016/01/01 00:00:00.000 and 2016/12/31 12:59:59.999 are treated as time stamp specifications.

The rules for specifying ranges of time stamps are as follows:

- If there are not enough digits in the values you want to specify in *YYYY*, *MM*, *DD*, *hh*, *mm*, or *ss*, pad the left side of the value with zeros (0).
- The data type in the archive range column for which you can specify a time stamp range is as follows:
  - TIMESTAMP
- If the fractional seconds have more digits than the fractional seconds defined in the TIMESTAMP-type column, the excess digits are truncated.
- If fractional seconds are omitted, the start time and end time are corrected to reflect the number of digits in the fractional seconds defined in the TIMESTAMP-type column. The following table lists details about how this correction is made.

Table 43-8: Correction of the start time and end time when fractional seconds are omitted

No.	Number of digits in the fractional seconds defined in the TIMESTAMP-type column	Start time	End time
1	3	.000 seconds	.999 seconds
2	6	.000000 seconds	.999999 seconds
3	9	.000000000 seconds	.999999999 seconds

No.	Number of digits in the fractional seconds defined in the TIMESTAMP-type column	Start time	End time
4	`12`	.000000000000 seconds	.999999999999 seconds

- For details about comparing the minimum and maximum values for time stamp specifications, see *Comparing datetime data* in *Data types that can be compared* in *Data types that can be converted, assigned, and compared* in the manual *HADB SQL Reference*.
- The following table lists the message IDs that are displayed when the specified time stamp range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then re-execute the `adbunarchivechunk` command.

Table 43-9: Message IDs that are displayed when the specified time stamp range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	`KFAA50225-E`	• Too many digits in the time stamp • Not enough digits in the time stamp	`-r '2015/7/1Δ00:00:00'-'2015/008/031Δ12:59:59'`	`-r '2015/07/01Δ00:00:00'-'2015/08/31Δ12:59:59'`
2		A time stamp outside the range is specified.	`-r '2015/01/01Δ00:00:00'-'2015/06/31Δ12:60:00'`	`-r '2015/01/01Δ00:00:00'-'2015/06/30Δ12:59:59'`
3	• `KFAA90002-E` • `KFAA96840-I`	The range specification format is invalid.	**Specification example** `-r '2016/04/01Δ00:00:00'Δ-Δ'2016/06/30Δ12:59:59'` **Explanation** Because there is a space between the time stamp specification and the hyphen (-), the shell erroneously uses spaces to delimit the option arguments.	`-r '2016/04/01Δ00:00:00'-'2016/06/30Δ12:59:59'`
4			**Specification example** `-r 2016/04/01Δ00:00:00-2016/06/30Δ12:59:59` **Explanation** Because the time stamp specification is not enclosed in single quotation marks ('), the shell sees the space between the date and time values, and erroneously uses spaces to delimit the option arguments.	

Legend:

Δ: Space

(d) Specifying ranges of fixed-length character strings

Use fixed-length character strings to specify a range of chunks to be unarchived.

Format

```
-r a...a-a...a
```

Connect the minimum value (fixed-length character string specification) and maximum value (fixed-length character string specification) by using a hyphen (`-`). There must be no spaces between the minimum value and the hyphen (`-`), or between the maximum value and the hyphen (`-`).

Explanation

a...a: String of one or more characters

Specification example

`-r 20160101-20161231`

`20160101` and `20161231` are treated as fixed-length character string specifications.

The rules for specifying ranges of fixed-length character strings are as follows:

- Specify a fixed-length character string of one or more characters.
- Fixed-length character string specifications are case sensitive.
- The data type in the archive range column for which you can specify a fixed-length character string range is as follows:
  - `CHARACTER`
- If the fixed-length character string includes a character that has a special meaning in the OS or shell, such as a double quotation mark (`"`) or vertical bar (`|`), specify the escape character (`\`) immediately before that character. Alternatively, enclose the entire fixed-length character string in single quotation marks (`'`).

  Specification example

  Specification example 1: `-r a\|c-def`

  Specification example 2: `-r 'a|c'-def`

  In both examples, `a|c` and `def` are treated as fixed-length character string specifications.
- If a character string is enclosed in double quotation marks (`"`) and then the entire string is enclosed in single quotation marks (`'`), both the character string and the double quotation marks (`"`) are treated as a fixed-length character string.

  Specification example

  `-r '"abc"'-def`

  `"abc"` and `def` are treated as fixed-length character string specifications.
- If you specify a fixed-length character string that includes any hyphens (`-`), enclose the entire fixed-length character string in single quotation marks (`'`). This is to differentiate hyphens (`-`) in the specification from hyphens used to separate the minimum and maximum values.

  Specification example

  - When enclosing a specification in `\'` (backslashes and single quotation marks)

    `-r \'a-c\'-def`
  - When first enclosing a specification in single quotation marks (`'`), and then enclosing the entire string in double quotation marks (`"`)

    `-r "'a-c'"-def`

  In both examples, `a-c` and `def` are treated as fixed-length character string specifications.

  In addition, even if a specified fixed-length character string does not include any hyphens (`-`), if you enclose the specification in single quotation marks (`'`), the enclosed value is treated as the fixed-length character string specification, with the single quotation marks (`'`) omitted.

Specification example

- When enclosing a specification in `\'` (backslashes and single quotation marks)

  `-r \'abc\'-def`

  `abc` and `def` are treated as fixed-length character string specifications.

- If a specified fixed-length character string includes any single quotation marks (`'`), specify two single quotation marks (`'`) in succession to express one single quotation mark (`'`).

  Specification example

  - When specifying two `\'` (backslashes and single quotation marks)

    `-r a\'\'c-def`

  - When enclosing a specification that includes a single quotation mark (`'`), in double quotation marks (`"`)

    `-r "a''c"-def`

  In both examples, `a'c` and `def` are treated as fixed-length character string specifications.

- For details about comparing the minimum and maximum values for fixed-length character string specifications, see *Comparing character string data* in *Data types that can be compared* in *Data types that can be converted, assigned, and compared* in the manual *HADB SQL Reference*.

- The following table lists the message IDs that are displayed when the specified fixed-length character string range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then re-execute the `adbunarchivechunk` command.

Table 43-10: Message IDs that are displayed when the specified fixed-length character string range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification

1	`KFAA50225-E`	The fixed-length character string specification includes multiple hyphens (`-`) that are used to separate the minimum and maximum values.	`-r a-c-def`
			`-r 'a-c-def'`
				`-r \'a-c\'-def`

2	• `KFAA90002-E` • `KFAA96840-I`	The range specification format is invalid.	**Specification example** `-r abcΔ-Δdef` **Explanation** Because there is a space between the fixed-length character string specification and the hyphen (`-`), the shell erroneously uses spaces to delimit the option arguments.	`-r abc-def`

Legend:

Δ: Space

(e) Specifying ranges of decimal numbers

Use decimal numbers to specify a range of chunks to be unarchived.

Format

```
-r [{±|-}]{a...a[.[b...b]]|.b...b}
     -[{±|-}]{a...a[.[b...b]]|.b...b}
```

Connect the minimum value (decimal specification) and maximum value (decimal specification) by using a hyphen (-). There must be no spaces between the minimum value and the hyphen (-), or between the maximum value and the hyphen (-).

Explanation

+, -: Signs (the "+" sign can be omitted)

a...a: Integer part (*a* is 0-9)

.: Decimal point

b...b: Fractional part (*b* is 0-9)

Examples of specifying DECIMAL(5,2)

Specification example 1: `-r -100-100`

Specification example 2: `-r -100.00-+100.00`

`-100.00` and `+100.00` are treated as decimal specifications.

The rules for specifying ranges of decimal numbers are as follows:

- Either the integer part or the fractional part must be specified. If there is no fractional part, the decimal point can be omitted.
- The data type in the archive range column for which you can specify a decimal range is as follows:
  - DECIMAL
- If the fractional parts have more digits than the fractional parts defined in the DECIMAL-type column, the excess digits are truncated.
- The following table lists the message IDs that are displayed when the specified decimal range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then re-execute the adbunarchivechunk command.

Table 43-11: Message IDs that are displayed when the specified decimal range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	KFAA50225-E	A numerical value has been specified that cannot be used with the data type in the archive range column.	**Specification example** -r 0-123456789012345678901 234567890123456789 **Explanation** A decimal specification contains 39 digits.	**Specification example** -r 0-123456789012345678901 23456789012345678 **Explanation** Specify a decimal using 38 digits.
2	• KFAA90002-E • KFAA96840-I	The range specification format is invalid.	**Specification example** -r 0Δ-Δ1 **Explanation** Because there is a space between the decimal specification and the hyphen (-), the shell erroneously uses spaces to delimit the option arguments.	-r 0-1

Legend:

Δ: Space

(f) Specifying ranges of integers

Use integers to specify a range of chunks to be unarchived.

Format

```
-r [{+|-}]a...a-[{+|-}]a...a
```

Connect the minimum value (integer specification) and maximum value (integer specification) by using a hyphen (-). There must be no spaces between the minimum value and the hyphen (-), or between the maximum value and the hyphen (-).

Explanation

+, -: Signs (the "+" sign can be omitted)

a...a: Numerical value (*a* is 0-9)

Specification example

Specification example 1: `-r -100-100`

Specification example 2: `-r -100-+100`

`-100` and `+100` are treated as integer specifications.

The rules for specifying ranges of integers are as follows:

- The data types in the archive range column for which you can specify an integer range are as follows:
  - `INTEGER`
  - `SMALLINT`
- The following table lists the message IDs that are displayed when the specified integer range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then re-execute the `adbunarchivechunk` command.

Table 43-12: Message IDs that are displayed when the specified integer range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	KFAA50225-E	A numerical value has been specified that cannot be used with the data type in the archive range column.	**In the case of INTEGER** -r 0-9223372036854775808 **In the case of SMALLINT** -r 0-2147483648	**In the case of INTEGER** -r 0-9223372036854775807 **In the case of SMALLINT** -r 0-2147483647
2	• KFAA90002-E • KFAA96840-I	The range specification format is invalid.	**Specification example** -r 0Δ-Δ1 **Explanation** Because there is a space between the integer specification and the hyphen (-), the shell erroneously uses spaces to delimit the option arguments.	-r 0-1

Legend:

Δ: Space

(g) Specifying ranges of floating point numbers

Use floating point numbers to specify a range of chunks to be unarchived.

Format

```
-r [{+|-}]{a...a[.[b...b]]|.b...b}[{E|e}[{+|-}]c...c]
    - [{+|-}]{a...a[.[b...b]]|.b...b}[{E|e}[{+|-}]c...c]
```

Connect the minimum value (floating point specification) and maximum value (floating point specification) by using a hyphen (`-`). There must be no spaces between the minimum value and the hyphen (`-`), or between the maximum value and the hyphen (`-`).

Explanation

`+`, `-`: Signs (the "`+`" sign can be omitted)

a...a: Integer part of the mantissa (*a* is `0`-`9`)

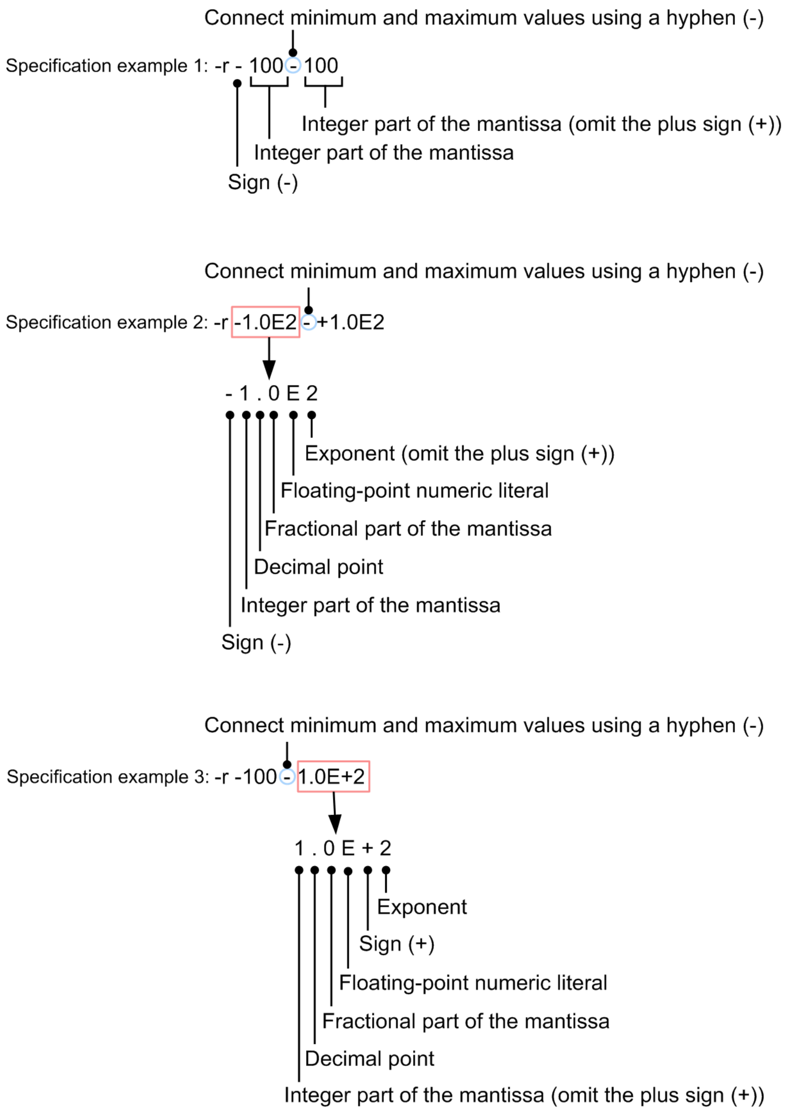
`.`: Decimal point

b...b: Fractional part of the mantissa (*b* is `0`-`9`)

`E`, `e`: Floating-point numeric literal

c...c: Exponent (*c* is `0`-`9`)

Figure 43-5: Example of specifying a floating point range by using the -r option


Connect minimum and maximum values using a hyphen (-)
Specification example 1: -r - 100-100
Integer part of the mantissa (omit the plus sign (+))
Integer part of the mantissa
Sign (-)
Connect minimum and maximum values using a hyphen (-)
Specification example 2: -r -1.0E2-+1.0E2
- 1 . 0 E 2
Exponent (omit the plus sign (+))
Floating-point numeric literal
Fractional part of the mantissa
Decimal point
Integer part of the mantissa
Sign (-)
Connect minimum and maximum values using a hyphen (-)
Specification example 3: -r -100-1.0E+2
1 . 0 E + 2
Exponent
Sign (+)
Floating-point numeric literal
Fractional part of the mantissa
Decimal point
Integer part of the mantissa (omit the plus sign (+))

In the case of specification example 1, `-100` and `100` are treated as floating point specifications.

In the case of specification example 2, `-1.0E2` and `+1.0E2` are treated as floating point specifications.

In the case of specification example 3, `-100` and `1.0E+2` are treated as floating point specifications.

The rules for specifying ranges of floating point numbers are as follows:

- The data type in the archive range column for which you can specify a floating point range is as follows:
  - `DOUBLE PRECISION`
- The following table lists the message IDs that are displayed when the specified floating point range is invalid, and describes possible errors. If any of these errors occurs, correct the error, and then re-execute the `adbunarchivechunk` command.

Table 43-13: Message IDs that are displayed when the specified floating point range is invalid, and possible errors

No.	Message ID that is displayed	Possible error	Example of invalid specification	Example of corrected specification
1	`KFAA50225-E`	The number of specifiable characters is exceeded.	**Specification example** `-r 0-123...(omitted)...789` **Explanation** A floating point specification contains 510 characters.	**Specification example** `-r 0-123...(omitted)...78` **Explanation** Specify a floating point specification using 509 characters.
2	• `KFAA90002-E` • `KFAA96840-I`	The range specification format is invalid.	**Specification example** `-r 0Δ-Δ1` **Explanation** Because there is a space between the floating point specification and the hyphen (-), the shell erroneously uses spaces to delimit the option arguments.	`-r 0-1`

Legend:

Δ: Space

43.2.2 Format of unarchive chunk options

(1) Specification format

```
[set adb_unarcv_rthd_num = number-of-processing-real-threads-to-be-used-by-adbunarchi
vechunk-command]
[set adb_unarcv_read_size = buffer-size]
[set adb_unarcv_buff_blk_num = number-of-buffer-blocks-for-unarchiving-chunks]
[set adb_unarcv_sort_buff_size = sort-buffer-size]
[set adb_unarcv_dividx_rd_buff_size = B-tree-and-text-index-input-buffer-size]
[set adb_unarcv_dividx_wt_buff_size = B-tree-and-text-index-output-buffer-size]
[set adb_unarcv_txt_buff_size = buffer-size-for-creating-text-indexes]
[set adb_unarcv_decompress_buff_size = buffer-size-for-decompressing-archive-files]
```

The rules for specifying these unarchive chunk options are the same as for the server definition. For the rules for specifying server definitions, see the topic *Syntax rules for the server definition* in the *HADB Setup and Operation Guide*.

(2) Explanation of specification format

- adb_unarcv_rthd_num = *number-of-processing-real-threads-to-be-used-by-adbunarchivechunk-command*

  ~<integer>((2 to 4,096))

  Specifies the number of processing real threads to be used for executing the adbunarchivechunk command.

  Specification of this option is recommended. The adbunarchivechunk command uses the number of processing real threads specified in this option. Therefore, specify an appropriate number of processing real threads for use by the adbunarchivechunk command. For details about the processing of the adbunarchivechunk command that uses processing real threads and the number of processing real threads to be used for each step of processing, see 43.1.2 Relationships among threads used by the adbunarchivechunk command.

  If this option is omitted, the number of processing real threads to be used for executing the adbunarchivechunk command is determined by the adb_sql_exe_max_rthd_num operand in the server definition. The following table shows the relationship between the value of this option or the adb_sql_exe_max_rthd_num operand in the server definition and the number of processing real threads used for executing the command.

Table 43-14: Number of processing real threads used for executing the command

No.	Value in this option or in the server definition		Number of processing real threads used for executing the command
1	When this option is omitted	If the adb_sql_exe_max_rthd_num operand in the server definition is not specified	The default value of the adb_sql_exe_max_rthd_num operand in the server definition is used.
2		If 0 or 1 is specified in the adb_sql_exe_max_rthd_num operand in the server definition	--
3		If 2 or a greater value is specified for the adb_sql_exe_max_rthd_num operand in the server definition	The value specified for the adb_sql_exe_max_rthd_num operand in the server definition is used.
4	If 2 or a greater value is specified in this option		The value specified in this option is used.

Legend:

--: The adbunarchivechunk command results in an error.

Note

For details about the adb_sql_exe_max_rthd_num operand in the server definition, see *Operands related to performance (set format)* in the *HADB Setup and Operation Guide*.

- adb_unarcv_read_size = *buffer-size*

  ~<integer>((32 to 51,200))<<1,024>> (kilobytes)

  The buffer size used for the read processing of archive files is specified in kilobytes.

- adb_unarcv_buff_blk_num = *number-of-buffer-blocks-for-unarchiving-chunks*

  ~<integer>((5 to 4,096))<<64>>

  Specifies the number of buffer blocks to be used for chunk unarchive processing.

  There is generally no problem with using the default values, but increasing the value specified for this option might improve the performance of chunk unarchive processing.

- adb_unarcv_sort_buff_size = *sort-buffer-size*

  ~<integer>((1 to 2,048))<<256>> (megabytes)

  Consider specifying this option when B-tree or text indexes are defined for the table to be processed.

  Specify the size (in megabytes) of the sort buffer to be used for creating B-tree and text indexes. Specify a large value if there is sufficient memory. Better performance can be expected as this option's value becomes larger.

- `adb_unarcv_dividx_rd_buff_size` = *B-tree-and-text-index-input-buffer-size*

  ~<integer>((32 to 51,200))<<1,024>> (kilobytes)

  Consider specifying this option if B-tree or text indexes are defined for the table to be processed.

  Specify the size (in kilobytes) of the input buffer to be used for creating B-tree and text indexes. Specify a large value if there is sufficient memory. Better performance can be expected as this option's value becomes larger.

- `adb_unarcv_dividx_wt_buff_size` = *B-tree-and-text-index-output-buffer-size*

  ~<integer>((32 to 51,200))<<1,024>> (kilobytes)

  Consider specifying this option if B-tree or text indexes are defined for the table to be processed.

  Specify the size (in kilobytes) of the output buffer to be used for creating B-tree and text indexes. Specify a large value if there is sufficient memory. Better performance can be expected as this option's value becomes larger.

- `adb_unarcv_txt_buff_size` = *buffer-size-for-creating-text-indexes*

  ~<integer>((1 to 2,048))<<256>> (megabytes)

  Consider specifying this option if text indexes are defined for the table to be processed.

  Specify the size (in megabytes) of the buffer to be used for creating text indexes.

  Increasing the value specified for this option often improves performance. However, the amount of buffer memory specified by this option will be used for all processing real threads on which text indexes will be built. Therefore, be careful when you increase the value specified for this option.

- `adb_unarcv_decompress_buff_size` = *buffer-size-for-decompressing-archive-files*

  ~<integer>((32 to 51,200))<<1,024>> (kilobytes)

  Specify the size (in kilobytes) of the buffer used for decompressing archive file data.

  When archive file data is decompressed, only the buffer amount specified in this option is used.

  Use the value specified in the unarchive chunk option adb_unarcv_read_size as a guideline for specifying this option.

43.3 Examples

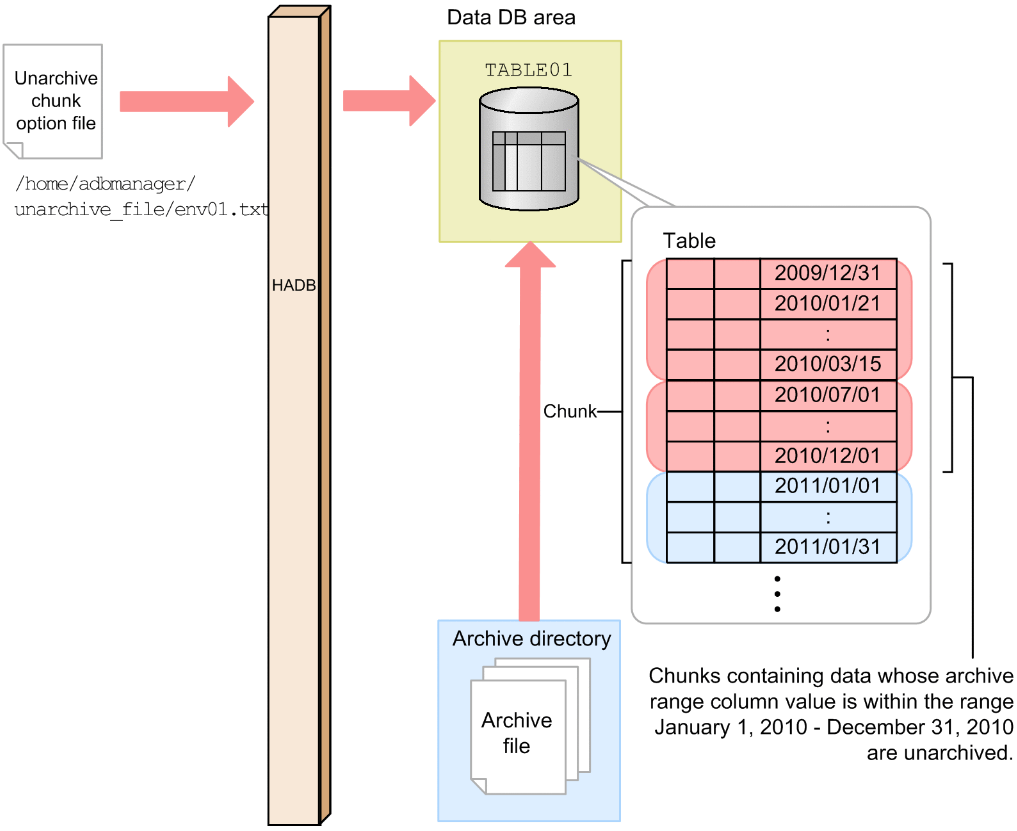
This section explains by way of examples how to use the adbunarchivechunk command to unarchive chunks.

If the adbunarchivechunk command terminates abnormally during execution, take appropriate action based on the explanation in 43.5 Handling abnormal termination of the adbunarchivechunk command.

43.3.1 When unarchiving chunks that include specific date range data

Unarchive chunks that were created in a table, that include specific date range data, under the following conditions:

- The base table TABLE01 is an archivable multi-chunk table.
- The data type in the archive range column of the base table TABLE01 is DATE.
- Among the chunks created in the base table TABLE01, unarchive chunks whose values in the archive range column contain data in the range from January 1st, 2010 to December 31st, 2010.


Unarchive chunk option file
/home/adbmanager/unarchive_file/env01.txt
HADB
Data DB area
TABLE01
Table
2009/12/31
2010/01/21
:
2010/03/15
2010/07/01
:
2010/12/01
2011/01/01
:
2011/01/31
Chunk
Archive directory
Archive file
Chunks containing data whose archive range column value is within the range January 1, 2010 - December 31, 2010 are unarchived.

This example uses the following procedure to execute the adbunarchivechunk command.

Procedure

1. Create an unarchive chunk option file.
2. Execute the adbunarchivechunk command.

Using examples, the following subsections explain each of these steps.

(1) Creating an unarchive chunk option file

Create an unarchive chunk option file, and specify unarchive chunk options in that file. In this example, the number of processing real threads used for executing the adbunarchivechunk command is specified.

Example of specifying an unarchive chunk option file

```
set adb_unarcv_rthd_num = 2
      :
```

(2) Executing the adbunarchivechunk command

```
adbunarchivechunk -u ADBUSER01                                  ...1
                  -p '#HelloHADB_01'                            ...2
                  -z /home/adbmanager/unarchive_file/env01.txt  ...3
                  -r 2010/01/01-2010/12/31                      ...4
                  TABLE01                                       ...5
```

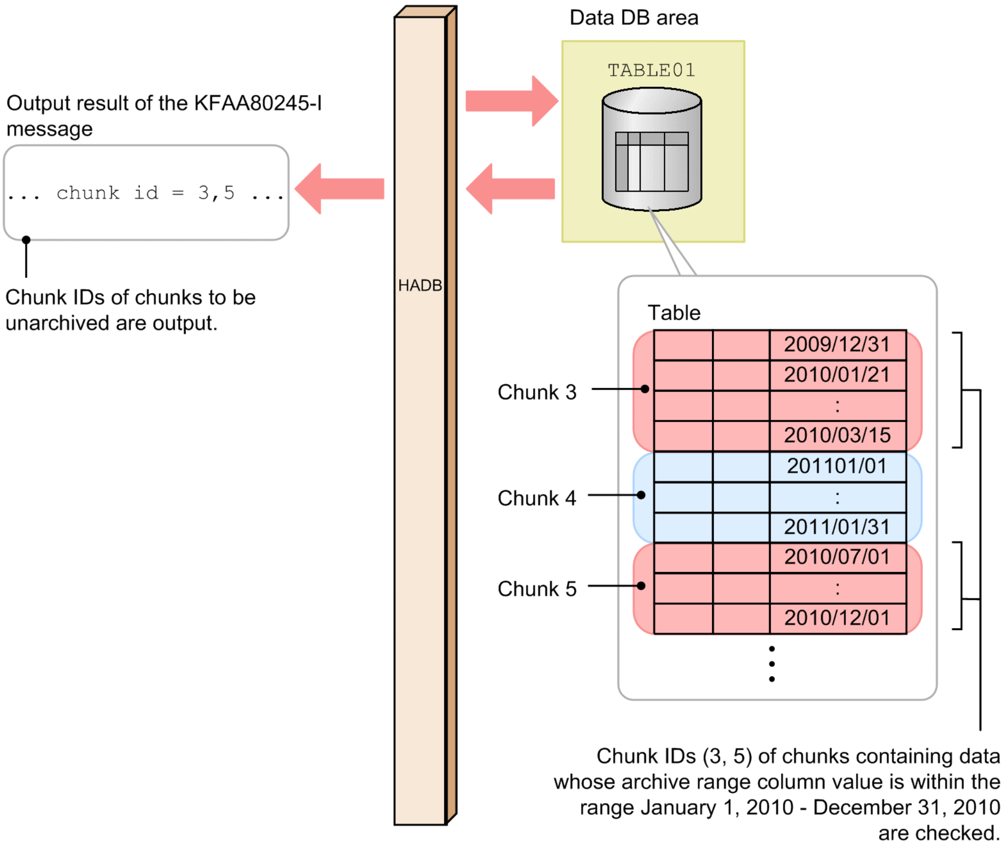
Explanation:

1. Specifies the authorization identifier of the HADB user who executes the adbunarchivechunk command.
2. Specifies the password for the authorization identifier that was specified in the -u option.
3. Specifies the absolute path name of the unarchive chunk option file created in (1) Creating an unarchive chunk option file.
4. Uses dates to specify a range of chunks to be unarchived.
5. Specifies the table to be processed.

43.3.2 When you want to check which chunks are to be unarchived based on a specified date range before actually executing chunk unarchive processing

Check which chunks are to be unarchived based on a specified date range before actually executing chunk unarchive processing, under the following conditions:

- The base table TABLE01 is an archivable multi-chunk table.
- The data type in the archive range column of the base table TABLE01 is DATE.
- Among the chunks created in the base table TABLE01, check the chunk IDs of the chunks to be unarchived whose values in the archive range column contain data in the range from January 1st, 2010 to December 31st, 2010.


Data DB area
TABLE01
Output result of the KFAA80245-I message
... chunk id = 3,5 ...
Chunk IDs of chunks to be unarchived are output.
HADB
Table
2009/12/31
2010/01/21
:
2010/03/15
201101/01
:
2011/01/31
2010/07/01
:
2010/12/01
Chunk 3
Chunk 4
Chunk 5
Chunk IDs (3, 5) of chunks containing data whose archive range column value is within the range January 1, 2010 - December 31, 2010 are checked.

This example uses the following procedure to execute the `adbunarchivechunk` command.

Procedure

1. Execute the `adbunarchivechunk` command.
2. Check the `KFAA80245-I` message to see which chunks include data in the specified date range.

Using an example, the following subsections explain each of these steps.

(1) Executing the adbunarchivechunk command

```
adbunarchivechunk -u ADBUSER01                ...1
                  -p '#HelloHADB_01'          ...2
                  -r 2010/01/01-2010/12/31    ...3
                  -t                          ...4
                  TABLE01                     ...5
```

Explanation:

1. Specifies the authorization identifier of the HADB user who executes the `adbunarchivechunk` command.
2. Specifies the password for the authorization identifier that was specified in the `-u` option.
3. Uses dates to specify a range of chunks to be unarchived.
4. This option is specified in order to check the chunk IDs of chunks that will actually be unarchived before executing chunk unarchive processing.
5. Specifies the table to be processed.

(2) Using the KFAA80245-I message to check which chunks include data in the specified date range

See the KFAA80245-I message that is output after the adbunarchivechunk command is executed with the -t option specified.

The KFAA80245-I message outputs the chunk IDs of chunks to be unarchived whose values in the archive range column include data in the specified date range (January 1st, 2010 to December 31st, 2010).

To perform chunk unarchive processing on these chunks, do not change any options other than the -t option, and execute the adbunarchivechunk command with the -t option omitted. To revise which chunks are to be unarchived, correct the specified date range (in the -r option), and then execute the adbunarchivechunk command.

43.4 Return codes

The following table lists and describes the return codes of the `adbunarchivechunk` command.

Table 43-15: Return codes of the adbunarchivechunk command

Return code	Description
`0`	The `adbunarchivechunk` command terminated normally.
`4`	The `adbunarchivechunk` command terminated normally, but a warning message was issued. Check the displayed message or the message output to a message log file and take appropriate action.
`8`	One of the following errors occurred: • The specified `adbunarchivechunk` command did not execute because it was invalid. • An error occurred during execution of the `adbunarchivechunk` command. For the cause of the error and corrective action to take, see the displayed error message or the message that has been output to a message log file.

43.5 Handling abnormal termination of the adbunarchivechunk command

This section explains the corrective action to take when the `adbunarchivechunk` command terminates abnormally.

43.5.1 First thing to check

Check if the following messages have been issued:

- `KFAA50225-E`
- `KFAA96606-E`

If either of these messages has been output, there is an error in the specified command options or unarchive chunk options. Correct the error in the command options or unarchive chunk options, and then re-execute the `adbunarchivechunk` command.

If neither of the preceding messages was output, eliminate the cause of the command terminating abnormally, and re-execute the `adbunarchivechunk` command. For details, see 43.5.2 Corrective actions to take when the adbunarchivechunk command is canceled, by time of cancellation.

For details about invalid `-c` option specifications, see (3) Errors that can be caused by the -c option in 43.2.1 Explanation of the specification format and options.

Also, for details about invalid `-r` option specifications, see the following resources for each type of range specification.

- Date range specification

  See Table 43-5: Message IDs that are displayed when the specified date range is invalid, and possible errors in (a) Specifying ranges of dates in (4) Explanation of the -r option (range specification) in 43.2.1 Explanation of the specification format and options.
- Time range specification

  See Table 43-7: Message IDs that are displayed when the specified time range is invalid, and possible errors in (b) Specifying ranges of times in (4) Explanation of the -r option (range specification) in 43.2.1 Explanation of the specification format and options.
- Time stamp range specification

  See Table 43-9: Message IDs that are displayed when the specified time stamp range is invalid, and possible errors in (c) Specifying ranges of time stamps in (4) Explanation of the -r option (range specification) in 43.2.1 Explanation of the specification format and options.
- Fixed-length character string range specification

  See Table 43-10: Message IDs that are displayed when the specified fixed-length character string range is invalid, and possible errors in (d) Specifying ranges of fixed-length character strings in (4) Explanation of the -r option (range specification) in 43.2.1 Explanation of the specification format and options.
- Decimal range specification

  See Table 43-11: Message IDs that are displayed when the specified decimal range is invalid, and possible errors in (e) Specifying ranges of decimal numbers in (4) Explanation of the -r option (range specification) in 43.2.1 Explanation of the specification format and options.
- Integer range specification

  See Table 43-12: Message IDs that are displayed when the specified integer range is invalid, and possible errors in (f) Specifying ranges of integers in (4) Explanation of the -r option (range specification) in 43.2.1 Explanation of the specification format and options.

- Floating point range specification

  See Table 43-13: Message IDs that are displayed when the specified floating point range is invalid, and possible errors in (g) Specifying ranges of floating point numbers in (4) Explanation of the -r option (range specification) in 43.2.1 Explanation of the specification format and options.

Important

As a rule, do not change the values specified in the `-c` option or the `-r` option when re-executing the `adbunarchivechunk` command. If you change these values, the target chunks of chunk unarchive processing might change. If this happens, unarchiving might not be performed even if you re-execute the `adbunarchivechunk` command that terminated abnormally on the chunks that were not unarchived.

43.5.2 Corrective actions to take when the adbunarchivechunk command is canceled, by time of cancellation

The corrective actions to take when the `adbunarchivechunk` command terminates abnormally, differ depending on when the `adbunarchivechunk` command was canceled.

The following table describes the state of tables and indexes, and corrective action to take depending on when the command was canceled.

Table 43-16: Table and index states, and corrective action to take depending on when the adbunarchivechunk command was canceled

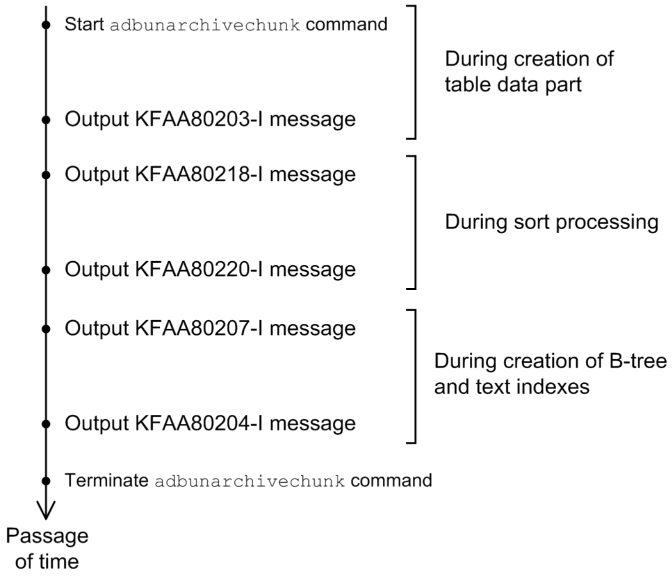
When canceled[#1]	Table and index states	Corrective action to take
During creation of a table data part	- **Table state** The table is in either of the following states: - No data - Same as before the `adbunarchivechunk` command was executed - **Index state** B-tree, text, and range indexes are in either of the following states: - No data - Same as before the `adbunarchivechunk` command was executed	See (1) If the command is canceled during creation of a table data part.
During sort processing	- **Table state** The table is non-updatable. - **Index state** Import of range indexes is complete. Import of B-tree and text indexes is incomplete.	See (2) If the command is canceled during sort processing or creation of indexes.
During creation (but not during update) of B-tree and text indexes[#2]	- **Table state** The table is non-updatable. - **Index state** Import of range indexes is complete. Import of B-tree and text indexes is incomplete.	

Note:

B-tree indexes that correspond to a primary key are also included in the B-tree indexes.

#1

You can determine when the command was canceled by the messages that are issued, as shown in the following:


Start adbunarchivechunk command
Output KFAA80203-I message
During creation of table data part
Output KFAA80218-I message
During sort processing
Output KFAA80220-I message
Output KFAA80207-I message
During creation of B-tree and text indexes
Output KFAA80204-I message
Terminate adbunarchivechunk command
Passage of time

#2

If the `KFAA80204-I` message (with a return code of `0` or `4`) is output, the import processing of tables and indexes is complete. In this case, the `adbunarchivechunk` command does not need to be re-executed.

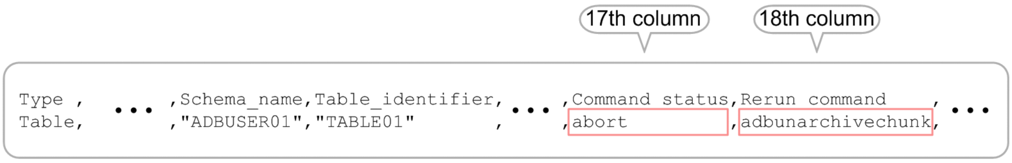
If the `KFAA80204-I` message is not output, execute the following command, and then check the second line in the output results (the column titles occupy the first line in the output results):

```
adbdbstatus -c table -n name-of-table-for-which-adbunarchivechunk-command-was-canc
eled
```

Specify *name-of-table-for-which-adbunarchivechunk-command-was-canceled* in the format *schema-name*`.`*table-identifier*.

- **When the value of the `Command_status` column (17th column) is `abort`, and the value of the `Rerun_command` column (18th column) is `adbunarchivechunk`**

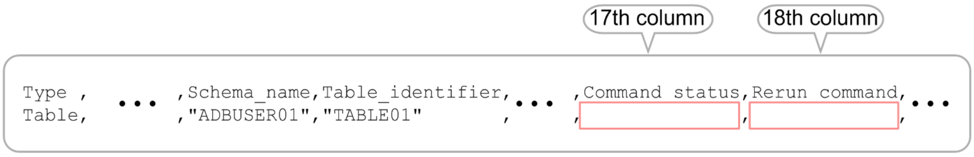
  Chunk unarchive processing is incomplete. Re-execute the `adbunarchivechunk` command.



```
Type ,   ... ,Schema_name,Table_identifier, ... ,Command_status,Rerun_command    , ...
Table,       ,"ADBUSER01","TABLE01"       ,     ,abort         ,adbunarchivechunk,
```

- **When no value is output to the `Command_status` column (17th column) or the `Rerun_command` column (18th column)**

  Chunk unarchive processing is complete. You do not need to re-execute the `adbunarchivechunk` command.



```
Type ,   ... ,Schema_name,Table_identifier, ... ,Command_status,Rerun_command, ...
Table,       ,"ADBUSER01","TABLE01"       ,     ,              ,             ,
```

Important

Even if chunk unarchive processing is complete, end processing (deletion of temporary work files) might be incomplete. In this case, if you execute the `adbunarchivechunk` command on the same table you specified the last time you executed the `adbunarchivechunk` command, only the incomplete end processing is performed (a `KFAA80216-I` message displaying `process = Terminate` is output).

If you execute the `adbunarchivechunk` command not to complete end processing but to execute chunk unarchive processing, and a `KFAA80216-I` message is output, you need to execute the `adbunarchivechunk` command again after the end processing is completed.

Furthermore, when you execute the `adbunarchivechunk` command to complete end processing, any changes made to the `-c` option or `-r` option specifications are invalid (chunk unarchive processing is performed on chunks corresponding to the range specified in the `-c` option or `-r` option last time you executed the `adbunarchivechunk` command).

- **If you are using the multi-node function**

  If the `adbunarchivechunk` command is canceled and the table to be processed becomes non-updatable, make the base table updatable by re-executing the canceled `adbunarchivechunk` command.

  If you can use the node on which the canceled `adbunarchivechunk` command was executed, re-execute the `adbunarchivechunk` command on the same node.

(1) If the command is canceled during creation of a table data part

The following shows the measures to be taken if the `KFAA80218-I` message, which is output at the start of sort processing, has not been output.

Procedure

1. Check the messages that have been issued and identify the cause of the error.
2. Take appropriate action to eliminate the cause of the error.
3. Re-execute the `adbunarchivechunk` command.

(2) If the command is canceled during sort processing or creation of indexes

The following shows the measures to be taken if the `KFAA80218-I` message, which is output at the start of sort processing, has been output.

Procedure

1. Check the messages that have been issued and identify the cause of the error.
2. Take appropriate action to eliminate the cause of the error.
3. Re-execute the `adbunarchivechunk` command.

43.6 Notes

1. The adbunarchivechunk command can only be executed when the HADB server is running.
2. Use the adbcancel command to cancel execution of the adbunarchivechunk command. Alternatively, use a method such as the adbstop --force command or the OS's kill command to forcibly terminate the server process. For details about how to use the adbcancel command to suspend an executing command, see 1.6 When suspending (forcibly terminating) an executing command.
3. Chunk unarchive processing is performed by server processes. Therefore, using the **Ctrl**+**C** keys to forcibly terminate the adbunarchivechunk command process will not interrupt unarchiving chunks that is being performed by server processes (such processing continues). When the chunk unarchive processing is complete, the command outputs the KFAA81001-I message to the message log file.
4. If the adbunarchivechunk command is executed while another command or application program is running, HADB might not be able to allocate the required number of processing real threads. If this happens, HADB waits until the required number of processing real threads can be allocated before executing the adbunarchivechunk command. If allocation of processing real threads is placed in wait status, see the topic *Points to consider about the number of processing real threads to be used during command execution* in *Points to consider when executing commands concurrently* in *Preparing Resources* in the *HADB Setup and Operation Guide* and take appropriate action.
5. Work files (temporary work files) are created temporarily during chunk unarchive processing. If the adbunarchivechunk command terminates successfully (the return code is 0 or 4), the temporary work files are deleted automatically. However, if the adbunarchivechunk command does not terminate successfully, the files might not be deleted.

   Some of the temporary work files are needed when other commands are re-executed. For this reason, do not delete temporary work files before you re-execute other commands. If you delete them, the other commands will not execute correctly.
6. Because temporary work files might cause a shortage of disk space, we recommend that you delete temporary work files from time to time if there is no need to re-execute other commands.

   Check the following directories periodically for any remaining temporary work files and delete any that are found:

   - $DBDIR/ADBSYS/ADBUTL
   - Temporary work file storage directories specified in the -w option (or in $DBDIR/ADBWORK when the -w option was omitted)
7. Make sure that there is no duplication of the absolute paths for the directories and files specified in the adbunarchivechunk command's options. This applies to the following directories and files:

   - Temporary work file storage directories (specified in the -w option or specified in the directory path file specified in the -w option)
   - Directory path files specifying the storage locations of temporary work files (specified in the -w option)
   - Unarchive chunk option file (specified in the -z option)

   Make sure that the absolute paths of these directories and files are unique, and different from those of any directories or files used by any programs other than the adbunarchivechunk command. If there are any duplicate paths, data in the files might be lost. Also make sure that symbolic links and relative paths are specified correctly.
8. If you execute the adbunarchivechunk command with the -t option specified, chunk unarchive processing is not executed. Specify the -t option if you want to check in advance which chunks are to be unarchived, among those corresponding to the range specified in the -r option or the -c option.

To perform chunk unarchive processing on the chunks after checking which chunks are to be unarchived, do not change any options other than the `-t` option, and execute the `adbunarchivechunk` command with the `-t` option omitted.

Index

Hitachi, Ltd.
6-6, Marunouchi 1-chome, Chiyoda-ku, Tokyo, 100-8280 Japan