Utilities Manual

Altibase 7.3

Altibase® Tools & Utilities

1 / 124

Altibase Tools & Utilities Manual

Release 7.3

Copyright ⓒ 2001~2023 Altibase Corp. All Rights Reserved.

Altibase Corp

10F, Daerung PostTower II,

306, Digital-ro, Guro-gu, Seoul 08378, Korea

Telephone : +82-2-2082-1000

Fax   : +82-2-2082-1099

Customer Service Portal : http://support.altibase.com/en/

Homepage   : http://www.altibase.com

Table of Contents

Preface

About This Manual

This manual describes how to use Altibase utilities.

Audience

This manual has been prepared for the following Altibase users:

- Database administrators
- Performance administrators
- Database users
- Application developers
- Technical Supporters

It is recommended for those reading this manual possess the following background knowledge:

- Basic knowledge in the use of computers, operating systems, and operating system utilities
- Experience in using relational database and an understanding of database concepts
- Computer programming experience
- Experience in database server management, operating system management, or network administration

Organization

This manual is organized as follows:

- Chapter 1: aexport
  This chapter describes aexport, a tool for supporting automated data migration between Altibase databases.

- Chapter 2: altiComp
  This chapter describes the altiComp utility's features and explains the capabilities of comparing and matching inconsistent data.

- Chapter 3: aku
  This chapter describes aku, the utility to support the data replication between pods using Kubernetes.

- Chapter 4: Utilities
  This chapter describes the rest of the utilities except aexport, altiComp, and aku.

Documentation Conventions
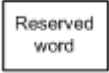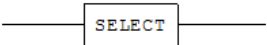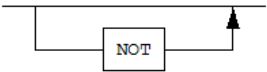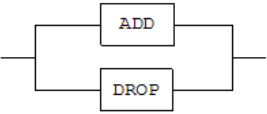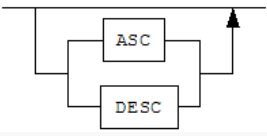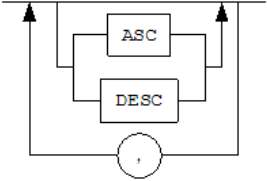
This section describes the conventions used in this manual. Understanding these conventions will make it easier to find information in this manual and in the other manuals in the series.

There are two sets of conventions:

- Syntax diagram conventions
- Sample code conventions

Syntax Diagram Conventions

This manual describes command syntax using diagrams composed of the following elements:

Elements	Meaning
Reserved word	Indicates the start of a command. If a syntactic element starts with an arrow, it is not a complete command.
(right arrow)	Indicates that the command continues to the next line. If a syntactic element ends with this symbol, it is not a complete command.
(right arrow)	Indicates that the command continues from the previous line. If a syntactic element starts with this symbol, it is not a complete command.
(arrow to ;)	Indicates the end of a statement.
SELECT	Indicates a mandatory element.
NOT	Indicates an optional element.
ADD / DROP	Indicates a mandatory element comprised of options. One, and only one, option must be specified.
ASC / DESC	Indicates an optional element comprised of options.
ASC / DESC ,	Indicates an optional element in which multiple elements may be specified. A conma must precede all but the first element.

Sample Code Conventions

The code examples explain SQL statements, stored procedures, iSQL statements, and other command line syntax.

The following table describes the printing conventions used in the code examples.

Rules	Meaning	Example
[ ]	Indicates an optional item	VARCHAR [(*size*)] [[FIXED	] VARIABLE]
{ }	Indicates a mandatory field for which one or more items must be selected.	{ ENABLE	DISABLE	COMPILE }
		A delimiter between optional or mandatory arguments.	{ ENABLE	DISABLE	COMPILE } [ENABLE	DISABLE	COMPILE]

Rules	Meaning	Example
. . .	Indicates that the previous argument is repeated, or that sample code has been omitted.	SQL> SELECT ename FROM employee; ENAME ---------------------- SWNO HJNO HSCHOI . . . 20 rows selected.
Other Symbols	Symbols other than those shown above are part of the actual code.	EXEC :p1 := 1; acc NUMBER(11,2)
Italics	Statement elements in italics indicate variables and special values specified by the user.	SELECT * FROM *table_name*; CONNECT *userID/password*;
Lower case words	Indicate program elements set by the user, such as table names, column names, file names, etc.	SELECT ename FROM employee;
Upper case words	Keywords and all elements provided by the system appear in upper case.	DESC SYSTEM.*SYS_INDICES*;

Related Documentations

For more detailed information, please refer to the following documents.

- Installation Guide
- Administrator's Manual
- Replication Manual
- iSQL User's Manual
- iLoader User's Manual

Altibase Welcomes Your Comments and Feedbacks

Please let us know what you like or dislike about our manuals. To help us with better future versions of our manuals, please tell us if there is any corrections or classifications that you would find useful.

Include the following information:

- The name and version of the manual that you are using
- Any comments about the manual
- Your name, address, and phone number

If you need immediate assistance regarding any errors, omissions, and other technical issues, please contact Altibase's Support Portal.

Thank you. We always welcome your feedbacks and suggestions.

1. aexport

Introducing aexport

Overview

aexport supports automated data migration between Altibases. This utility stores logical structures and data in text format, and automatically creates a script to load the stored text data into a new database.

The objects and components that aexport can extract from a database to which it is connected are database users, user privileges, tables, tablespaces, table constraints, indexes, views, materialized views, stored procedures, sequences, and replication objects.

Because aexport creates SQL scripts corresponding to logical structures in the database and downloads all data in text form, it can migrate data between databases of different versions or platforms. This utility should be used when the database is running, but not actively providing service (when clients are not connected).

aexport Features

aexport can extract the following database objects and structural elements:

- Database users
- User privileges
- Roles
- Tablespaces
- Tables
- Table constraints
- Indexes
- Views
- Materialized views
- Stored procedures
- Replication objects

In order to execute, aexport generates the SQL scripts to create the database components listed above and the shell scripts to run them.

aexport Modes and Script Files

aexport can be executed in different modes to extract different portions of the database. The desired mode can be specified on the command-line.

The aexport execution mode and the SQL script files generated for each mode are described in the section below.

Full DB Mode

Full DB mode extracts the entire database and is only available for the SYS user.

The following SQL script files are generated for this mode:

- SYS_CRT_DIR.sql: Creates all directory objects.
- SYS_CRT_USER.sql: Creates all users and roles.
- SYS_CRT_SYNONYM.sql: Create all synonym objects.
- SYS_CRT_REP.sql: Create all replicaiton objects.
- ALL_CRT_VIEW_PROC.sql: Creates all views and stored procedures.
- ALL_CRT_TBS.sql: Creates all tablespaces.
- ALL_CRT_TBL.sql: Creates all user tables.
- ALL_CRT_INDEX.sql: Creates all user indexes.
- ALL_CRT_FK.sql: Creates all user-defined foreign keys.
- ALL_CRT_TRIG.sql: Creates all user-defined triggers.
- ALL_CRT_SEQ.sql: Creates all user-defined sequences.
- ALL_CRT_LINK.sql: Creates all user-defined database link objects.
- ALL_EXE_STATS.sql: Creates statistics for all user-defined tables, columns, and indexes.
- ALL_REFRESH_MVIEW.sql: Refreshes all user materialized views.
- ALL_ALT_TBL.sql : Switches the data access mode for tables and partitions of all users.

> Note:
>
> A role can only be extracted in full DB mode, because it is a non-schema object.

User Mode

This mode exports all objects owned by a specified user and is available only for the SYS user or the user whose objects are to be exported. Set the -u command-line option to the desired user for this mode.

The following SQL script files are generated for this mode:

- {User name}_CRT_TBL.sql: Creates all tables of the specified user.
- {User name}_CRT_INDEX.sql: Creates all indexes of the specified user.
- {User name}_CRT_FK.sql: Creates all foreign keys of the specified user.
- {User name}_CRT_TRIG.sql: Creates all triggers of the specified user.
- {User name}_CRT_SEQ.sql: Creates all sequences of the specified user.
- {User name}_CRT_LINK.sql: Creates all database link objects of the specified user
- {User name}_EXE_STATS.sql: Creates statistics for all tables, columns and indexes of specified user
- {User name}_REFRESH_MVIEW.sql: Refreshes all materialized views of the specified user.
- {User name}_ALT_TBL.sql : Switches the data access mode for tables and partitions of the specified user.

Object Mode

Object mode exports a specified set of objects (user.object) and is available only for the SYS user or the user whose objects are to be exported. Use the -object command-line option for this mode.

All specified objects must belong to the same user; however, the SYS user can export any user object.

The following SQL script files are generated for this mode

- {User name}*{Object name}*CRT.sql: Creates the specified user object.

- {User name}*{Object name}*STATS.sql: Creates the specified user statistics.

Shell Script Files

In addition to the above SQL scripts, the following shell script files are also created when aexport is executed:

- run_il_in.sh: Loads data.

- run_il_out.sh: Downloads data.

- run_is.sh: Creates schema.

- run_is_con.sh: Creates constraints. This script includes SQL scripts for creating indexes, foreign keys, triggers, and replication objects. This script is created if the TWO_PHASE_SCRIPT property is set to ON.

- run_is_fk.sh: Creates foreign keys and triggers. This script is not created if the TWO_PHASE_SCRIPT property is set to ON.

- run_is_index.sh: Creates indexes. This script is not created if the TWO_PHASE_SCRIPT property is set to ON.

- run_is_repl.sh: Creates replication objects. This script is not created if the TWO_PHASE_SCRIPT property is set to ON.

- run_is_refresh_mview.sh: Refreshes materialized views. This script is not created if the TWO_PHASE_SCRIPT property is set to ON.

- run_is_alt_tbl.sh : Switches the data access mode of tables and partitions. This script is not created if the TWO_PHASE_SCRIPT property is set to ON.

When one of the above shell script files is executed on the destination database, the logical structure of the source database is created on the destination database. Additionally, all data that exists on the source database is loaded into the destination database. These shell script files use iLoader to download and upload data; the iLoader process is automated within the shell script.

All files generated by aexport are text files, so the user can modify them as desired.

- **aexport Properties and Script Files**

  This section discusses script files generated by aexport properties.

  Please refer to aexport Properties for more information.

  - INVALID_SCRIPT = ON,INVALID.sql is generated. This script file contains SQL scripts for all invalid views and stored procedures. A shell script file for executing INVALID.sql is not generated.

- ○ TWO_PHASE_SCRIPT = OON, ALL_OBJECT.sql is generated for all objects and ALL_OBJECT_CONSTRAINTS.sql for all indexes, foreign keys, triggers, and replication objects. The run_is_con.sh shell script file for executing ALL_OBJECT_CONSTRAINTS.sql is also generated.

Prerequisite

Install DBMS_METADATA Package

The DBMS_METADATA package provides functionality to extract object creation DDL statements or GRANT statements from the database dictionary.

To use aexport, it is necessary to have the DBMS_METADATA package installed in Altibase, as aexport has a dependency on this package. If not, the following error occurs:

```
$ aexport -s localhost -u sys -p manager
-----------------------------------------------------------------
     Altibase Export Script Utility.
     Release Version 7.3.0.0.0
     Copyright 2000, ALTIBASE Corporation or its subsidiaries.
     All Rights Reserved.
-----------------------------------------------------------------
[ERR-91144 : DBMS_METADATA package does not exist.]
```

> For more detailed information about DBMS_METADATA package, please refer to **_Stored Procedures Manual_** - DBMS_METADATA.

Setting aexport

aexport requires the following information to connect to a server:

- ALTIBASE_HOME
  The path where the server or client is installed.

- server_name
  The name or IP address of the computer hosting the database from which data is to be downloaded.

- port_no
  The port number to be used to connect over TCP or IPC.

- user_id
  The database user identifier used by aexport to connect to the database.

- Password
  The password for the database user identifier.

- NLS_USE
  The character set in which to display data.

The path where the server or client is installed can only be set with the ALTIBASE_HOME environment variable. The rest can be set with command-line options. For more detailed information about command-line options, please refer to "How to Use aexport".

The ALTIBASE_HOME environment variable must be correctly set, and the aexport property settings file (aexport.properties) must exist and be properly configured for aexport to run successfully. For more detailed information about the aexport.properties file, please refer to "aexport Properties".

ALTIBASE_HOME is usually set automatically when the server is installed. For the client, the user must set it manually. If it is not set, it may not work properly, so it is recommended to check whether it is set correctly before executing.

port_no and NLS_USE can be set with environment variables or the altibase.properties file. If set in more than one way, the following methods take precedence in descending order.

1. Command-line options
2. Environment variables (ALTIBASE_PORT_NO, ALTIBASE_NLS_USE)
3. The altibase.properties file

On omission, the user is prompted to enter a value immediately after aexporthas been executed. aexport may not work normally if the value is invalid.

If the option is not set, the first time aexport is run, it prompts for an option and prompts the user for a value.If incorrect format or an invalid value is entered, aexport may not work properly.

One exception is the NLS_USE option. On omission, the user is not prompted to enter a value, but the database character set is used by default. If the user omits the NLS_USE option in an environment that does not use the US7ASCII character set, aexport will run abnormally with the risk of data loss. Therefore, the user should set the NLS_USE option to a value that is compatible with his or her operating environment.

The user is recommended to set the following environment variables for aexport to run normally:

- ALTIBASE_HOME: The path where the server or client is installed.
- ALTIBASE_PORT_NO: The port number used to connect to the server.
- ALTIBASE_NLS_USE: The character set used to export and import data.
- PATH: The path to the aexport executable file. It is normally $ALTIBASE_HOME/bin.

Environment Variables

ALTIBASE_HOME

Sets the directory in which the package was installed. This must be set to use aexport.

ALTIBASE_PORT_NO

Sets the port number used to connect to the server. This can also be set with the -port command-line option, or set in advance in the altibase.properties file.

If different values are set for the ALTIBASE_PORT_NO environment variable and the altibase.properties file, the environment variable takes precedence. However, the value set with the -port command-line option overrides both. On omission, the user will be prompted to enter a value after aexport has started.

ALTIBASE_SSL_PORT_NO

Sets the server port number that aexport is to connect to over SSL/TLS.

For the port number in SSL, the -PORT option, environment variables, ALTIBASE_SSL_PORT_NO, and the properties in the altibase.properties file take precedence over each other (in consecutive order). On omission, the user is prompted to enter the port number.

ALTIBASE_NLS_USE

Sets the character set use when connecting to the server. This can also be set with the - nls_use command-line option, or be set in advance in the altibase.properties file.

If different values are set for the ALTIBASE_NLS_USE environment variable and the altibase.properties file, the environment variable takes precedence. However, the value set with the -nls_use command-line option overrides both.

Note: If the server character set and the value set in ALTIBASE_NLS_USE are different, it may not work properly. It is recommended to set the appropriate value.

How to Use aexport

Syntax

Parameters

Parameter	Description
-h	Displays the Help menu
-s	Sets the host name or IP of the server from which to download data. On omission, the user is prompted for the host name. This can be a host name, an IPv4 address, or an IPv6 address. If it is an IPv6 address, it must be enclosed in square brackets ("[ ]"). The localhost (the same computer on which aexport is executed) can be set as the computer's host name, the localhost's IPv4 address (usually 127.0.0.1), or the localhost IPv6 address (usually [::1]). For more information about Altibase and IPv6 address notation, please refer to the *Administrator's Manual.*

Parameter	Description
-u	This sents the name of the Altibase user to access the server from whichd data is to be downloaded. On omission, the user is prompted for the user name.This option must be set to the SYS user to perform a full DB mode export. Use double quotation marks if there are lower case letters, special characters or spaces in the username. -u "user name"
-p	Sets the password for the user. On omission, the user is prompted for a password.
-port	Sets the server port number to access for data download. On omission, the ALTIBASE_PORT_NO environment variable and the altibase.properties setting are checked in turn to determine the port number. If neither is set, the user is prompted for a port number.
-object	This option specifies an object which will be extracted along with the owner name. Use double quotation marks if there are lower case letters, special characters or space in the object name. -object "user name"."table name"
-tserver	Sets the destination server (the server to which the exported data is to be uploaded). This information is written to the script files that are created when aexport is executed, and used when those scripts are subsequently executed. As with the -s option, this can be a host name, an IPv4 address or an IPv6 address.
-tport	Sets the destination server port number. This information is written to the script files that are created when aexport is executed, and used when those scripts are subsequently executed.
-nls_use	Sets the character set to export (download) data from the source database and import (upload) data to the destination database. At present, the supported character sets are US7ASCII, KO16KSC5601, MS949, BIG5, GB231280, MS936, UTF8, SHIFTJIS, MS932 and EUCJP.
-prefer_ipv6	This option determines whether to first attempt to resolve a host name to an IPv4 address or an IPv6 address. If a host name is specified for the -s option and this option is used, aexport will first attempt to resolve the host name to an IPv6 address. In contrast, if a host name is specified for the -s option and this option is omitted, aexport will first attempt to resolve the host name to an IPv4 address. That is, the default behavior is to attempt to resolve the host name to an IPv4 address. If aexport fails to connect using the preferred IP address type, it attempts to connect using the other IP address type. For example, when localhost is specified for the -s option and this option is used, aexport first tries to connect to the [::1] IPv6 address. If this attempt fails, aexport then attempts to connect to the 127.0.0.1 IPv4 address.
-ssl_ca *CA_file_path*	Specifies the location of the certification authority (CA) certificate in which the public key of the Altibase server to be connected to is incorporated.

Parameter	Description
-ssl_capath *CA_dir_path*	Specifies the directory under which the certification authority (CA) certificate in which the public key of the Altibase server to be connected is incorporated.
-ssl_cert *certificate_file_path*	Specifies the location of the client authentication file.
-ssl_key *key_file_path*	Specifies the location of the client private key file.
-ssl_verify	Verifies the certificate the client receives from the server.
-ssl_cipher *cipher_list*	Specifies a cipher list for SSL encryption. Please refer to the SSL_CIPHER_LIST property in the *General Reference.*

> For more detailed information about SSL connection, please refer to Chapter 2. Connecting and Disconnecting in the *iSQL User's Manual.*

The Data Migration Process

The process of using aexport to migrate data can be roughly divided into the following steps:

- Generate SQL script files for creating the structure of the objects to be exported from the source database and shell script files for executing the SQL script files
- Export (download) the data from the source database
- Create the required database structures in the destination database
- Import (upload) the data to the destination database
- Refresh the materialized view in the destination database, and create indexes and foreign keys in the destination database, and then switch the data access mode

Exporting the Source Database Structure

aexport is first used to generate SQL script files that contain information about the structure of the source database and shell script files for executing the SQL script files.

- Execute aexport

```
$ aexport –s 127.0.0.1 –u sys –p manager
```

- Enter the passwords of the Altibase users at the prompts. This sets the password for each user that is created in the destination database.
- When using aexport to back up the data on a remote server, indicate the address of the remote server and the port through which to connect to the remote server.

```
$ aexport –s 222.112.84.200 –port 20300 –u sys –p manager
```

Exporting Data from the Source Database

Export (download) the data from the source database by executing the shell script that was created by aexport in the previous step.

- Check the disk to which the data are to be downloaded to ensure that it has enough free space to hold the data. Because data in text form can occupy more space than the data in internally used data files, it is recommended that the amount of available free space be twice the size of the original data files.

- Execute the "run_il_out.sh" script.

```
$ sh run_il_out.sh
```

Creating the Destination Database Structure

- Copy all SQL scripts and shell scripts and all 'fmt', 'log', and 'dat' format files created by the "run_il_out.sh" shell script to the system on which the destination database is located. Skip this step if the destination database is on the same system as the source database.

- Start up the destination database.

- Execute the "run_is.sh" script.

```
$ sh run_is.sh
```

- Use iSQL to access the database and check whether all of the required database objects were properly created. If the required database structure was not properly created, inspect the output that was displayed on the screen while run_is.sh was executing to determine the cause of the problem.

Importing Data into the Destination Database

- Execute the "run_il_in.sh" script.

```
$ sh run_il_in.sh
```

- Check the directory containing the "run_il_in.sh" shell script file to see whether it contains any files that have the "*.bad" filename extension and are greater than 0 bytes in size. If such a file exists, inspect the contents of the "*.bad" file and the log files related to the table having the same name as the "*.bad" file and take suitable steps to resolve the problem. For more information on how to resolve such problems, please refer to the *iLoader User's Manual.*

Refresh the materialized view in the destination database, and create indexes and foreign keys in the destination database, and then switch the data access mode

When the TWO_PHASE_SCRIPT property is set to "Off":

- Execute the "run_is_refresh_mview.sh" script.

```
$ sh run_is_refresh_mview.sh
```

- Execute the "run_is_index.sh" script.

```
$ sh run_is_index.sh
```

- Execute the "run_is_fk.sh" script.

```
$ sh run_is_fk.sh
```

- Execute the "run_is_alt_tbl.sh" script.

```
$ sh run_is_alt_tbl.sh
```

When TWO_PHASE_SCRIPT = ON:

- Execute the "run_is_con.sh" script.

```
$ sh run_is_con.sh
```

Notes

- If a normal user who is not the SYS user executes aexport, scripts are created only for the user's schema.
- If a normal user who is not the SYS user executes aexport, scripts are created for replication objects.
- If a normal user who is not the SYS user executes aexport, CREATE TABLE privileges are required. This is because aexport creates temporary tables to analyze object interdependencies.
- Do not run two or more aexport processes at the same time. Because aexport uses a temporary table to store created SQL scripts, running two or more aexport processes at the same time will yield unpredictable results.
- If the EXECUTE and TWO_PHASE_SCRIPT aexport properties are both set to ON and the OPERATION property is set to IN when uploading data, this uploading operation will not be affected by the value of the INDEX property, because no SQL script file dedicated to creating the index is generated.
  Therefore, when it is desired to perform the uploading operation (EXECUTE = ON and OPERATION = IN) with ON for the INDEX property, the TWO_PHASE_SCRIPT property must be set to OFF.
- When the "run_is.sh" script is executed, all existing users and objects will be deleted from the database. Therefore, care must be taken to avoid executing this script on the source database.
- If the -tserver and -tport options are not specified, then the -s and -p parameters are used not only to identify the source server from which data are downloaded, but are also used in all created scripts to identify the destination server to which data will be uploaded. To specify a destination server and port that are different from the source server and port, use the -tserver and -tport options. In this case, the -s and -p options will only identify the source server from which data are downloaded, whereas the values specified for the -tserver and -tport options will be written into the created scripts.

```
$ aexport -s 127.0.0.1 -u sys -p manager -tserver 192.168.1.10 -tport 21300

$ cat run_il_in.sh

iloader -s 192.168.1.10 -port 21300 -u SYS -p MANAGER in -f SYS_T1.fmt -d
SYS_T1.dat -log SYS_T1.log -bad SYS_T1.bad
```

- If CALLBACK functions are specified using PASSWORD_VERIFY_FUNCTION at user creation, the user's password must be set to correspond to validity functions before user importation. Validity functions must also be imported before importing the user to the database.

- Double quotation marks should be used if there are lower case letters, special characters or spaces in the user name or the name of objects which will be extracted.

Limitations

- If a stored procedure needs to refer to when creating a stored procedure is not created in advance, the operation will fail. aexport cannot guarantee the order in which stored procedures are created because they cannot access information about dependencies between stored procedures. In this case, the stored procedure creation can fail, so the stored procedure must be created in the destination database.

- aexport only has limited access to the meta information of the sequence. Because of this restriction, only the sequence characteristics specified by the INCINMENT BY of a sequence created in an account other than the SYS user are reflected, and the remaining attributes are set to the default values. If these constraints are a problem, the sequence must be created in the destination database.

- In creating an object in the destination database, the base table must be created before creating the Material View. Because aexport does not guarantee the table creation order for creating materialized views, it may fail to create a materialized view. In this case, the user must manually create a materialized view

- When aexport extracts the materialized view creation statement from the source database, it imports the statement that originally created the materialized view. That is, even if the DDL statement that changes the refresh method or refresh change time is executed for the materialized view in the source database, the change is not reflected in the statement extracted by aexport.

SSL Connection and Script Files

- If aexport is executed on SSL, the SSL option that was specified to execute aexport is used with the source database connection script (run_il_out.sh).

- In to connect an SSL to the target databe, the SSL related properties must be set in the property file. Refer to ILOADER_ARRAY in the aexport property section for a detailed description.
  ILOADER_ARRAY = *count* (Default: 1)
  Speifcies the number of rows to process at once when downloading or uploading data to iLoader.

- ILOADER_COMMIT
  ILOADER_COMMIT = *count* (Default: 1000)
  Specifies the unit (number) to commit when uploading data to iLoader. The value of the -commit option can be specified with this property.

- ILOADER_PARALLEL
  ILOADER_PARALLEL = *count* (Default: 1)
  Specifies the number of threads to process in parallel when downloading or uploading to iLoader.

- ILOADER_ASYNC_PREFETCH
  ILOADER_ASYNC_PREFETCH = OFF|ON|AUTO (Default: OFF)
  Sets asynchronous prefetch behavior when downloading data to iLoader. For more informaiton please refer to the '-async_prefetch' option in the *iLoader User's Manual.*

- Refer to the SSL_ENABLE property.

aexport Properties

Setting the aexport Properties

Some of the settings that govern the use of aexport are made in the aexport.properties file. The aexport.properties file must be located in the $ALTIBASE_HOME/conf directory. (This file is not to be confused with the altibase.properties file, which by default is located in the same directory.)

When Altibase is installed, the $ALTIBASE_HOME/conf directory does not actually contain a file called aexport.properties, but it does contain a sample properties file called aexport.properties.sample. It is thus necessary to copy the aexport.properties.sample, paste it into the same directory, and rename it as aexport.properties before executing aexport. If aexport cannot find the aexport.properties file in $ALTIBASE_HOME/conf, it will raise an error and terminate.

List of aexport Properties

- OPERATION
  OPERATION = IN/OUT
  If this property is set to OUT, scripts for exporting all schemas and data will be created. When the data export script, which consists of iLoader commands, is executed, form files (.fmt) and data files (.dat) will be created.
  If this property is set to IN, the schema creation script and the data loading script, which were created by previously executing aexport with this property set to OUT, will be executed, the schema will be created in the destination database, and the data will be loaded into the destinaiton database. The schema creation script and the data loading script can be executed manually at a shell prompt without executing aexport.

- EXECUTE
  This property determines whether to automatically execute the scripts that were created.
  EXECUTE = ON/OFF
  If it is set to ON, the scripts that are appropriate for the current operation (set using the OPERATION property) will be executed automatically. The file names of these scripts are set using the ILOADER_OUT, ILOADER_IN, ISQL, ISQL_CON, ISQL_INDEX, ISQL_FOREIGN_KEY, ISQL_ALT_TBL, and ISQL_REPL properties.
  If it is set to OFF, the scripts will be created, but not executed.

- INVALID_SCRIPT
  This property determines whether to group all of the object creation scripts for invalid objects in a single script file.
  INVALID_SCRIPT = ON/OFF
  If this property is set to OFF, a SQL script file will be generated for each of the invalid objects in the database; that is, they will be treated just like the valid database objects.

- TWO_PHASE_SCRIPT
  This property determines whether to group all of the object creation scripts in two script files.
  TWO_PHASE_SCRIPT = ON/OFF
  If this property is set to ON, aexport will create only two SQL script files and two shell script files: ALL_OBJECT.sql, ALL_OBJECT_CONSTRAINS.sql, ALL_OBJECT.sql, run_is.sh, run_is_con.sh
  If this property is set to OFF, aexport will generate different SQL script files for each of the objects in a database.

- CRT_TBS_USER_MODE
  CRT_TBS_USER_MODE = ON/OFF (Default: OFF)
  This property determines whether or not to extract a statement for creating tablespace in user mode.
  If this property is set to ON, the SQL statement creating tablespace related to the user in the user mode is extracted. The user related tablespaces are default tablespace, default temporary tablespace, and the tablespace specified whether or not to available to be accessed.

- INDEX
  INDEX = ON/OFF
  This property determines whether or not to create the indexes when creating the rest of the schema in the destination database. If it is desired to create the indexes after the data have been located into the destination database, set this property to ON. It is used when the TWO_PHASE_SCRIPT property is set to OFF.

- USER_PASSWORD
  USER_PASSWORD = *password*
  This property is used to set the password when the users exported from the source database are created in the destination database. (Because aexport does not know the passwords of users exported from the source database, the passwords must be manually set.) If this property is not set, a prompt for setting each user's password will appear.

- VIEW_FORCE
  VIEW_FORCE = ON/OFF
  If this property is set to ON, views will be forcibly created, even if the underlying tables or other objects don't exist.

- DROP
  This property determines whether to include DROP statements in created scripts.
  DROP = ON/OFF
  If this property is set to ON, and if the destination database already contains objects corresponding to those that are to be created, the existing objects will be dropped. Because this option specifies that existing objects are to be dropped, it should be used with caution.
  Note) If aexport is executed in Object Mode, DROP statements are not generated, regardless of the setting of this property.

- ILOADER_OUT
  ILOADER_OUT = *run_il_out.sh*
  This property determines the name of the shell script file that is created to export (download) the data from the source database. It is used when the OPERATION property is set to OUT.

- ILOADER_IN
  ILOADER_IN = *run_il_in.sh*
  This property determines the name of the shell script file that will be used to import (upload) the data into the destination database.

- ISQL
  ISQL = *run_is.sh*
  This property determines the name of the script file that will be used to create the database schema in the destination database.

- ISQL_CON
  ISQL_ CON = *run_is_con.sh*
  This property determines the name of the shell script file that is used to execute the SQL script files for creating indexes, foreign keys, triggers and replication objects. It is used when the TWO_PHASE_SCRIPT property is set to ON.

- ISQL_INDEX
  ISQL_INDEX = *run_is_index.sh*
  This property determines the name of the shell script that will be used to create indexes in the destination database. If no value is specified for this property in the aexport.properties file, this shell script file will not be generated.

- ISQL_FOREIGN_KEY
  ISQL_ FOREIGN_KEY = *run_is_fk.sh*
  This property determines the name of the shell script file that is used to execute the SQL script files for creating foreign keys. If no value is specified for this property in the aexport.properties file, this shell script file will not be generated.

- ISQL_REPL
  ISQL_REPL = *run_is_repl.sh*
  This property determines the name of the shell script that will be used to create replication objects in the destination database. If no value is specified for this property in the aexport.properties file, this shell script file will not be generated.

- COLLECT_DBMS_STATS
  This property determines whether to display the statistics for tables, columns and indexes of specified user.
  COLLECT_DBMS_STATS = ON/OFF
  The default value is OFF and no statistical information is exported. When this property's value is ON, statistics information is exported.

- ISQL_REFERSH_MVIEW
  ISQL_REFERSH_MVIEW = *run_is_refresh_mview.sh*
  This property determines the name of the shell script that will be used to execute the SQL script files for refreshing the materialized view in the destination database. If no value is specified for this property in the aexport.properties file, this shell script file will not be generated.

- ISQL_ALT_TBL
  ISQL_ALT_TBL = *run_is_alt_tbl.sh*
  This property sets the name of the shell script file executing the SQL script which switches the data access mode for the tables and partitions of the target database. On omission, aexport does not generate this shell script file.

- ILOADER_FIELD_TERM
  ILOADER_FIELD_TERM = *field_term*
  This property is used to set the field delimiters that are used when the data in tables are saved as text. If this property is not set, the default delimiter between values is the comma (","), no block delimiters are used for numeric values, and double quotation marks (" " ") are used as block delimiters around strings.
  Note) The pound (i.e. hash or number sign) character "#" cannot be specified as a delimiter, because it is used to denote comments in the properties file (The remainder of the line after the "#" will be ignored).

- ILOADER_ROW_TERM
  ILOADER_ ROW _TERM = *row_term*
  This property sets the record delimiter to use when texting table data. If not set, the default is <LF>.
  Note) The pound (i.e. hash or number sign) character "#" cannot be specified as the record delimiter, because it is used to denote comments in the properties file (The remainder of the line after the "#" will be ignored).

- ILOADER_PARTITION
  This property determines whether the iLoader scripts are generated for each partition when there is a partitioned table in the source database.
  ILOADER_ PARTITION = ON/OFF

  When this property is set to ON, scripts for extracting form files and data for each partition are generated in run_il_out.sh. Similarly, scripts for loading data for each partition are generated in run_il_in.sh.

  When this property is set to OFF, scripts for handling partitioned tables are generated to use a single data file for extraction and loading, similar to non-partitioned tables.

  For more detailed information, please refer to the *iLoader User's Manual.*

- ILOADER_ERRORS
  ILOADER_ERRORS = *count* (Default: 50)
  This property specifies the number of allowable maximum errors when uploading data with iLoader. The default value of this property is set to 50, and the upload is continuously executed regardless of the number of incurring errors if the default value is set to 0.

- ILOADER_ARRAY
  ILOADER_ARRAY = *count* (Default: 1)
  This property specifies the number of rows to be processed at once when downloading or uploading data with iLoader.

- ILOADER_COMMIT
  ILOADER_COMMIT = *count* (Default: 1000)
  This property specifies the unit(number) to commit when uploading the data with iLoader. The value of -commit option can be specified as well.

- ILOADER_PARALLEL
  ILOADER_PARALLEL = *count* (Default: 1)
  This property specifies the number of threads which will be executed with parallel processing when uploading or downloading with iLoader.

- ILOADER_ASYNC_PREFETCH
  ILOADER_ASYNC_PREFETCH = OFF|ON|AUTO (Default: OFF)
  This property sets asynchronous prefect behavior when downloading data to iLoader. For more information, please refer to the '-async_prefetch' option in the *iLoader User's Manual.*

- SSL_ENABLE
  This property specifies whether to connect to the target database using the SSL protocol.
  SSL_ENABLE = ON/OFF
  If this property is set to ON, SSL-related options are enabled for iSQL and iLoader commands in the shell scripts (run_is.sh and run_il_in.sh) to be executed on the target database. SSL-related options can be enabled with the SSL_CA, SSL_CAPATH, SSL_CERT, SSL_KEY, SSL_CIPHER, SSL_VERIFY properties. For further information about these properties, please refer to Parameters.

Example

Execution in Full DB Mode

```
$ aexport -s 127.0.0.1 -u sys -p manager
-----------------------------------------------------------------
    Altibase Export Script Utility.
    Release Version 7.1.0.0.0
    Copyright 2000, ALTIBASE Corporation or its subsidiaries.
    All Rights Reserved.
-----------------------------------------------------------------
##### TBS #####
##### USER  #####
##### SYNONYM #####
##### DIRECTORY #####
##### TABLE #####
##### QUEUE #####
##### SEQUENCE #####
##### DATABASE LINK #####
##### VIEW #####
##### MATERIALIZED VIEW #####
##### STORED PROCEDURE #####
##### STORED PACKAGE #####
##### TRIGGER #####
##### LIBRARY #####
##### REPLICATION #####
##### JOB #####
-------------------------------------------------------
  ##### The following script files were generated. #####
  1. run_il_out.sh            : [ iloader formout, data-out script ]
  2. run_is.sh                : [ isql table-schema script ]
  3. run_il_in.sh             : [ iloader data-in script ]
  4. run_is_refresh_mview.sh  : [ isql materialized view refresh script ]
  5. run_is_index.sh          : [ isql table-index script ]
  6. run_is_fk.sh             : [ isql table-foreign key script ]
```

```
  7. run_is_repl.sh          : [ isql replication script ]
  8. run_is_job.sh           : [ isql job script ]
  9. run_is_alt_tbl.sh       : [ isql table-alter script ]
-----------------------------------------------------

$ ls -l
ALL_ALT_TBL.sql
ALL_CRT_DIR.sql
ALL_CRT_FK.sql
ALL_CRT_INDEX.sql
ALL_CRT_JOB.sql
ALL_CRT_LIB.sql
ALL_CRT_LINK.sql
ALL_CRT_REP.sql
ALL_CRT_SEQ.sql
ALL_CRT_SYN.sql
ALL_CRT_TBL.sql
ALL_CRT_TBS.sql
ALL_CRT_TRIG.sql
ALL_CRT_USER.sql
ALL_CRT_VIEW_PROC.sql
ALL_REFRESH_MVIEW.sql
run_il_in.sh
run_il_out.sh
run_is.sh
run_is_alt_tbl.sh
run_is_fk.sh
run_is_index.sh
run_is_job.sh
run_is_refresh_mview.sh
run_is_repl.sh
```

Execution in User Mode

```
iSQL> CREATE USER user1 IDENTIFIED BY user1;
Create success.
$ aexport -s 127.0.0.1 -u user1 -p user1
-----------------------------------------------------------------
    Altibase Export Script Utility.
    Release Version 7.1.0.0.0
    Copyright 2000, ALTIBASE Corporation or its subsidiaries.
    All Rights Reserved.
-----------------------------------------------------------------
##### USER #####
##### SYNONYM #####
##### TABLE #####
##### QUEUE #####
##### SEQUENCE #####
##### DATABASE LINK #####
##### VIEW #####
##### MATERIALIZED VIEW #####
##### STORED PROCEDURE #####
##### STORED PACKAGE #####
```

```
##### TRIGGER #####
##### LIBRARY #####
-------------------------------------------------------
  ##### The following script files were generated. #####
 1. run_il_out.sh           : [ iloader formout, data-out script ]
 2. run_is.sh               : [ isql table-schema script ]
 3. run_il_in.sh            : [ iloader data-in script ]
 4. run_is_refresh_mview.sh : [ isql materialized view refresh script ]
 5. run_is_index.sh         : [ isql table-index script ]
 6. run_is_fk.sh            : [ isql table-foreign key script ]
 7. run_is_repl.sh          : [ isql replication script ]
 8. run_is_job.sh           : [ isql job script ]
 9. run_is_alt_tbl.sh       : [ isql table-alter script ]
-------------------------------------------------------

$ ls -l
USER1_ALT_TBL.sql
USER1_CRT_DIR.sql
USER1_CRT_FK.sql
USER1_CRT_INDEX.sql
USER1_CRT_LIB.sql
USER1_CRT_LINK.sql
USER1_CRT_SEQ.sql
USER1_CRT_SYN.sql
USER1_CRT_TBL.sql
USER1_CRT_TRIG.sql
USER1_CRT_USER.sql
USER1_CRT_VIEW_PROC.sql
USER1_REFRESH_MVIEW.sql
run_il_in.sh
run_il_out.sh
run_is.sh
run_is_alt_tbl.sh
run_is_fk.sh
run_is_index.sh
run_is_job.sh
run_is_refresh_mview.sh
run_is_repl.sh
```

Execution in Object Mode

```
iSQL> CREATE USER user1 IDENTIFIED BY user1;
Create success.
iSQL> CONNECT user1/user1;
iSQL> CREATE TABLE t1(i1 INTEGER);
Create success.
iSQL> CREATE VIEW v1 AS SELECT i1 FROM t1;
Create success.
iSQL> CREATE MATERIALIZED VIEW m1 AS SELECT * FROM t1;
Create success.
iSQL> CREATE OR REPLACE PROCEDURE proc1(p1 IN INTEGER)
AS a INTEGER;
```

```
BEGIN
SELECT * INTO a FROM t1 WHERE i1 = 1;
END;
/
Create success.

$ aexport -s 127.0.0.1 -u user1 -p user1 -object user1.t1
----------------------------------------------------------------
    Altibase Export Script Utility.
    Release Version 7.1.0.0.0
    Copyright 2000, ALTIBASE Corporation or its subsidiaries.
    All Rights Reserved.
----------------------------------------------------------------
##### TABLE #####
$ ls
user1_t1_CRT.sql

$ aexport -s 127.0.0.1 -u user1 -p user1 -object user1.m1
----------------------------------------------------------------
    Altibase Export Script Utility.
    Release Version 7.1.0.0.0
    Copyright 2000, ALTIBASE Corporation or its subsidiaries.
    All Rights Reserved.
----------------------------------------------------------------
##### MATERIALIZED VIEW #####
$ ls
user1_m1_CRT.sql

$ aexport -s 127.0.0.1 -u user1 -p user1 -object user1.t1,user1.v1,user1.proc1
----------------------------------------------------------------
    Altibase Export Script Utility.
    Release Version 7.1.0.0.0
    Copyright 2000, ALTIBASE Corporation or its subsidiaries.
    All Rights Reserved.
----------------------------------------------------------------
##### TABLE #####
##### VIEW #####
##### STORED PROCEDURE #####
$ ls
user1_proc1_CRT.sql
user1_t1_CRT.sql
user1_v1_CRT.sql
```

SSL Property Setting

```
SSL_ENABLE = ON  # OFF
SSL_CA     = ${ALTIBASE_HOME}/cert/ca-cert.pem
#SSL_CAPATH = ${ALTIBASE_HOME}/cert
SSL_CERT   = ${ALTIBASE_HOME}/cert/client-cert.pem
SSL_KEY    = ${ALTIBASE_HOME}/cert/client-key.pem
SSL_CIPHER = RC4-SHA:RC4-MD5
SSL_VERIFY = ON  # OFF
```

2. altiComp

This chapter describes the altiComp utility and its features such as consistency control.

Introducing altiComp

The altiComp utility monitors the progress of replication between two Altibase databases and resolves data inconsistencies that arise during the course of replication.

altiComp compares Altibase with another HDB on a table-by-table basis, and outputs information about any inconsistencies it finds. It also has a feature for synchronizing two databases in the event of data inconsistencies.

altiComp Terms

Master Server

Master Server is a server to be corrected when a discrepancy record is found between two servers. When running altiComp, either server can be designated as master.

Master DB

The database on the master server.

Slave Server

When the discrepancy record between two servers is found, Slave Server is the server to be corrected according to the reference database. When running altiComp, either server can be designated as a slave.

Slave DB

The database on the slave server.

Different Records

Different records are records whose column values do not match based on the primary key between the Master DB and Slave DB.

There are three types of differences:

- MOSX difference: When a record based on a primary key can be found in the Master DB but not in the Slave DB.
- MOSO difference: When a record based on a primary key can be found in both the master and slave tables but the record contents are different.
- MXSO difference: When a record based on a primary key can be found in the Slave DB but not in the Master DB.

Synchronization Policy

A synchronization policy is a policy that specifies how to synchronize different records. alticomp usually treats the Master DB as the reference DB and synchronizes the Slave DB with it.

Altibase provides four synchronization policies:

- SU Policy: This policy resolves MOSO differences by updating the Slave DB with the contents of the Master DB.

- SI Policy: This policy resolves MOSX differences by inserting records from the Master DB into the Slave DB.

- MI Policy: This policy resolves MXSO differences by inserting records from the Slave DB into the Master DB.

- SD Policy: This policy resolves MXSO differences by deleting records from the Slave DB.

The SU policy, SI policy, MI policy, and SD policy are set in the altiComp environment file. Note that the MI policy and the SD policy are mutually exclusive, meaning that they cannot both be enabled at the same time.

DIFF

Creates an execution result file that identifies inconsistent records found during replication between the Master DB and the Slave DB.

SYNC

Identifies inconsistent records between the Master DB and the Slave DB, bidirectionally resolves inconsistencies according to the synchronization policy set in the altiComp environment file, and creates an execution result file including execution summary information and error information.

altiComp Environment File

An environment file is for setting options for altiComp. This file includes conneciton information, altiComp function settings, synchronization policies, etc.

How to Use altiComp

This chapter describes the altiComp environment file that contains information for running altiComp, and explains the DIFF and SYNC features.

How to Execute altiComp

To use altiComp, an altiComp environment file, which contains information about the table(s) on which DIFF or SYNC is to be executed, must first be created. The altiComp environment file will be explained in the How to Use altiComp section.

altiComp commands have the following form:

```
$ altiComp -f script_file_name
```

script_file_name : File name including the path of the environment file

If the current directory is: /user/charlie/altibase_home/altiComp:

```
/user/charlie/altibase_home/altiComp> altiComp script_file_name
```

Or

```
/user/charlie/altibase_home/altiComp> altiComp  ./script_file_name
```

How to Set altiComp Properties

Each comparison and synchronization task that is described in the environment file, has its own unique properties. The properties provide information necessary for running altiComp (Please refer to the sample.cfg file in the ALTIBASE_HOME/altiComp directory).

Rules for Setting Properties

Properties follow the format "**property name = property value**" and are case-insensitive.

The following symbols have special meanings when used in the environment file:

- " **#** " (sharp) indicates a comment and causes the remainder of the line to be ignored.
- " **{ }** " (curly braces) used to indicate that a property value spans multiple lines.
- " **;** " (semicolon) serves as a delimiter to separate multiple values.
- " **"** " (double quotation marks) are used to enclose a string (such as a user name, password, table name, or column name) that includes one or more reserved words or special characters.
- In Altibase, the following are special characters: ~, !, \@, #, $, %, \^, &, *, (,), _, +, |

Property Names

A property name consists of characters other than spaces, and identifies a property within a group.

Property Values

A property can take a single value, multiple values, or an expression.

- An expression may include blanks. Most properties follow this format:
  Ex) TABLE = EMPLOYEE
- Multiple values consist of several values separated by the "**;**" delimiter, and must be contained within "{}" if they occupy more than one line (see Example 2). The "EXCLUDE" group allows multiple values.
  Ex) EXCLUDE = ENO; DNO; ENAME
  Or EXCLUDE = {ENO; DNO; ENAME}
- Expressions are character strings, can include spaces, and must be enclosed within "{}". The "WHERE" property is an expression.
  Ex) WHERE = { ENO > '1000' and ENO < '2000' }

Data Type Support

The EXCLUDE property is used as follows to exclude a particular column or columns from altiComp targets.

Example 4) Exclude a certain column from altiComp targets, if a CLOB column exists in the EMP table.

```
TABLE = EMP
EXCLUDE = { CCC }
```

Property Options

Use the following properties to specify information for accessing the local and remote servers, comparison (DIFF) and synchronization (SYNC) tasks, and synchronization policies for inconsistent records.

DB_MASTER

This is used to set the server whose contents are to be accepted as correct if inconsistent records are found between two servers.

Sets the user name and password, the name or IP address of the server, and NLS_USE. The property values must match the information in the property file in the home directory of Altibase.

- TCP Connection:

```
DB_MASTER = altibase://sys:manager@DSN=192.188.1.1;PORT_NO=20300;NLS_USE=US7ASCII
```

- SSL Connection:

```
DB_MASTER =
altibase://sys:manager@DSN=192.188.1.1;PORT_NO=${ALTIBASE_SSL_PORT_NO};NLS_USE=US
7ASCII;CONNTYPE=6;SSL_CA=/home/altibase/cert/ca-
cert.pem;SSL_CERT=/home/altibase/cert/client-
ert.pem;SSL_KEY=/home/altibase/cert/client-key.pem
```

For more detailed information about connection string attributes for SSL, please refer to the *SSL/TLS User's Guide.*

DB_SLAVE

This is used to set the other server.

This sets the user name and password, the name or IP address of the server, and NLS_USE. The property values must match the information in the property file in the home directory of Altibase.

OPERATION

This is set to "DIFF" for a comparison task, or to "SYNC" for a synchronization task.

INSERT_TO_SLAVE

This sets the SI policy used to resolve MOSX inconsistencies. Specifies whether to insert the record in question into the Slave DB. The property value is set to "ON" to specify that the record is to be inserted, and "OFF" to specify that it is not to be inserted.

INSERT_TO_MASTER

This sets the MI policy used to resolve MXSO inconsistencies. Specifies whether to insert the 52 Utilities Manual record in question into the Master DB. The property value is set to "ON" to specify that the record is to be inserted, and "OFF" to specify that it is not to be inserted.

This property and DELETE_IN_SLAVE cannot both be set to "ON" simultaneously.

DELTE_IN_SLAVE

Sets the SD policy used to resolve MXSO inconsistencies. Specifies whether to delete the record in question from the Slave DB. The property value is set to "ON" to specify that the record is to be deleted, and "OFF" to specify that it is not to be deleted.

This property and INSERT_TO_MASTER cannot both be set to "ON" simultaneously.

UPDATE_TO_SLAVE

This sets the SU policy used to resolve MOSO inconsistencies. Specifies whether to update the record in question in the Slave DB. The property value is set to "ON" to specify that the record is to be changed, and "OFF" to specify that it is not to be changed.

CHECK_INTERVAL

This sets the interval between the completion of a SYNC operation on a table and the start of a SYNC operation on the next table. Expressed in units of ms (milliseconds).

MAX_THREAD

This specifies the maximum number of threads that can run concurrently. Set to -1 to specify an unlimited number of threads.

COUNT_TO_COMMIT

This specifies the number of changed data(INSERT, DELETE or UPDATE) to be committed at once. The default value is 1000.

FILE_MODE_MAX_ARRAY

If its value is greater than 1, altiComp writes the fetched data to a file and then starts a SYNC or DIFF operation on the file. This value is used to set the maximum size of array(s) for fetching data. altiComp fetches a number of records equal to this value and writes them to a csv file.

This option can be used to realize better performance. However, when a target table has many LOB type columns, this option may not improve performance.

This option can only be used between Altibases.

Ex) FILE_MODE_MAX_ARRAY = 1000

TABLES Group

This defines information related to target table(s). The number of descriptions in the group must equal the number of target tables, and the name of each group must correspond to the name of a table in the Master DB.

The following properties can be set:

WHERE

Sets conditions for selecting table records. This property is described in the same way as a WHERE clause of a SQL statement. Multiple values are permitted, but the ";" delimiter cannot be used to specify multiple values. Moreover, this property cannot be commented.

This applies to the comparison (DIFF) and synchronization (SYNC) functions.

EXCLUDE

This sets conditions for the projection of table records. The property may have multiple values. The specified columns are excluded from comparison and synchronization operations.
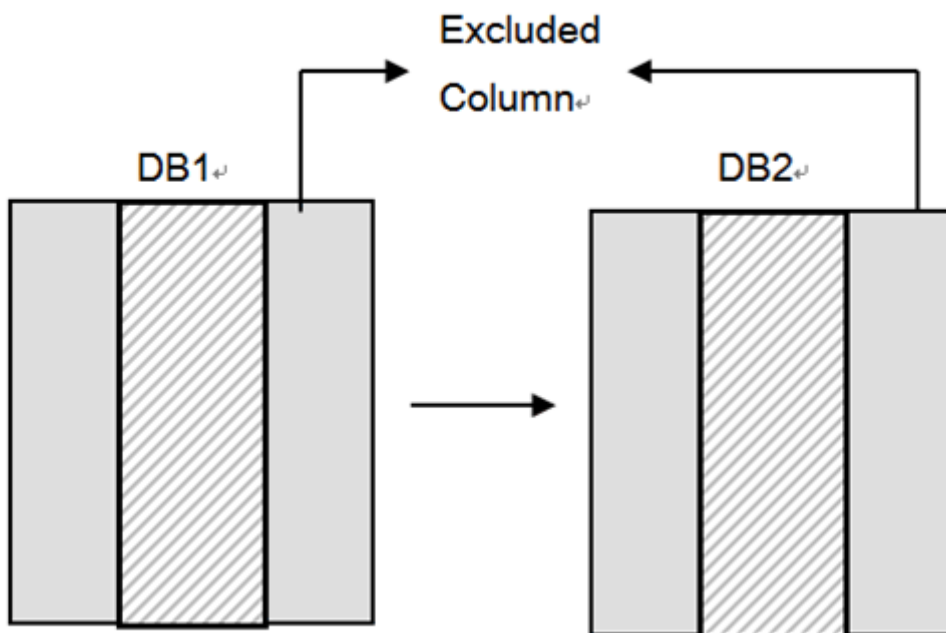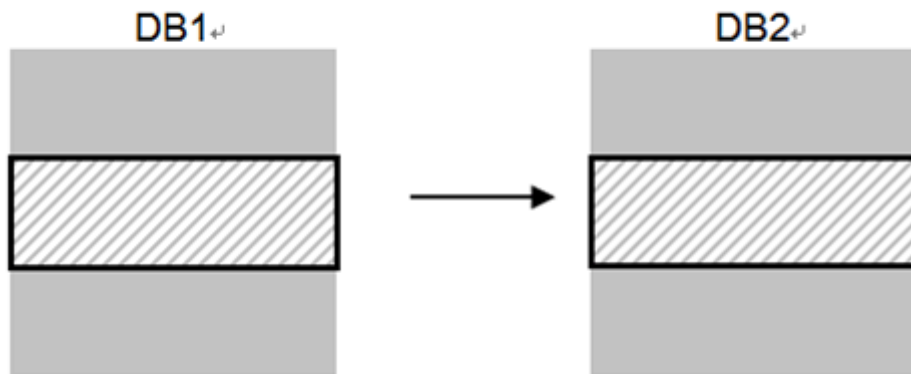
With the proper combination of WHERE and EXCLUDE, the result of combining selection and projection altiComp can be used.
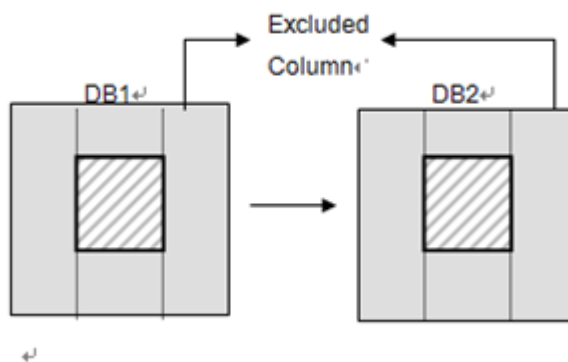
TABLE

This sets the Slave DB table name. In cases where the table names on the Master DB and the Slave DB differ from each other, this must be explicitly described in order to use the comparison (DIFF) and synchronization (SYNC) functions. if omitted, it is assumed that the table name on the Slave DB is the same as that on the Master DB.

The table name can contain Roman alphabetic characters, numbers, and the following special characters: ((space,~, !, \@, #, $, %, \^, &, *, (,), _, +, |)

However, it cannot contain Korean characters.

SCHEMA

This specifies the table schema in the Slave database

If the schema name of the connecting user of Slave is different from the schema of the target table, it must be described. If omitted, the schema of the connecting user of Slave is used.

Comparison (DIFF) Function

This function identifies different records that are found during replication between the Master and Slave databases, and creates an execution result file.

Environment File

In the altiComp environment file, set the OPERATION property to "DIFF".

All execution option properties must be specified, and the table group properties WHERE, EXCLUDE, TABLE, and SCHEMA can be optionally specified.

Execution

The comparison (DIFF) function is executed as follows:

```
$ altiComp -f script_file_name
```

script_file_name: File name including the path of the environment file

Execution Results

This function compares the contents of the Master and Slave Databases with each execution log file and table, and creates an execution result file that includes the contents of inconsistent columns of inconsistent records.

For example, if the following altiComp command is running successfully, /user/charlie/altibase_home/altiComp> altiComp sample.cfg a "mastertable-username.slavetable.log" file is created for each table in the altiComp directory, alongside sample.log.

Execution Log File

This file is created as "script_file_name.log" and displays the contents of the executed environment file with a summary of the comparison (DIFF) task for each table in the TABLES group.

The contents of the environment file are displayed as follows:

```
INFO[ MNG ] Tread #  0 init is   OK!
INFO[ MNG ] Tread #  0 start is  OK!

[TAB_2->TAB_2]
Fetch Rec In Master: 3
Fetch Rec In Slave : 2
MOSX = DF, Count :          1
MXSO = DF, Count :          0
MOSO = DF, Count :          1


 SCAN TPS:   20547.95
     Time:       0.00 sec
```

Execution Result File

This file is created as "mastertable username.slavetable.log" and displays the comparison results in the following format.

```
DF[m,n]-> COL_N (Vn_M, Vn_S):PK->{ PCOL_V }
```

- DF : The type of inconsistency (MOSX, MOSO, MXSO)
- m : The record number on the Master server
- n : The record number on the Slave server
- COL_N : The name of the first column that has different values after comparison
- Vn_M : The value in the corresponding column on the Master server
- Vn_S : The value in the corresponding column on the Slave Server

However, for records that have LOB type columns, the LOB column value is not output.

Comparison (DIFF) Examples

The following examples compare the EMP table of host1 with the EMPLOYEES table of host2, and the DEPARTMENTS table of host1 and the DEPARTMENTS table of host2.

DIFF Example 1

Specify DB_MASTER as host1 and DB_SLAVE as host2. The environment file for comparing all the records in each table should look like this:

```
DB_MASTER = "altibase://sys:manager@DSN=host1;PORT_NO=10111;NLS_USE=US7ASCII"
DB_SLAVE = "altibase://sys:manager@DSN=host2;PORT_NO=20111;NLS_USE=US7ASCII"
OPERATION = DIFF
MAX_THREAD = -1

DELETE_IN_SLAVE = ON
INSERT_TO_SLAVE = ON
INSERT_TO_MASTER = ON
UPDATE_TO_SLAVE = ON
```

```
LOG_DIR = "./"
LOG_FILE = "sample.log"

[EMP]
TABLE = EMPLOYEES
SCHEMA = SYS

[DEPARTMENTS]
TABLE = DEPARTMENTS
SCHEMA = SYS
```

The name of the target table for the Master Server (host1) and Slave Server (host2) may differ, as shown in the above example.

DIFF Example 2

From the EMP table, select the values to be compared according to the ENO column, and exclude the JOIN_DATE and GENDER columns as below.

The CONDITION property specifies that the EMP records to be compared as limited to "ENO >= 1 and ENO <= 20".

The EXCLUDE property specifies that the JOIN_DATE and GENDER columns are not to be compared.

In other words, if all columns (other than the JOIN_DATE and GENDER columns) are identical, it is assumed that the record is the same.

```
[EMP]
TABLE = EMPLOYEES
WHERE = {ENO >= 1 and ENO <= 20}
EXCLUDE = {JOIN_DATE; SEX}
[DEPARTMENTS]
```

DIFF Example 3

From the EMP table, select the values to be compared according to the ENO and JOIN_DATE columns, and exclude the GENDER column as below.

```
[EMP]
TABLE = EMPLOYEES
WHERE = {(ENO >= 1 and ENO <= 20) or (JOIN_DATE >= '20001010')}
EXCLUDE = {SEX}

[DEPARTMENTS]
```

The WHERE property determines whether the EMP records for comparison are to be limited by "ENO >= 1 and ENO <= 20" or "JOIN_DATE >= 20001010". The EXCLUDE property specifies that the GENDER column is not to be compared.

Synchronization (SYNC) Function

This function identifies records that are inconsistent between the Master and Slave databases, bidirectionally resolves the differences according to the synchronization policy in the altiComp configuration file, and creates an execution result file including execution summary information and error information.

Environment File

In the altiComp environment file, set the OPERATION property to "SYNC".

All execution option properties must be described, and the table group properties WHERE, EXCLUDE, TABLE, and SCHEMA can be optionally specified.

Execution

The synchronization (SYNC) function is executed as follows:

```
$ altiComp -f script_file_name
```

script_file_name: File name including the path of the environment file

Execution Results

This function compares the contents of the Master and Slave databases with each execution log file and table, and creates an execution result file that consists of information about synchronization tasks conducted on different records and an error log that includes information about errors which occurred during synchronization.

Execution Log File

This file is created as "script_file_name.log" and displays the contents of the executed environment file as well as the summary of the synchronization (SYNC) task for the table(s) in each TABLES group.

The contents of the environment file are written to the log file as follows:

```
INFO[ MNG ] Tread #  0 init is   OK!
INFO[ MNG ] Tread #  0 start is  OK!

[TAB_2->TAB_2]
Fetch Rec In Master: 3
Fetch Rec In Slave : 2
MOSX =  -, SI
MXSO =  -, -
MOSO =  -, SU
MXSX =  -, -


----------------------------------------
 Operation  Type      MASTER          SLAVE
----------------------------------------
 INSERT     Try          0              1
            Fail          0              0

 UPDATE     Try          X              1
```

```
            Fail            X               0

  DELETE    Try             X               0
            Fail            X               0
----------------------------------------
  UPDATE    Try             0               2
            Fail            0               0
  OOP  TPS:    13698.63
  SCAN TPS:    20547.95
      Time:        0.00 sec
```

If a failure occurs for any record, the cause of the error and the record contents are written to the log file.

Synchronization (SYNC) Examples

The following examples show how to specify OPERATION and TABLE for the synchronization policy to resolve data inconsistency.

SYNC Example 1

Insert an MOSX inconsistent record (a record that exists in the Master server, but not in the Slave server) into the Slave server, and ignore an MXSO inconsistent record (a record that exists in the Slave server, but not in the Master server).

```
Master Server = "altibase://sys:manager@DSN=host1;PORT_NO=10111;NLS_USE=US7ASCII"
Slave Server = "altibase://sys:manager@DSN=host2;PORT_NO=20111;NLS_USE=US7ASCII"
OPERATION = SYNC
MAX_THREAD = -1

DELETE_IN_SLAVE = OFF
INSERT_TO_SLAVE = ON
INSERT_TO_MASTER = OFF
UPDATE_TO_SLAVE = ON

LOG_DIR = "./"
LOG_FILE = "sample.log"

[EMP]
TABLE = EMPLOYEES
SCHEMA = SYS

[DEPARTMENTS]
TABLE = DEPARTMENTS
SCHEMA = SYS
```

The INSERT_TO_SLAVE property has been set to "ON" because the SI policy is required to resolve MOSX inconsistency. Likewise, the INSERT_TO_MASTER and DELETE_IN_SLAVE properties have been set to "OFF" because the MI and SD policies required to resolve MSXO inconsistency have been ignored.

SYNC Example 2

Insert an MOSX inconsistent record (a record that exists in the Master Server, but not in the Slave server) into the Slave server, and an MSXO inconsistent record (a record that exists in the Slave server, but not in the Master server) into the Master server.

```
Master Server = "altibase://sys:manager@DSN=host1;PORT_NO=10111;NLS_USE=US7ASCII"
Slave Server  = "altibase://sys:manager@DSN=host2;PORT_NO=20111;NLS_USE=US7ASCII"
OPERATION = SYNC
MAX_THREAD = -1

DELETE_IN_SLAVE = OFF
INSERT_TO_SLAVE = ON
INSERT_TO_MASTER = ON
UPDATE_TO_SLAVE = ON

LOG_DIR = "./"
LOG_FILE = "sample.log"

[EMP]
TABLE = EMPLOYEES
SCHEMA = SYS

[DEPARTMENTS]
TABLE = DEPARTMENTS
SCHEMA = SYS
```

The INSERT_TO_SLAVE property has been set to "ON" because the SI policy is required to resolve MOSX inconsistency. Likewise, the INSERT_TO_MASTER property has been set to "ON" because the MI policy is required to resolve MXSO inconsistency. However, the DELETE_IN_SLAVE property has been set to "OFF" because the SD policy is unnecessary.

SYNC Example 3

Synchronize the Master and Slave servers.

```
Master Server = "altibase://sys:manager@DSN=host1;PORT_NO=10111;NLS_USE=US7ASCII"
Slave Server  = "altibase://sys:manager@DSN=host2;PORT_NO=20111;NLS_USE=US7ASCII"
OPERATION = SYNC
MAX_THREAD = -1

DELETE_IN_SLAVE = ON
INSERT_TO_SLAVE = ON
INSERT_TO_MASTER = OFF
UPDATE_TO_SLAVE = ON

LOG_DIR = "./"
LOG_FILE = "sample.log"

[EMP]
TABLE = EMPLOYEES
SCHEMA = SYS

[DEPARTMENTS]
TABLE = DEPARTMENTS
```

```
SCHEMA = SYS
```

The SI and SD policies are necessary to synchronize the Master and Slave servers. Therefore, the INSERT_TO_SLAVE and DELETE_IN_SLAVE properties have been set to "ON".

SYNC Example 4

Please refer to schema.sql in the $ALTIBASE_HOME/sample/APRE/schema directory.

Synchronize the EMPLOYEES table of the local server host1 with the EMPLOYEES table of the remote server host2 (delete 16 to 20 from the ENO column), and the DEPARTMENTS table of host1 with the DEPARTMENTS table of host2.

First, set up replication between the local and remote servers.

For the local server (IP: 192.168.1.11)

```
iSQL> CREATE REPLICATION rep1 WITH '127.0.0.1', 56342 FROM sys.employees TO
sys.employees, FROM sys.departments TO sys.departments;
Create Success
iSQL>
```

For the remote server (IP: 127.0.0.1)

```
iSQL> CREATE REPLICATION rep1 WITH '192.168.1.11', 65432 FROM sys.employees TO
sys.employees, FROM sys.departments TO sys.departments;
Create Success
iSQL>
```

If the current directory is /user/charlie/altibase_home/ altiComp:

```
$ vi sample.cfg
Master Server =
"altibase://sys:manager@DSN=127.0.0.1;PORT_NO=20582;NLS_USE=US7ASCII"
Slave Server  =
"altibase://sys:manager@DSN=192.168.1.11;PORT_NO=20582;NLS_USE=US7ASCII"

OPERATION  = SYNC
MAX_THREAD = -1

DELETE_IN_SLAVE = ON
INSERT_TO_SLAVE = ON
INSERT_TO_MASTER = OFF
UPDATE_TO_SLAVE = ON

LOG_DIR = "./"
LOG_FILE = "sample.log"

[ EMPLOYEE S]
WHERE   = {ENO >= 1 and ENO <= 20}
TABLE   = EMPLOYEES
SCHEMA  = SYS
[ DEPARTMENTS ]
TABLE   = DEPARTMENTS
SCHEMA  = SYS
```

```
$ altiComp -f sample.cfg
$ cat sample.log
INFO[ MNG ] Tread #  0 init is   OK!
INFO[ MNG ] Tread #  1 init is   OK!
INFO[ MNG ] Tread #  0 start is  OK!
INFO[ MNG ] Tread #  1 start is  OK!

[DEPARTMENTS->DEPARTMENTS]
Fetch Rec In Master: 5
Fetch Rec In Slave : 5
MOSX = NO
MXSO = NO
MOSO = SU


--------------------------------------------
 Operation  Type      MASTER          SLAVE
--------------------------------------------
  INSERT     Try          0               0
             Fail         0               0

  UPDATE     Try          X               0
             Fail         X               0

  DELETE     Try          X               0
             Fail         X               0
--------------------------------------------
  UPDATE     Try          0               0
             Fail         0               0
  OOP  TPS:       0.00
  SCAN TPS:   60240.96
     Time:       0.00 sec

[EMPLOYEES->EMPLOYEES]
Fetch Rec In Master: 20
Fetch Rec In Slave : 15
MOSX = NO
MXSO = NO
MOSO = SU


--------------------------------------------
 Operation  Type      MASTER          SLAVE
--------------------------------------------
  INSERT     Try          0               5
             Fail         0               0

  UPDATE     Try          X               0
             Fail         X               0

  DELETE     Try          X               0
             Fail         X               0
--------------------------------------------
  UPDATE     Try          0               5
             Fail         0               0
  OOP  TPS:     576.04
  SCAN TPS:    2304.15
```

```
    Time:        0.01 sec
```

3. aku

Introducing aku

Overview

Altibase Kubernetes Utility (AKU) is a utility that helps you perform tasks such as synchronizing data in Altibase with the start and termination of Pods or resetting synchronization information when scaling in a Statefulset in Kubernetes. Aku supports data replication among Pods but does not support Altibase's data scale-out feature.

> StatefulSets are one of Kubernetes' workloads for supporting stateful applications like databases, and scaling means creating or terminating pods. A Pod is a resource in Kubernetes that contains containers, and Altibase server runs on these containers.

When scaling up or down on a StatefulSet, you can use aku to start or terminate pods that meet the following conditions. You should add the command in the appropriate location so that aku runs on the Altibase container.

When scaling up

You can use aku, if you want to create a Pods with the same data as Altibase server in an existing Pod.

The pod generated at first is called a "Master pod", and the pod generated by further scaling up is called a "Slave pod".

When scaling down

You can use aku, if you want to initialize the replication information of Altibase server when Pods are terminated.

Component

> ⚠️ Aku should be running with a same version in all Pods, and the aku configuration files should have same values. Basically, aku and aku configuration file are included in the Altibase container image.
>
> Altibase Server and aku should be executed within the same container.

aku

Aku is located in $ALTIBASE_HOME/bin. To execute aku, you need to set $ALTIBASE_HOME and add $ALTIBASE_HOME/bin to $PATH.

aku.conf

aku.conf is the configuration file for aku. When executing aku, it reads aku.conf file to obtain the information needed for Altibase data synchronization. Altibase provides aku.conf.sample in $ALTIBASE_HOME/conf directory. Before running aku, generate the aku.conf file in the same directory using aku.conf.sample.

The aku.conf.sample file is as follows:

⚠️ Line after # symbol is commented out.

```
# aku.conf.sample

#******************************************************************
# Copyright 2022, Altibase Corporation or its subsidiaries.
# All rights reserved.
# Property File for Altibase AKU Utility
#******************************************************************


#================================================================
# Kubernetes setting Properties
#================================================================
AKU_STS_NAME                 = altibase-sts    # statefulsets name
AKU_SVC_NAME                 = altibase-svc    # services name
AKU_SERVER_COUNT             = 4               # the number of Pods
#================================================================
# Common Properties
#================================================================
AKU_SYS_PASSWORD             = manager
AKU_PORT_NO                  = 20300
AKU_REPLICATION_PORT_NO      = 20301
AKU_QUERY_TIMEOUT            = 3600
AKU_QUERY_RETRY_COUNT        = 5
AKU_QUERY_RETRY_DELAY_MSEC   = 1000
#================================================================
# aku start/end Properties
#================================================================
AKU_ADDRESS_CHECK_COUNT      = 30
AKU_FLUSH_AT_START           = 1
AKU_FLUSH_TIMEOUT_AT_START   = 300
AKU_DELAY_START_COMPLETE_TIME = 0

AKU_FLUSH_AT_END             = 1
AKU_REPLICATION_RESET_AT_END = 1
#================================================================
# Replication Properties
#================================================================
REPLICATIONS = (
    REPLICATION_NAME_PREFIX = AKU_REP
    SYNC_PARALLEL_COUNT      = 1
    (
        SYS.T1, SYS.T2, SYS.T3
    )
)
```

For details of each properties, refer to aku-properties.

Setup to run aku

Prerequisite

To ensure stable usage of aku in a Kubernetes environment, the following conditions must be met:

- It should be used only in **StatefulSets** among Kubernetes workload.

- The **Pod management policy should be OrderedReady**. OrderedReady is the default policy for StatefulSets.
- The maximum number of scalable replicas is **up to 6.**
- **Altibase server and aku** should be executed within the same container.
- `aku -p start` command should be performed after Altibase server has started successfully.
- aku may not work normally when you execute `aku -p start` on multiple pods at the same time. To prevent this, it is required to configure the **Startup Probe** in Kubernetes.
- **Startup Probe** configuration is needed to verify if `aku -p start` command has been successfully executed. You can use the presence of the aku_start_completed file in the /tmp directory as an indicator for verification.
- Set **publishNotReadyAddress** to true.
- `aku -p end` command should be performed before stopping the Altibase server.
- A Pod should be terminated after `aku -p end` command completes successfully.
- Kubernetes's **terminationGracePeriodSeconds** should be set to a large value to allow aku to complete its tasks successfully before Pod is terminated.

Altibase Environment Variable and Properties

- Altibase Environment
  - $ALTIBASE_HOME
    - Sets the directory in which Altibase server was installed. This must be set to use aku.
- Altibase Properties
  - set ADMIN_MODE to 1.
  - set REMOTE_SYSDBA_ENABLE to 1.

aku Properties

Category	Property name	Default value	Description
Kubernetes Setting Properties	AKU_STS_NAME		The name of the StatefulSet defined in the Kubernetes object specification.
	AKU_SVC_NAME		The Service name that provides the Network Service defined in the Kubernetes object specification.
	AKU_SERVER_COUNT	4	The maximum number of Altibase servers that can be synchronized using aku. It also refers to the number of Pods that can be scaled up in Kubernetes. It can be set from 1 to 6.
Common Properties	AKU_SYS_PASSWORD		Database SYS user password
	AKU_PORT_NO	20300	Altibase Server Port number. It can be set from 1024 to 65535.

Category	Property name	Default value	Description
	AKU_REPLICATION_PORT_NO	20301	Altibase Replication Port number. It can be set from 1024 to 65535.
	AKU_QUERY_TIMEOUT	3600	It refers to the Altibase server property QUERY_TIMEOUT. If the execution time of SQL statements executed by aku exceeds AKU_QUERY_TIMEOUT value, the statement is canceled.
	AKU_QUERY_RETRY_COUNT	5	If a query executed on the Altibase server fails, it retries the specified number of times with this value. If this value is set to 0, it does not retry.
	AKU_QUERY_RETRY_DELAY_MSEC	1000 (ms)	If a query executed on the Altibase server fails, it retries after waiting the specified time with this value. . If this value is set to 0, it does not wait and retries.
aku start/end Properties	AKU_ADDRESS_CHECK_COUNT	30	The number of attempts to connect to the local IP for checking if the DNS address of the currently created Pods is registered in the Kubernetes service (indicating whether communication between internal Pods is possible) when running `aku -p start`.
	AKU_FLUSH_AT_START	1	This property determines whether replication gaps should be removed or not, during the execution of the `aku -p start` command. If this value is set to 1, replication gaps are removed by using FLUSH command. If this value is set to 0, replication gaps will not be removed.
	AKU_FLUSH_TIMEOUT_AT_START	300	This property sets the *wait_time* for the FLUSH WAIT command. If this value is set to 0, it performs FLUSH. If this value is set to 1 or greater, it performs FLUSH WAIT with the specified *wait_time*. This setting is valid only if AKU_FLUSH_AT_START is set to 1.

Category	Property name	Default value	Description
	AKU_DELAY_START_COMPLETE_TIME	0(sec)	This property specifies a waiting time (in seconds) during `aku -p start` process on the slave pod, specifically after the data synchronization is completed internally and before changing the Altibase property ADMIN_MODE to 0.
	AKU_FLUSH_AT_END	1	This property determines whether replication gaps should be removed or not, during the execution of the `aku -p end` command on the Slave Pod. If this value is set to 1, replication gaps are removed by using FLUSH ALL command. If this value is set to 0, replication gaps will not be removed.
	AKU_REPLICATION_RESET_AT_END	1	This property determines whether to perform a reset of the replication information or not by using RESET command. If this value is set to 1, the replication information is reset. If this value is set to 0, the replication information is not reset.
Replication Properties	REPLICATIONS/REPLICATION_NAME_PREFIX		This property specifies the prefix of the replication object's name that is created by aku. The maximum value is 37 bytes. Please refer to the Naming rule of the replication object in aku.
	REPLICATIONS/SYNC_PARALLEL_COUNT	1	The number of threads for sending/receiving during replication sync. It can be set from 1 to 100.

> ⓘ **Note**
>
> **Naming rule of the replication object in aku**
>
> The naming rule of the replication object in aku is as follow:
>
> *REPLICATION_NAME_PREFIX*_[*Pod Number*][*Pod Number*]
>
> - REPLICATION_NAME_PREFIX: The strings specified as REPLICATIONS/REPLICATION_NAME_PREFIX property.
> - Pod number: The number located after *pod_name*_ in the names of pods created by the Kubernetes StatefulSet. Each pod has a unique sequential number assigned to it. In the names of Altibase replication objects, the numbers of the two pods that form the replication pair are included, and the smaller number appears first.

Example) The names of the replication objects created in each Pod when
AKU_SERVER_COUNT = 4 and REPLICATION_NAME_PREFIX = "AKU_REP"

Pod Number	Replication object name	Description
pod_name-0	AKU_REP_01	Replication object name between pod_name-0 and pod_name-1
	AKU_REP_02	Replication object name between pod_name-0 and pod_name-2
	AKU_REP_03	Replication object name between pod_name-0 and pod_name-3
pod_name-1	AKU_REP_01	Replication object name between pod_name-0 and pod_name-1
	AKU_REP_12	Replication object name between pod_name-1 and pod_name-2
	AKU_REP_13	Replication object name between pod_name-1 and pod_name-3
pod_name-2	AKU_REP_02	Replication object name between pod_name-0 and pod_name-2
	AKU_REP_12	Replication object name between pod_name-1 and pod_name-2
	AKU_REP_23	Replication object name between pod_name-2 and pod_name-3
pod_name-3	AKU_REP_03	Replication object name between pod_name-0 and pod_name-3
	AKU_REP_13	Replication object name between pod_name-1 and pod_name-3
	AKU_REP_23	Replication object name between pod_name-2 and pod_name-3

⚠ Do not create/drop/modify the Altibase replication objects created by aku.

How to configure Altibase Replication Tables

Users can specify the tables to be managed in replication in the aku configuration file. To specify replication tables, users need to add table information to the replication property section in the format [user name].[table name].

Each replication table is separated by a comma, and more than one replication table can be written on a single line.

Here's an example of configuring tables *T1* to *T9* to be split and managed across three replications.

```
#===============================================================
```

```
# Replication Properties
#================================================================
REPLICATIONS = (
    REPLICATION_NAME_PREFIX = AKU_REP1
    SYNC_PARALLEL_COUNT     = 1
    (
        SYS.T1, SYS.T2, SYS.T3
    )
),
(
    REPLICATION_NAME_PREFIX = AKU_REP2
    SYNC_PARALLEL_COUNT     = 1
    (
        SYS.T4, SYS.T5, SYS.T6
    )
),
(
    REPLICATION_NAME_PREFIX = AKU_REP3
    SYNC_PARALLEL_COUNT     = 1
    (
        SYS.T7,
        SYS.T8,
        SYS.T9
    )
)
```

Usage of aku

Syntax

```
aku
[-h]
[--help]
[-v]
[--version]
[-i]
[--info]
[-p [start | stop | clean] ]
[--pod [start | stop | clean] ]
```

Parameters

-h, --help

Displays the usage of aku

-v, --version

Displays the version information of aku. It is recommended to use the same version of aku as
Altibase server.

-i, --info

Displays the following informations defined in aku.conf file.

- Altibase Server Information

  - Server ID

  - Host

  - User

  - Password

  - Port

  - Replication Port

  - Max Server Count : Maximum number of scalable Pods

- Replication Information

  - Replication object's name

- Replication Items

  - Replication Target table and User Name

-p, --pod {*pod_action*}

Specify the action to be performed with aku. *pod_action* options are "start", "end", and "clean".

pod_action with aku

The followings introduce the action performed during execution of aku.

aku -p start

It creates Altibase replication objects and synchronizes data. You can use the command when starting Pods.

The following shows the detailed behavior of `aku -p start` command.

> ⚠️ `aku -p start` command should be performed after Altibase server has started successfully.

Creation of Master Pod (*pod_name*-0)

Since Altibase replication objects need to be created on all Pods, you should execute `aku -p start` command even if creating *pod_name*-0 in a StatefulSet.

The following explains the detailed behavior of `aku -p start` command while creating *pod_name*-0 in a StatefulSet.
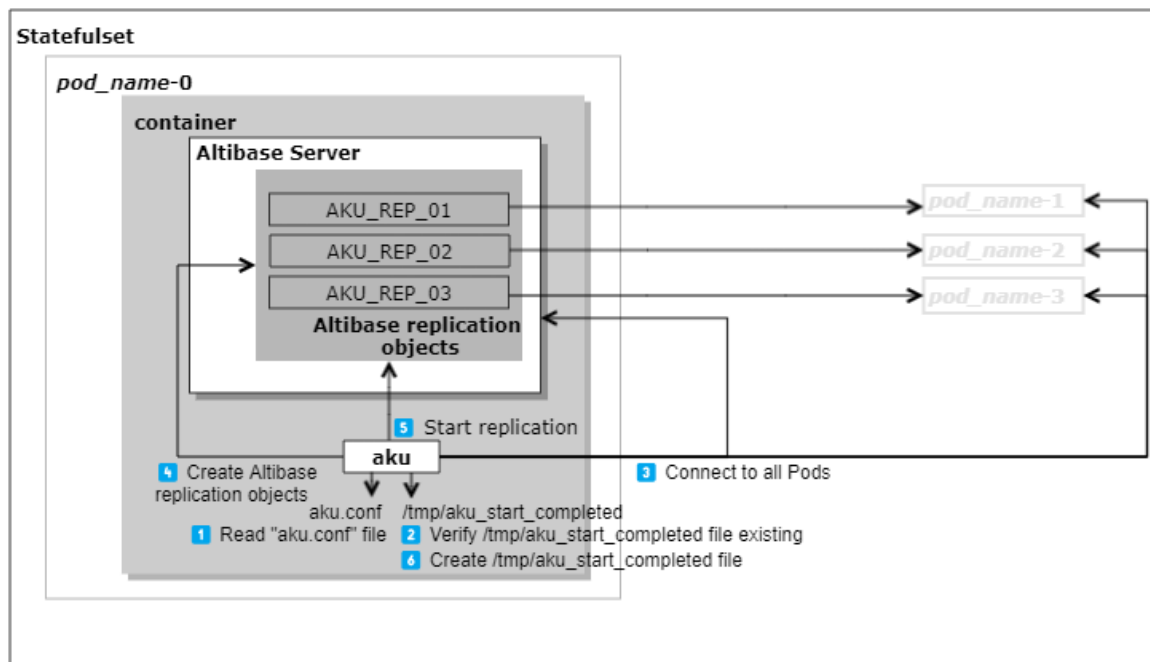
① Read "aku.conf" file.

② Verify if the aku_started_completed file exists in /tmp directory. In the usual case, if the `aku -p start` command has not been executed previously, this file does not exist. If it exists, an error message is displayed and aku terminates the process because it is considered a duplicate execution of the `aku -p start` command.

③ Connect to all pods that are the target servers for replication. Typically, as this is an initially created pod, connection attempts to other pods fail. This is the expected behavior.

④ Create the Altibase replication objects. If a replication object with the same name already exists, the replication creation step will be skipped.

⑤ Start replication associated with all pods that successfully connect from *pod_name*-0, and start replication associated with *pod_name*-0 from other connected pods. Typically, as this is the initially created pod, there are no connected pods, so this action is not performed.

⑥ Create a file named "aku_start_completed" in /tmp directory.

Scale up

When scaling up in a StatefulSet, a Slave Pod is created. A Pod can be created and terminated repeatedly. The process of `aku -p start` execution is a little different when a Pod is first created and when it is restarted after being terminated.

> **When creating a Slave Pod for the first time, or restart (Default behavior, AKU_REPLICATION_RESET_AT_END =1)**

The following explains the detailed behavior of `aku -p start` command on *pod_name*-1 when the Slave Pod is created for the first time and is restarted normally.
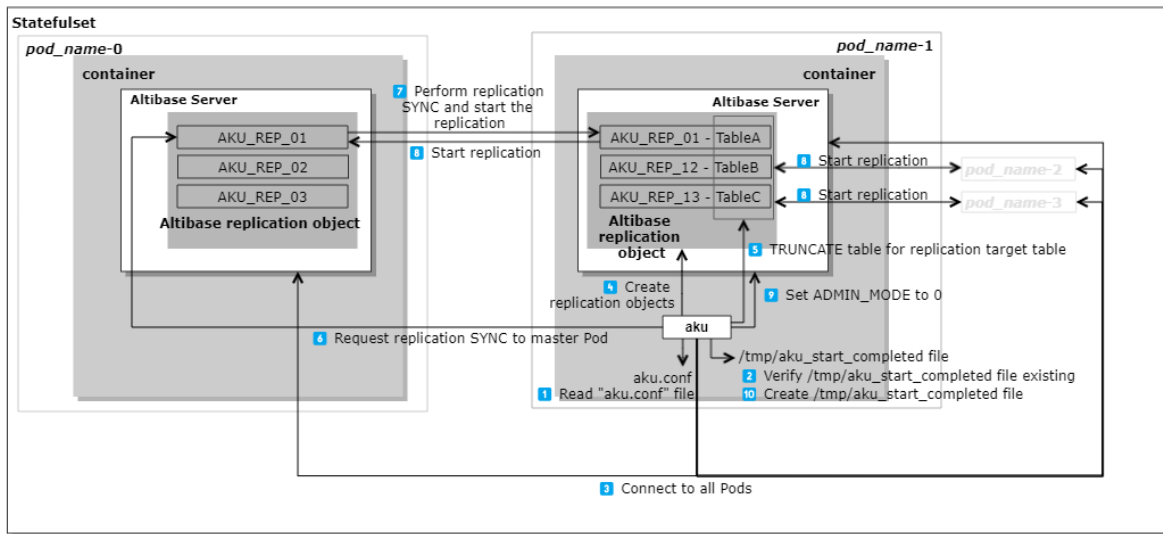
① Read "aku.conf" file.

② Verify if the aku_started_completed file exists in /tmp directory. In the usual case, if the `aku -p start` command has not been executed previously, this file does not exist. If it exists, an error message is displayed and aku terminates the process because it is considered a duplicate execution of the `aku -p start` command.

③ Connect to all pods that are the target servers for replication. Typically, only the connection with *pod_name*-0 succeeds, and the connection to *pod_name*-2 or *pod_name*-3 fails because those pods are not created yet.

④ Create the Altibase replication objects. If *pod_name*-1 is restarted pod, the replication object with the same name may exist and this step will be skipped.

⑤ Execute 'TRUNCATE table' on the replication target table in *pod_name*-1.

⑥ Request *pod_name*-0(Master Pod) to perform replication synchronization.

⑦ Perform replication synchronization from *pod_name*-0 to *pod_name*-1 and start the replication.

⑧ Start replication associated with all pods that successfully connect from *pod_name*-1, and start replication associated with *pod_name*-1 from other connected pods. Typically, only AKU_REP_01, the replication related to *pod_name*-0, starts on *pod_name*-0 and *pod_name*-1.

⑨ Set the Altibase server property ADMIN_MODE to 0 on *pod_name*-1 to allow to access for database user.

⑩ Create a file named "aku_start_completed" in /tmp directory.

Restarting Slave Pods that Haven't Reset the Replication Information

> **Case of restarting a Slave Pod that has not reset the replication information (Default behavior, AKU_FLUSH_AT_START = 1)**

The following explanation describes the basic behavior of aku when executing `aku -p start` on a Slave Pod that has not reset the replication information.

If a Pod is abnormally terminated or terminated with the AKU_REPLICATION_RESET_AT_END property sets to 0, the replication information is not initialized. If the replication information were not initialized, when restarting Pods the previous replication information would exist, so the below data synchronization actions from other nodes are skipped.

- Executes 'TRUNCATE table' on the replication target table.

- Requests for performing replication synchronization.

In this case, if the AKU_FLUSH_AT_START property sets to 1, data synchronization from itself to other nodes will be performed.

Note that if the replication information is not reset and remains, the XSN of the replication object has a value other than -1. Refer to the Cautions 4.

① Read "aku.conf" file.

② Verify if the aku_started_completed file exists in /tmp directory. In the usual case, if the `aku -p start` command has not been executed previously, this file does not exist. If it exists, an error message is displayed and aku terminates the process because it is considered a duplicate execution of the `aku -p start` command.

③ Connect to all pods that are the target servers for replication. Typically, only the connection with *pod_name*-0 succeeds.

④ Start replication associated with all pods that successfully connect from *pod_name*-1, and start replication associated with *pod_name*-1 from other connected pods. Typically, only AKU_REP_01, the replication related to *pod_name*-0, starts on *pod_name*-0 and *pod_name*-1.

⑤ Execute ALTER REPLICATION ~ FLUSH ALL command to the replications associated with all pods that successfully connect from *pod_name*-1. This command sends all unsynchronized data to other pods from *pod_name*-1.

⑥ On other connected pods, execute ALTER REPLICATION ~ FLUSH ALL command to replications related to *pod_name*-1 to send unsynchronized data. If AKU_FLUSH_TIMEOUT_AT_START is not set to 0, execute ALTER_REPLICATION ~ FLUSH WAIT *wait_time*.

⑦ Set the Altibase server property ADMIN_MODE to 0 on *pod_name*-1 to allow to access for database user.

⑧ Create a file named "aku_start_completed" in /tmp directory.

> **Case of restarting a Slave Pod that hasn't reset the replication information (AKU_FLUSH_AT_START = 0)**

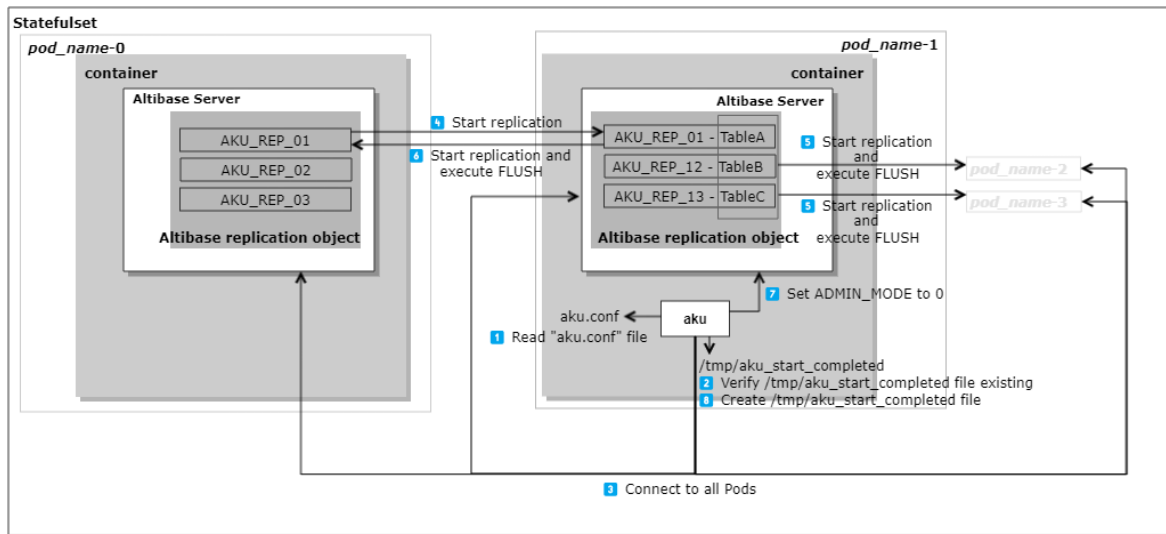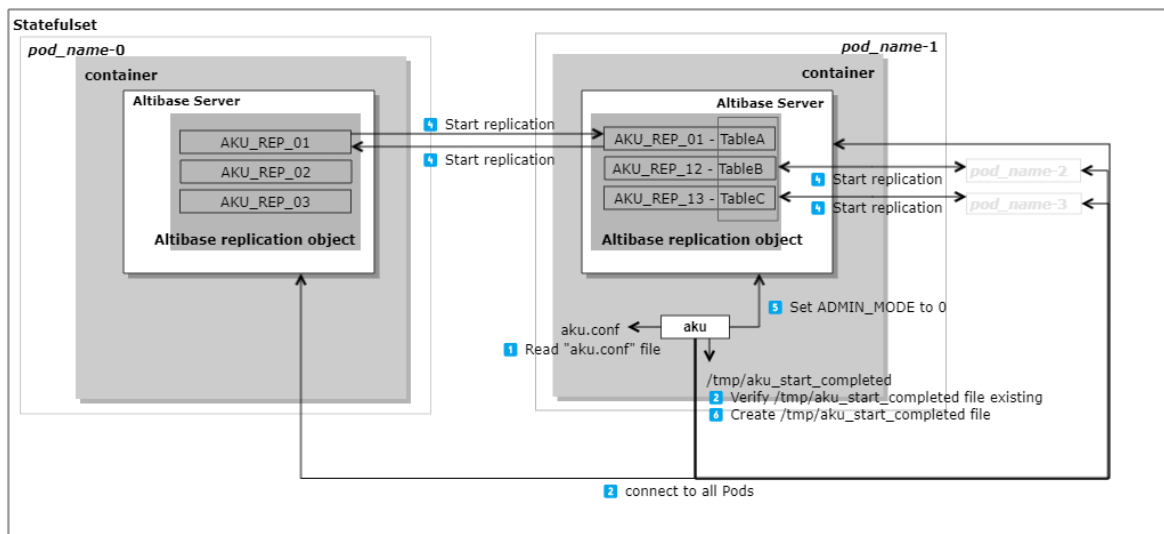The following explanation describes the behavior of aku when executing `aku -p start` on a Slave Pod that has not reset the replication information, with the property AKU_FLUSH_AT_START set to 0.

① Read "aku.conf" file.

② Verify if the aku_started_completed file exists in /tmp directory. In the usual case, if the `aku -p start` command has not been executed previously, this file does not exist. If it exists, an error message is displayed and aku terminates the process because it is considered a duplicate execution of the `aku -p start` command.

③ Connect to all pods that are the target servers for replication. Typically, only the connection with *pod_name*-0 succeeds.

④ Start replication associated with all pods that successfully connect from *pod_name*-1, and start replication associated with *pod_name*-1 from other connected pods. Typically, only AKU_REP_01, the replication related to *pod_name*-0, starts on *pod_name*-0 and *pod_name*-1.

⑤ Set the Altibase server property ADMIN_MODE to 0 on *pod_name*-1 to allow to access for database user.

⑥ Create a file named "aku_start_completed" in /tmp directory.

aku -p end

The command should be used when terminating Pods. It performs to stop Altibase replication and reset the replication information.
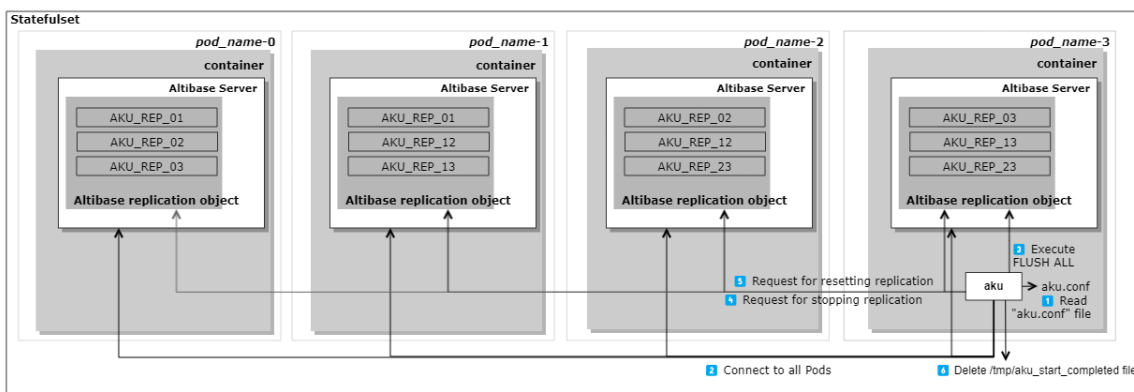
① Read "aku.conf" file.

② Connect to all Pods, which are connected with the current Pod. Since Pods are terminated sequentially, connection errors can occur when attempting to connect to already deleted Pods. This is the expected behavior.

③ Send the replication change logs to the replication objects of current Pod by executing 'ALTER REPLICATION *replication_name* FLUSH ALL'. If the AKU_FLUSH_TIMEOUT_AT_START property is set to 0, this step is skipped.

④ Request to perform "ALTER REPLICATION *replication_name* STOP" on all Pods related to current Pod.

⑤ Request to perform "ALTER REPLICATION *replication_name* RESET" on all Pods related to current Pod. If the AKU_REPLICATION_RESET_AT_END property is set to 0, this step is skipped.

> ⚠️ `aku -p end` command should be performed before stopping the Altibase server.

⑥ Delete the "aku_start_completed" file in the /tmp directory.

aku -p clean

This command deletes all Altibase replication objects on the pods and the "aku_start_completed" file in the /tmp directory. Users can use this command when synchronization between pods is no longer needed.

Cautions

1) aku.conf

- Every aku property which does not have its default value must be specified. If not, the "Property [*property_name*] should be specified by configuration." error occurs.

- Use # symbol to write a comment in the "aku.conf" file. Note that if there is no letter after # symbol, it causes the syntax error.

2) Storage corruption in Master Pod

aku does not recover data corruption due to storage corruption in Master Pod.

3) In case `aku -p start` command fails due to a master pod failure

Master pod failure refers to a scenario where the `aku -p start` command fails on the master pod under the following circumstances:

- One or more slave pods are running.

- In the master pod, some or all of the replication object information between the running slave pod(s) and the master pod is lost.

When a master pod failure occurs, the following steps should be taken to recover the master pod:

1. Synchronize data from slave pods to the master pod to ensure data consistency.

2. Start replication on the master pod.

3. Retry the command `aku -p start`.

Refer to the example below for detailed steps of master pod failure recovery.

Example of Master Pod Failure Recovery

Assuming a failure occurs in the master pod named *pod_name*-0 in the following environment:

- Configuration of the aku property in the master pod:
  - AKU_SERVER_COUNT = 4
  - REPLICATION_NAME_PREFIX = AKU_REP
- Configuration of Altibase server properties in the master pod:
  - ADMIN_MODE = 1
- Slave pod *pod_name*-1 is running.
- Replication targets tables are *T1*, *T2*, and *T3*.
- Information of replication object AKU_REP_01 (the replication between the master pod and slave pod *pod_name*-1) in the master pod is lost.

The master pod failure situation can be identified through the following log:

```
$ aku -p start
AKU started with START option.
[AKU][2024/04/19 19:21:31.012670][140276343642368] [INFO][akuRunStart:828][-][-]
Start as MASTER Pod.
[AKU][2024/04/19 19:21:31.012991][140276343642368] [ERROR][akuRunStart:1030][-]
[-] The MASTER server is detected to have failed. Check and perform a manual
recovery.
AKU failed to run.
```

At this point, when querying the XSN of the replication is lost in the master pod, the value is output as -1:

```
iSQL> SELECT REPLICATION_NAME, XSN FROM SYSTEM_.SYS_REPLICATIONS_;
REPLICATION_NAME                 XSN
---------------------------------------------------------
AKU_REP_01                        -1
1 rows selected.
```

ⓘ **Note**

XSN is the identification number of XLog, which delivers replication information between remote servers and local servers through sender and receiver threads. When initializing replication objects, this value becomes -1.
The following recovery procedures can be sequentially performed to resolve the failure of the master pod:

1. Synchronize data from slave pods to the master pod to ensure data consistency.

   1. Delete records of replication target tables on the master pod.

      To prevent conflicts during data synchronization, TRUNCATE the replication target tables managed by the master pod. To execute the TRUNCATE command, the Altibase server property REPLICATION_DDL_ENABLE must be set to 1.

      ```
      # Change the REPLICATION_DDL_ENABLE property to 1.
      iSQL> ALTER SYSTEM SET REPLICATION_DDL_ENABLE = 1;
      Alter success.

      # TRUNCATE records of replication target tables.
      iSQL> TRUNCATE TABLE T1;
      ```

```
Truncate success.
iSQL> TRUNCATE TABLE T2;
Truncate success.
iSQL> TRUNCATE TABLE T3;
Truncate success.

# Restore the REPLICATION_DDL_ENABLE property to 0.
iSQL> ALTER SYSTEM SET REPLICATION_DDL_ENABLE = 0;
Alter success.
```

2. Perform data synchronization on the slave pod.

Before synchronizing data, RESET the XSN value to initialize it. Once synchronization is complete, replication will automatically start.

```
# Stop replication on the slave pod (can be omitted if replication has
not started).
iSQL> ALTER REPLICATION AKU_REP_01 STOP;
Alter success.

# Reset the replication object between the master pod and the slave pod.
iSQL> ALTER REPLICATION AKU_REP_01 RESET;
Alter success.

# Perform replication SYNC to synchronize data consistency between the
slave pod and the master pod.
iSQL> ALTER REPLICATION AKU_REP_01 SYNC;
Alter success.
```

2. Start replication on the master pod.

Start replication on the master pod to complete the recovery of lost replication object information.

```
iSQL> ALTER REPLICATION AKU_REP_01 START;
Alter success.
```

3. Retry the `aku -p start` command.

Once recovery is completed, the master pod will have the same data as the slave pod, and the XSN value of the replication object AKU_REP_01 will be updated. Now, when the `aku -p start` command is retried, it will be processed successfully.

```
# Read aku.conf and create replication objects.
$ aku -p start
# START option means that aku -p start is executed.
AKU started with START option.
[AKU][2024/04/19 19:21:01.011887][139922191153408] [INFO][akuRunStart:828][-]
[-] Start as MASTER Pod.

# Remove replication gap between the master pod and the slave pod.
[AKU][2024/04/19 19:21:09.057372][139922191153408] [INFO][akuRunStart:891][-]
[-] Flush replications.
[AKU][2024/04/19 19:21:09.086363][139922191153408] [INFO][akuRunStart:896][-]
[-] Replication flush has ended.

# After all procedures are successfully completed, aku is terminated.
AKU run successfully.
```

4) If the situation in which Pod was force terminated before `aku -p end` command completed or terminated with the property AKU_REPLICATION_RESET_AT_END set to 0, continues for a long time

If the situation in which a Pod was force-terminated before the `aku -p end` command completed or terminated with the property AKU_REPLICATION_RESET_AT_END set to 0 continues for a long time, there is a possibility of uninitialized replication information remaining in the terminated Pod as well as in other Pods. When this happens, the other Pods do not delete the online log files that are required for replication to the terminated Pod. If the online log file accumulates a lot, it can lead to disk space exhaustion and result in Altibase server not being able to operate normally. To prevent this situation, if you notice that there are long periods of time in which a Pod was terminated before the `aku -p end` command completed, you should stop replication and initialize the replication information. Refer to the commands below.

```
ALTER REPLICATION replication_name STOP;
ALTER REPLICATION replication_name RESET;
```

Suppose *pod_name*-1 failed to complete `aku -p end` and and terminated abnormally while *pod_name*-0 and *pod_name*-1 are both operational. Let's check the XSN of SYSTEM.*SYS_REPLICATIONS* on *pod_name*-0. You can see that the XSN value of the replication object AKU_REP_01 is not -1. It means that the replication information is not initialized. AKU_REP_01 is the replication object between *pod_name*-0 and *pod_name*-1.

```
iSQL> SELECT REPLICATION_NAME, XSN FROM SYSTEM_.SYS_REPLICATIONS_;
REPLICATION_NAME              XSN
------------------------------------------------------
AKU_REP_03                    -1
AKU_REP_02                    -1
AKU_REP_01                    859070110
3 rows selected.
```

Execute the statements for stopping and resetting replication of the object(AKU_REP_01) on *pod_name*-0.

```
iSQL> ALTER REPLICATION AKU_REP_01 STOP;
Alter sucess.
iSQL> ALTER REPLICATION AKU_REP_01 RESET;
Alter sucess.
```

And then, let's check the XSN of SYSTEM.*SYS_REPLICATIONS* on *pod_name*-0. The XSN value of the replication object AKU_REP_01 was changed to -1.

```
iSQL> SELECT REPLICATION_NAME, XSN FROM SYSTEM_.SYS_REPLICATIONS_;
REPLICATION_NAME                XSN
-------------------------------------------------------
AKU_REP_03                        -1
AKU_REP_02                        -1
AKU_REP_01                        -1
3 rows selected.
```

Examples

This section introduces various examples of aku usage.

The aku log in the example codes includes the following information:

```
[AKU][Date and time][Thread number] [Message type][Code information][Target pod
name][Replication name] Message
```

aku -i

This is the result of running `aku -i` and displays the information set in aku.conf. A server with Server ID 0 is that of the first Pod created by the StatefulSet.

```
$ aku -i
  #######################
 [ Server ]
  Server ID        : 0
  Host             : AKUHOST-0.altibase-svc
  User             : SYS
  Password         : manager
  Port             : 20300
  Replication Port : 20301
  Max Server Count : 4
  #######################
 [ Replications ]
 #### Serve[ID:0] Replication list ####
  Replication Name : AKU_REP_01
  Replication Name : AKU_REP_02
  Replication Name : AKU_REP_03
 #### Serve[ID:1] Replication list ####
  Replication Name : AKU_REP_01
  Replication Name : AKU_REP_12
  Replication Name : AKU_REP_13
 #### Serve[ID:2] Replication list ####
  Replication Name : AKU_REP_02
  Replication Name : AKU_REP_12
```

```
  Replication Name : AKU_REP_23
 #### Serve[ID:3] Replication list ####
 Replication Name : AKU_REP_03
 Replication Name : AKU_REP_13
 Replication Name : AKU_REP_23
 #######################
 [ Replication Items ]
  #### Replication [Prefix:AKU_REP] item list ####
 User Name         : SYS
 Table Name        : T1

  User Name         : SYS
 Table Name        : T2

  User Name         : SYS
 Table Name        : T3
  #######################
```

aku -p start on a Master Pod

This is an output of running `aku -p start` on a Master Pod (*pod_name*-0).

```
$ aku -p start
AKU started with START option.
[AKU][2024/03/18 12:34:58.136944][140708807235840] [INFO][akuRunStart:828][-][-]
Start as MASTER Pod.
AKU run successfully.
```

The Description of the output is as below.

```
# Read aku.conf and create replication objects.
# START option means that aku -p start is executed.
AKU started with START option.

# MASTER Pod indicates the first pod.
[AKU][2024/03/18 12:34:58.136944][140708807235840] [INFO][akuRunStart:828][-][-]
Start as MASTER Pod.

# After all procedures are successfully completed, aku is terminated.
AKU run successfully.
```

aku -p start on the 4th Pod

This is an output of `aku -p start` command on the 4th Pod (*pod_name*-3).

```
AKU started with START option.
[AKU][2024/03/18 14:01:59.604647][140678415444224] [INFO][akuRunStart:903][-][-]
Start as SLAVE Pod.
[AKU][2024/03/18 14:02:01.005068][140678415444224] [INFO][akuRunStart:959][-][-]
Truncate tables for replications.
[AKU][2024/03/18 14:02:01.025100][140678415444224] [INFO][akuRunStart:964][-][-]
Table truncation has ended.
[AKU][2024/03/18 14:02:01.025877][140678415444224] [INFO][akuRunStart:975][-][-]
Sync tables from MASTER Server.
[AKU][2024/03/18 14:02:05.045135][140678415444224] [INFO][akuRunStart:980][-][-]
Replication sync has ended.
AKU run successfully.
```

The Description of the output is as below.

```
# Read aku.conf and create replication objects.
# START option means that aku -p start is executed.
 AKU started with START option.

# SLAVE Pod indicates other pods but the first pod.
[AKU][2024/03/18 14:01:59.604647][140678415444224] [INFO][akuRunStart:903][-][-]
Start as SLAVE Pod.

# To prvent record conflict during SYNC processing, deletes all records on the
target table.
[AKU][2024/03/18 14:02:01.005068][140678415444224] [INFO][akuRunStart:959][-][-]
Truncate tables for replications.
[AKU][2024/03/18 14:02:01.025100][140678415444224] [INFO][akuRunStart:964][-][-]
Table truncation has ended.

# MASTER Server indicates the Altibase server of the Master Pod. The data has
been synchronized from the server to the local pod.
[AKU][2024/03/18 14:02:01.025877][140678415444224] [INFO][akuRunStart:975][-][-]
Sync tables from MASTER Server.
[AKU][2024/03/18 14:02:05.045135][140678415444224] [INFO][akuRunStart:980][-][-]
Replication sync has ended.

# After all procedures are successfully completed, aku is terminated.
AKU run successfully.
```

aku -p end on the 4th Pod

This is an output of `aku -p end` command with the property AKU_REPLICATION_RESET_AT_END set to 1 on the 4th Pod (*pod_name*-3). You can see that replication FLUSH and RESET commands are executed.

```
$ aku -p end
AKU started with END option.
[AKU][2024/03/18 14:02:49.246961][139626938108160] [INFO][akuRunEnd:1090][-][-]
Start as SLAVE Pod.
[AKU][2024/03/18 14:02:49.247094][139626938108160] [INFO][akuRunEnd:1095][-][-]
Flush replications.
[AKU][2024/03/18 14:02:49.247731][139626938108160] [INFO][akuRunEnd:1100][-][-]
Replication flush has ended.
[AKU][2024/03/18 14:02:52.001848][139626938108160] [INFO][akuRunEnd:1114][-][-]
Reset replications.
[AKU][2024/03/18 14:02:52.014300][139626938108160] [INFO][akuRunEnd:1119][-][-]
Replication reset has ended.
AKU run successfully.
```

The Description of the output is as below.

```
# Read aku.conf, stop the replication and reset the replication.
# END option means that aku -p end is executed.
AKU started with END option.

# SLAVE Pod indicates other pods but the first pod.
[AKU][2024/03/18 14:02:49.246961][139626938108160] [INFO][akuRunEnd:1090][-][-]
Start as SLAVE Pod.

# FLUSH unsent changes to other pods.
[AKU][2024/03/18 14:02:49.247094][139626938108160] [INFO][akuRunEnd:1095][-][-]
Flush replications.
[AKU][2024/03/18 14:02:49.247731][139626938108160] [INFO][akuRunEnd:1100][-][-]
Replication flush has ended.

# Reset all replication objects created on the aku of the local pod.
[AKU][2024/03/18 14:02:52.001848][139626938108160] [INFO][akuRunEnd:1114][-][-]
Reset replications.
[AKU][2024/03/18 14:02:52.014300][139626938108160] [INFO][akuRunEnd:1119][-][-]
Replication reset has ended.

# After all procedures are successfully completed, aku is terminated.
AKU run successfully.
```

4. Other Utilities

altiAudit

About altiAudit

When auditing is performed on the Altibase server, the $ALTIBASE_HOME/trc directory is the default location for where the audit log file is created and audit logs are written; this location can be changed with the AUDIT_LOG_DIR property. Audit logs in the audit log file are written in binary format and are therefore, illegible by the user. The altiAudit utility converts and prints the audit log file in text format to enable the user to analyze them.

```
altiAudit [-s] {audit_log_file_name}
```

Syntax
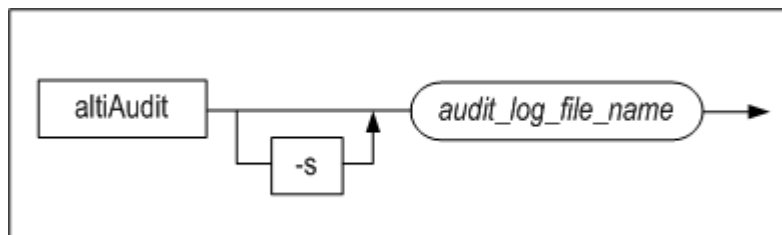
Descriptions

This converts and outputs audit logs written by the server in text format.

With the -s option, audit logs can also be printed in CSV format.

Examples

The following command prints audit logs in plain text format.

```
$ altiAudit $ALTIBASE_HOME/trc/alti-1366989680-0.aud
```

The results are printed as below:

```
[2015/03/05 14:59:29]
Session Info
  User Name        = SYS
  Session ID       = 1
  Client IP        = 127.0.0.1
  Client Type      = CLI-64LE
  Client App Info  = isql
  Action           = INSERT
  Auto Commit      = 1                         (0:non-autocommit 1:autocommit)

Query Info
  Statement ID     = 65540
  Transaction ID   = 150657
```

```
   Execute result    = 4                     (0:failure 1:rebuild 2:retry
3:queue empty 4:success)
   Fetch result      = 2                     (0:failure 1:success 2:no result
set)
   Success count     = 1
   Failure count     = 0
   Return code      = 0x02000
   Processed row     = 1
   Used memory       = 0                     bytes
   XA flag           = 0                     (0:non-XA 1:XA)

Query Elapsed Time
   Total time        = 0
   Soft prepare time = 0
   Parse time        = 0
   Validation time   = 0
   Optimization time = 0
   Execution time    = 0
   Fetch time        = 0

SQL
--------------------------------------------------------------------------------
insert into t1 values ('aaaa', 1)
--------------------------------------------------------------------------------
```

The following command prints audit logs in CSV format:

```
$ altiAudit -s $ALTIBASE_HOME/trc/alti-1366989680-0.aud
```

The results are printed as below:

```
1425535169,SYS,1,127.0.0.1,CLI-
64LE,isql,INSERT,1,65540,150657,4,2,1,0,1,0,0,0,0,0,0,0,0,0,"insert
into t1 values ('aaaa', 1)"
```

Output

In the output, each field has the following meaning:

Field Nam	Type	Description
Session Info		
User Name	String	The name of the user connected to the session
Session ID	INTEGER	The session ID
Client IP	String	The client IP address
Client Type	String	The connected client type

Field Nam	Type	Description
Client App Info	String	The application information
Action	String	The executed statement type
Auto Commit	INTEGER	0: Non-auto commit mode 1: auto commit mode
Query Info		
Statement ID	INTEGER	The statement ID
Transaction ID	INTEGER	The transaction ID
Execute result	INTEGER	The execution result: 0: failure 1: rebuild 2: retry 3: query empty 4: success
Fetch result	INTEGER	The fetch result: 0: failure 1: success 2: no result set
Success count	INTEGER	The number of times the statements conforming to the auditing condition succeeds. For the BY SESSION condition: the accumulated number of times the statements conforming to the auditing condition succeeds. For the BY ACCESS condition: if the statement conforming to the auditing condition succeeds, 1 is output.
Failure count	INTEGER	The number of times the statement conforming to the auditing condition fails. For the BY SESSION condition: the accumulated number of times the statements conforming to the auditing condition fails. For the BY ACCESS condition: if the statement conforming to the auditing condition fails, 1 is output.
Return code	INTEGER	The result code of the executed statement conforming to the auditing condition. The execution result is only output for the BY ACCESS condition.
Processed row	INTEGER	The number of processed records
Used memory	INTEGER	The memory usage (to be extended in the future)
XA flag	INTEGER	0: Non-XA 1: XA
Query Elapsed Time		
Total time	BIGINT	The total time consumed in query execution
Soft prepare time	BIGINT	The time consumed in preparing
Parse time	BIGINT	The time consumed in parsing

Field Nam	Type	Description
Validation time	BIGINT	The time consumed in validation
Optimization time	BIGINT	The time consumed in optimization
Execution time	BIGINT	The time consumed in execution
Fetch time	BIGINT	The time consumed in fetching
SQL		
	INTEGER	The executed SQL statement

altibase

About altibase

altibase is the executable server file that controls all Altibase services.

```
altibase {-v|n}
```

Syntax
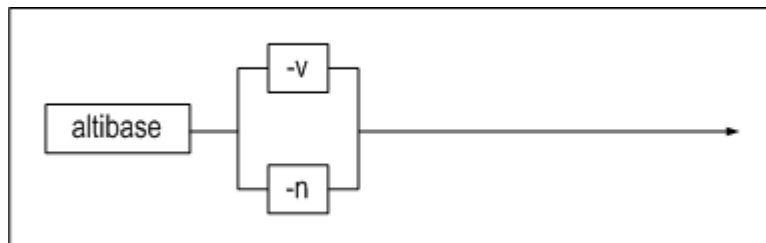
Parameters

Parameter	Description
-v	Displays the version of Altibase that is currently installed
-n	Executes Altibase in the foreground

Description

altibase is the executable server file that controls all Altibase services.

To start up or shut down Altibase normally in a production environment, do not use this command. Instead, log into iSQL in SYSDBA mode and use the startup or shutdown command, or use the server command. The server command is actually a shell comprising several commands related to starting up and shutting down the server. For more information about the server utility, please refer to server in this document.

For a complete explanation of how to start up and shut down Altibase, please refer to the *iSQL User's Manual* or the *Altibase Getting Started Manual*.

When the Altibase server process is started using iSQL, it runs in the background. In contrast, when the altibase command is executed at the shell prompt with the -n option, Altibase is executed in the foreground. This is used only for Altibase debugging purposes, and is not intended or recommended for live deployment, that is, for use in a production environment.

Running the altibase executable file with the -v option does not start up the server, but merely outputs information about the currently installed version of Altibase.

References

Please refer to the *Altibase Getting Started*, the *Altibase Administrator's Manual*, and the *Altibase iSQL User's Manual*

altiMon

About altiMon

altimon.sh monitors the status of the Altibase server process and the related host system.

altimon.sh {start | stop}

Syntax
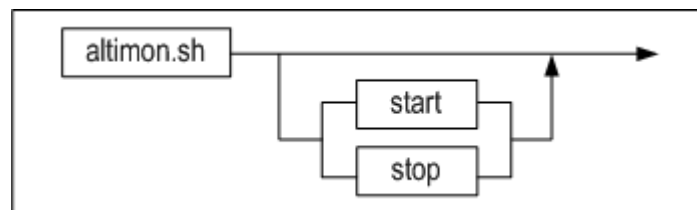
Parameters

Parameter	Description
start	Run altimon
stop	Terminates altimon

Description

altiMon consistently monitors Altibase server and the related host system in order to record the collected data into the log files.

altiMon mainly monitors information on operating system and database, and refer to the Setting altiMon Configuration for in-depth information.

start

1. Set the JAVE_HOME environment variables.
2. Run the command.

- Unix Platforms

```
$ altimon.sh start
```

3. If operation fails, verify the file $ALTIBASE_HOME/altiMon/logs/altimon.log.

stop

- Unix Platforms

```
$ altimon.sh stop
```

Operating System

altiMon uses PICL library written in C language in order to collect information on operating system. The PICL library is available on the operation systems describe in the chart below.

OS	CPU	PICL Library
AIX		
AIX 5.3 AIX 6.1 AIX 7.1	PowerPC	aix-ppc64-5.so
HP-UX		
HP-UX 11.31	Itanium (IA-64)	hpux-ia64-11.sl
Linux		
Red Hat Enterprise Linux 6.0	x86-64	linux-x64.so
Red Hat Enterprise Linux 6.5	PowerPC	linux-ppc64.so
Red Hat Enterprise Linux 7.2	PowerPC (Little Endian)	linux-ppc64.so

it can also be used after checking whether the PICL for the lower version works on the OS version that is not supported.

```
$ cd $ALTIBASE_HOME/altiMon # java -Dpicl="<picl_lib_file>" -jar
lib/com.altibase.picl.jar
```

Example)

This procedures verifies that the "aix-ppc64-5.so" PICL library is available on the unsupported version of the AIX operating system and runs altimon

1. Verify that PICL for lower versions works.

```
$ cd $ALTIBASE_HOME/altiMon
$ java -Dpicl="aix-ppc64-5.so" -jar lib/com.altibase.picl.jar
```

2. After the general operation is verified, open the $ALTIBASE_HOME/bin/altimon.sh file and set the PICL file in the PICL_LIB variable.

```
PICL_LIB=-Dpicl="aix-ppc64-5.so"
```

3. Run altimon

```
$ altimon.sh start
```

Notes

altiMon requires Java 8 version or later to operate.

The user should select the Java version equivalent to the number of bits of the PICL C library. For example, if the PICL C library is linux x64, 64bit Java should be used.

Setting altiMon Configuration

It is required that the following files located in the $ALTIBASE_HOME/altiMon/conf directory should be configured in order to properly use altiMon

- config.xml
- Metrics.xml
- GroupMetrics.xml

config.xml

This file configures Altibase access and altiMon control information.

Tag Name	Status	Description
<Altimon Name='String' monitorOsMetric="true|false">	Mandatory	The monitorOsMetric attribute specifies whether or not to measure the OsMetrics. Default Value: TRUE It should be set to false when there is no PICL C library that can be compatible to the user environment.
<DateFormat>	Optional	tag. Date and time format when recording logs. Default Value : yyyy-MM-dd HH:mm:ss Refer to the Java Document for the date format which is available to select.
<Interval>	Optional	Data collection cycle. Default Value : 60(sec) The specified values are applied unless intervals are specified in the settings, or .However, the is not affected.

Tag Name	Status	Description
<CpuSamplingInterval>	Optional	This is the execution cycle of the thread that measures CPU utilization (%). The default value is 3, and the unit is second. Since Altibase 7.1.0.8.4, OS CPU utilization measurement thread and ALTIBASE HDB CPU utilization measurement thread have been added to measure CPU utilization. Each thread measures the CPU usage rate at the CpuSamplingInterval cycle, and these values are referred to in the monitoring element. Since the OS CPU utilization measurement thread and the ALTIBASE HDB CPU utilization measurement thread operate as separate threads, a time difference may occur between the two measurement values when the CPU is overloaded.
<LogDir>	Optional	tag should be specified when using an extra disk;otherwise, the following directory would be set to default as following: $ALTIBASE_HOME/altiMon/logs
<MaintenancePeriod>	Optional	Log file maintenance period Default Value : 3 days
<Target Name='String'>	Mandatory	This tag indicates the monitoring target database. Name: name
<HomeDirectory>	Optional	tag sets ALTIBASE_Home directory as an absolute path. If it is not specified, the ALTIBASE_HOME environment variable will be used.
<DBConnectionWatchdogCycle>	Optional	This tag indicates the execution cycle of DB connection watchdog.DB connection watchdog enables monitoring target database to continue monitoring by periodically attempting to access to the database even after shutting down. Default Value : 60(sec)
<User>	Optional	Connected user. If this file is not specified, altiMon will connect via SYS user.

Tag Name	Status	Description
<Password Encrypted="Yes | No">	Mandatory	When the encrypted attributes is "No", and even if the user registered the password, altiMon alters the encrypted attributes to "YES", and thus, the password is also encrypted.
<Port>	Mandatory	Port number
<NLS>	Mandatory	NLS_USE
<DbName>	Optional	Database Name Default Value : mydb
<IPv6>	Optional	IPv6 status of use Default Value: false

Metrics.xml

The following categories provide information on the Metrics.xml file which can configure the predefined OS Metrics, and SQL Metrics, as well as user-defined OS Metrics(Command Metric).

Tag Name	Description
<OSMetric Name='PROC_CPU_USER' Activate='true' Interval='30' Logging='true'>	This tag configures the predefined OS Metric and uses predefined OS Metrics as in the following. TOTAL_CPU TOTAL_CPU_USER TOTAL_CPU_KERNEL PROC_CPU PROC_CPU_USER PROC_CPU_KERNEL TOTAL_MEM_FREE TOTAL_MEM_FREE_PERCENTAGE PROC_MEM_USED (=> RSS) PROC_MEM_USED_PERCENTAGE SWAP_FREE SWAP_FREE_PERCENTAGE DISK_FREE DISK_FREE_PERCENTAGE DISK_FREE or DISK_FREE_PERCENTAGE must have the tag as belows, and the 'Disk Name' should be uniquely named. <OSMetric Name='DISK_FREE' Activate='true'> <Disk Name='disk1'>/home</Disk> </OSMetric> <OSMetric Name='DISK_FREE' Activate='true'> <Disk Name='disk2'>/home2</Disk> </OSMetric>

Tag Name	Description
<SQLMetric Name='MEM_DATABASE_USAGE' Activate='true' Interval='60' Logging='true'>	This configures the SQL Metric, and it is required to define a Name. Name: Metric name (Required) Activate: Status of operation(Data Collection) The specifiable value is either true or false, and default value is true. Interval: If the cycle of data collection is not specified, the specified values in the config.xml file is used. Logging: It sets whether or not to log the result of data collection into the log file. The default value is set to true. Set to false if you want to record only the alert info.
<CommandMetric Name='MEM_VSZ' Activate='true' Interval='60' Logging='true'>	This tag configures the user-defined OS Metric. The altiMon executes a command or script specified in the tag, and the value output as stdout is used as measured value.
<Query>	This tag configures a query, and it is necessary in case of SQL Metric.
<Command>	The tag configures command or script file, and it is required for CommandMetric. The script file can be set up by an absolute or relative path. The relative path is based up on the following directory: $ALTIBASE_HOME/altiMon
<Alert Activate='true' ComparisonColumn='ALLOC_MEM_MB' ComparisonType='gt'>	Warning setup optional. The measured values which are compared with the threshold values should be numbers. Activate: Operation status, optional. (Default value: True) Comparison: Column the target column which will be compared with threshold value. (Optional) ComparisonType: comparison operator. (Required) -eq: is equal to the threshold value -ne: is not equal to the threshold value -gt: is greater than the threshold value -ge: is greater than or equal to the threshold value -lt: is less than the threshold value -le: is less than or equal to the threshold value
<WarningThreshold Value='500'> or <CriticalThreshold Value='800'>	This tag can specify the threshold values. (Possible to set 'Critical' and 'Warning' level) In case of , it is required. Value: threshold values

Tag Name	Description
<ActionScript>db_usage.sh</ActionScript>	This tag configures the script file name to be executed when the threshold value is out of the specified range. The script file should be existing in the following directory : $ALTIBASE_HOME/altiMon/action_scripts

GroupMetrics.xml

The GroupMetrics.xml is a file defining the Group Metrics, which is comprised of OS Metric, Command Metric, and SQL Metric. The GroupMetric can be defined by using metrics specified in Metrics.xml.

Unlike the common Metric measured values are stored to the *.log file, the data collected by group metric are stored to CSV files.

Tag Name	Description
<GroupMetric Name='group1' Interval='30'>	Configure Group Metric. Name: metric name (Required) Activate: Data collecttion status (Optional). Available value is either true or false. The default value is true. Interval: Data collection cycle (Necessary)
<Target MetricName='PROC_CPU_USER'/>	This specifies the base metric which will be included in the group metric. MetricName: The name of OSMetric, Command Metric or SQLMetric whick is defined in Metrics.xml. A SQL Metric should necessarily retrieve one row of a SELECT query. When specifying DISK_FREE, or DISK_FREE_PERCENTAGE, a character string which is linked with 'Metric Name' and 'Disk Name' with '.' should be specified as a MetricName. <Target MetricName='DISK_FREE.disk1'/>
<Column Name='LOG_GAP' />	In the case of the SQL Metric, the columns which will be included in the group metric are specified by this tag. If this tag is not specified, every select target will be included in the Group Metric. Name: The target column used in a SELECT query. If you specify an alias, the alias should be used.

Others

- action_scripts directory
  In order to normally execute a script file specifying the tag, the script file should be located in the directory $ALTIBASE_HOME/altiMon/action_scripts.
  For example, 'db_usage.sh' file in db_usage.sh should be located under the action_scripts directory in order for a script file to normally executes.

Example

The followings are examples of the altiMon configuration files.

config.xml

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

<config>
    <Altimon Name='rnd1'>
        <!-- <LogDir>/home/bethy/arugs/logs</LogDir> -->
        <DateFormat>yyyy-MM-dd HH:mm:ss</DateFormat>
        <MaintenancePeriod>3</MaintenancePeriod>
        <Interval>60</Interval>
    </Altimon>
    <Target Name='Altibase1'>

<HomeDirectory>/home/bethy/work/altibase_trunk/altibase_home</HomeDirectory>
        <User>sys</User>
        <Password Encrypted="NO">manager</Password>
        <Port>20020</Port>
        <DbName>mydb</DbName>
        <NLS>KSC5601</NLS>
        <IPv6>FALSE</IPv6>
    </Target>

</config>
```

OSMetrics.xml

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

<Metrics>
  <OSMetric Name='TOTAL_CPU' Activate='true' Description='TOTAL_CPU'>
  <OSMetric Name='PROC_CPU' Activate='true' Interval='60'>
    <Alert Activate='true' ComparisonType='gt'>
        <WarningThreshold Value='80'>
            <ActionScript>cpu_act.sh</ActionScript>
        </WarningThreshold>
    </Alert>
  </OSMetric>

  <SQLMetric Name='MEM_DATABASE_USAGE' Activate='true' Interval='30'>
    <Query>select
            trunc(mem_alloc_page_count*32/1024, 2) as alloc_mem_mb,
            trunc(mem_free_page_count*32/1024, 2) as free_mem_mb
            from v$database</Query>
    <Alert Activate='true' ComparisonColumn='ALLOC_MEM_MB' ComparisonType='GT'>
        <CriticalThreshold Value='8000' >
            <ActionScript>db_usage.sh</ActionScript>
        </CriticalThreshold>
    </Alert>
  </SQLMetric>
```

```
  <CommandMetric Name='MEM_VSZ'>
    <Command>scriptsDir/vsz.sh</Command>
    <Alert Activate='true' ComparisonType='gt'>
        <WarningThreshold Value='100000000'> <!-- in kB -->
            <ActionScript>mem_act.sh</ActionScript>
        </WarningThreshold>
    </Alert>
</CommandMetric>
</Metrics>
```

GroupMetrics.xml

```
<GroupMetrics>
    <GroupMetric Name='group1' Interval='30'>
        <Target MetricName='TOTAL_CPU'/>
        <Target MetricName='PROC_CPU'/>
        <Target MetricName='LOGFILE_COUNT'>
            <Column Name='LOG_GAP' />
        </Target>
    </GroupMetric>
    <GroupMetric Name='group2' Interval='60'>
        <Target MetricName='PROC_MEM_USED_PERCENTAGE'/>
        <Target MetricName='MEM_VSZ'/>
        <Target MetricName='LOGFILE_COUNT'/>
    </GroupMetric>
</GroupMetrics>
```

Output Items

The items are outptted with the folllowing format in the directories below.

logs directory

- altimon.log
  This is the file recording all sorts of logs(info, warn, error) by altiMon daemon.

- alert.log
  This file records the data relevant to the setting.

- OsMetrics.log
  This file records all the collected data by all the OS Metric.

- [SQLMetric_Name].log
  The collected data by SQL Metric are recorded into a different log file for each Metric.

- [GroupMetric_Name].csv
  The collected data by Group Metric are recorded into a different csv file for each Metric.

- report.html
  This file reports user configuration setting with a HTML format .

logs/archive directory

All the files in the logs directory except altimon.log, and *.csv are backed up into a archive/YYYY-MM-DD directory every day.

Among above directories, the directories with expired maintenance period specified in are removed at 1:50am every day.

logs/csv_backup directory

For Group Metrics, collected data(metric values) are constantly added to the same csv file unless the settings are modified.

However, in case of the changed metric settings meaning that when the target objects in the group metric are changed, the existing csv file is backed up into the logs/csv_backup directory, and data are recorded in a new file with the identical name.

altierr

About altierr

altierr searches for and displays detailed descriptions of Altibase server errors. Errors can be looked up using the error number, or a character string can be used as a search term and sought for within the error messages.

```
altierr {-w keyword pattern | [-n] error number}
```

Syntax

Parameters

Parameter	Description
-w	Searches for error messages containing the specified search term. All error messages that contain the search term will be displayed.
-n	Searches for an error corresponding to the specified error code number. The error code number can be a hexadecimal number, a positive integer, or a negative integer. Only the record that matches the error code number, if any, will be displayed. When searching for an error using the error code number, the numeric parameter indicator (-n) can be safely omitted.

Description

altierr searches the Altibase errors for strings that contain the specified error message or that match the specified error code number and displays the detailed description of any error that is found. The detailed description of the error includes the error code number, the error code string, the description, the cause of the error, and the action that the user must take in order to remedy the error. When an error occurs, the Altibase server writes the corresponding error code to altibase_boot.log in the following format:

ERR-error code

altierr can be used to search for the detailed description using either a hexadecimal or decimal error code, as shown below:

```
For 'ERR-00015'
$ altierr 0x00015
$ altierr -w 00015
$ altierr 21
```

When SQL-related errors occur in applications written using the C/C++ precompiler or applications that use ODBC, the error code will be set in the SQLCODE variable, or will be returned by the ODBC function. In these cases, the error code will be a negative integer. To search for the description of the corresponding error, use altierr as follows:

```
For -266286
$ altierr -266286
$ altierr 266286
$ altierr 0x4102E
```

altierr can be used to search the text of error messages for a search term. In this case, multiple records may be returned. Use a character string as a search term for searching the text of error message descriptions as follows:

```
$ altierr -w connect
$ altierr -w "does not"
```

References

Please refer to the *Altibase Error Message Reference.*

altipasswd

About altipasswd

altipasswd modifies the $ALTIBASE_HOME/conf/syspassword file. When database is not in service status, administrative job is executed by running iSQL with SYSDBA option. In this case, the password of SYS user is verified by reading the syspasswd file. Therefore, when SYS user's password is modified by using ALTER USER SQL statement, password of syspassword file should be equally modified by using altipasswd. If the password of the SYS user and syspassword are not identical, error will occur when SYSDBA tasks, such as starting up and shutting down the database, are performed.

```
altipasswd
```

Syntax

Description

Changes the password of the SYS user.

Example

To change the password of the SYS user from "manager" to "manager1234", type the following at a shell prompt:

```
$ altipasswd
Previous Password : manager
New Password : manager1234
Retype New Password : manager1234
```

altiProfile

About altiProfile

Altibase can write information about tasks that are executed on the server and server status to files for analysis. A file that contains information about server status is called a profile. altiProfile can convert a profile to character format or print STATEMENT statistics. The user can analyze the system status with this information.

```
altiProfile [-stat query|session] {profile_name [profile_name2 [profile_name3]
...}
```

Syntax
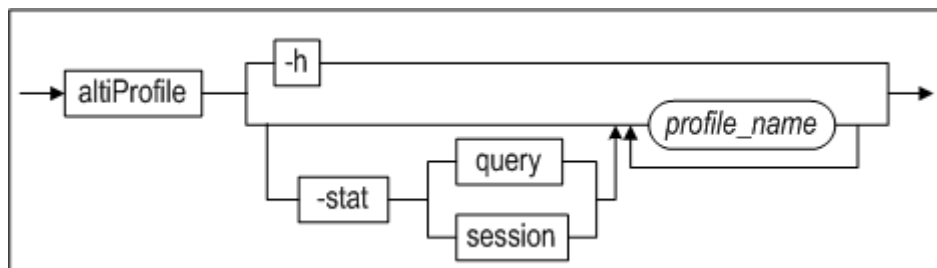
Parameters

Parameter	Description
-h	Displays help.
-stat query/session	Builds statistics from profile STATEMENTs and prints them in text and CSV formats. For more detailed information about statistics, please refer to How to use altiProfile.

Description

altiProfile converts a server profile to character format or prints STATEMENT statistics.

Example

```
iSQL> ALTER SYSTEM SET QUERY_PROF_FLAG = 1;
Alter success.
iSQL> ALTER SYSTEM SET TIMED_STATISTICS = 1;
Alter success.
iSQL>    --(Execute an SQL query here.)

$ cd $ALTIBASE_HOME/trc
$ altiProfile alti-1286503704-0.prof

$ altiProfile -stat query $ALTIBASE_HOME/trc/*.prof
```

How to use altiProfile

The QUERY_PROF_FLAG property must be set to a value larger than 0 to write information about server status and tasks. The following information is logged for the QUERY_PROF_FLAG property:

Value	Name	Description
0		No logging.
1	[STATEMENT]	Whenever a SQL statement is executed, the executed SQL statement, execution time, execution information, and information about index and disk access are output. The value for the TIMED_STATISTICS property must be set to 1 to print the proper execution time. For further information about the TIMED_STATISTICS property, please refer to the *General Reference.*
2	[BIND]	Whenever a SQL statement is executed, BIND parameter(s) is/are output.
4	[PLAN]	Whenever a SQL statement is executed, the execution plan is output

Value	Name	Description
8	[SESSION STAT]	Every 3 seconds, session information (i.e. the data in V$SESSTAT) is output.
16	[SYSTEM STAT]	Every 3 seconds, system information (i.e. the data in V$SYSSTAT) is output.
32	[MEMORY STAT]	Every 3 seconds, information about memory (i.e. the data in V$MEMSTAT) is output.

The above values can be combined to log the desired information. For example, if the property is set to 1+4+32=37, then whenever a SQL statement is executed, the execution information and execution plan for the SQL statement are output, and additionally, information about memory is output every 3 seconds.

If the QUERY_PROF_FLAG property is set to a nonzero value, the server will create and write to a profile file having a name that follows this convention:

Outputing Statistics

altiProfile can use the -stat option to build and out statistical information about executed SQL statements. This information helps you find the SQL statement to tune.

The -stat query option builds statistics on the following:

- COUNT: The number of times the query was executed.
- AVG: The amount of time (in microseconds) the query took to execute, on average.
- TOTAL: The amount of time (in microseconds) the query took to execute, in sum.
- MIN: The minimum amount of time (in microseconds) the query took to execute.
- MAX: The maximum amount of time (in microseconds) the query took to execute.
- SUCCESS: The number of times the query succeeded.
- FAIL: The number of times the query failed.
- QUERY: The SQL statement that was executed.

When the -stat session option is used, SESSION ID is added to the statistics built using the query option.

The following is an example of altiProfile building statistics on SQL statements by analyzing profiles in the $ALTIBASE_HOME/trc directory.

```
$ altiProfile -stat query $ALTIBASE_HOME/trc/*.prof

### Processing [/altibase_home/trc/alti-1423543095-0.prof]...
100% [====================]

### Writing CSV File [alti-prof-stat-1423543711.csv]...

### Writing TEXT File [alti-prof-stat-1423543711.txt]...

### Successfully done.
```

In the example above, statistics are saved in text and CSV formats. The names of the files are automatically generated as 'alti-prof-stat-#time.csv' and 'alti-prof-stat-#time.txt'.

The following is an example of a text file. Statistics are displayed in order under the column TOTAL.

```
$cat alti-prof-stat-1423543711.txt
COUNT      AVG          TOTAL         MIN          MAX      SUCCESS  FAIL   QUERY
================================================================================
==========================
     5      0.003730    0.018650     0.003035     0.004640    5      0      DROP
VIEW REVENUE
     5      0.003523    0.017616     0.003004     0.003745    5      0      CREATE
VIEW REVENUE (
...
```

The following is an example of a CSV file. The content displayed is the same as that of a text file, except that it is in CSV format. CSV format allows the user to move data between programs.

```
$ cat alti-prof-stat-1423543711.csv
COUNT,AVG,TOTAL,MIN,MAX,SUCCES,FAIL,QUERY
5,  0.003730,  0.018650,  0.003035,  0.004640,5,0,"DROP VIEW REVENUE"
5,  0.003523,  0.017616,  0.003004,  0.003745,5,0,"CREATE VIEW REVENUE (
...
```

Precaution

When the profiling function is activated, execution information is recorded for all SQL statements executed in the server, and the state of the server, such as session and system information, is recorded every 3 seconds. Therefore, the size of the profile file will increase rapidly, which could cause the disk to become full, consequently causing problems. So, care should be taken when considering whether to perform profiling.

Output

Files are output in the follwing format.

```
[STATEMENT]
..
[BIND]
..
[PLAN]
..
[SESSION STAT]
..
[SYSTEM STAT]
..
[MEMORY STAT]
..
```

Each item is output as follows.

[STATEMENT]

The following table shows the statement-related information that is logged.

Field Name	Value	Description
SQL	String	The SQL statement that was executed
User Info		
User ID	INTEGER	The user identifier
Client PID	BIGINT	The identifier of the client process
Client Type	VARCHAR(40)	The type of the connected client
Client AppInfo	VARCHAR(128)	A string containing information about the client applicaiton
Elapsed Time for this SQL statemen		
Total	BIGINT	The total query execution time
Parse	BIGINT	The time taken to parse the query
Valid	BIGINT	The time taken to validate the query
Optim	BIGINT	The time taken to optimize the query
Execu	BIGINT	The time taken to execute the query
Fetch	BIGINT	The time taken to fetch query results
Query Execute Info		
EXECUTE Result	INTEGER	0: failure 1: rebuild 2: retry 3: queue empty 4: success
Optimizer Mode	BIGINT	The optimization mode
Cost Mode	BIGINT	The optimization cost
Used Memory	BIGINT	Reserved for future use
SUCCESS SUM	BIGINT	The total number of successful executions
FAILURE SUM	BIGINT	The total number of failed executions
PROCESSED ROW	BIGINT	The number of processed records for this SQL statement
Result Set Info		
FETCH Result	INTEGER	0: failure 1: success 2: no results
Index Access Info		

Field Name	Value	Description
Memory Full Scan Count	BIGINT	The number of full scans that were performed on memory tables
Memory Index Scan Count	BIGINT	The number of index scans that were performed on memory tables
Disk Full Scan Count	BIGINT	The number of full scans that were performed on disk tables
Disk Index Scan Count	BIGINT	The number of index scans that were performed on disk tables
Disk Access Info		
READ DATA PAGE	BIGINT	The number of disk pages that were read from disk for the query
WRITE DATA PAGE	BIGINT	not used
GET DATA PAGE	BIGINT	The number of buffers that were accessed for a disk page during query execution
CREATE DATA PAGE	BIGINT	The number of disk pages that were created during query execution
READ UNDO PAGE	BIGINT	The number of disk pages in UNDO tablespace that were read from disk during query execution
WRITE UNDO PAGE	BIGINT	not used
GET UNDO PAGE	BIGINT	The number of buffers in UNDO tablespace that were accessed for a disk page during query execution
CREATE UNDO PAGE	BIGINT	The number of disk pages in UNDO tablespace that were created during query execution

[BIND]

Outputs information on variables that are bound to the SQL statement.

[PLAN]

Outputs the execution plan for the executed SQL statement. For more information on execution plans, please refer to the *Performance Tuning Guide.*

[SESSION STAT]

Outputs the data in the V$SESSTAT performance view every 3 seconds. For more information on the V$SESSTAT performance view, please refer to the chapter in the *General Reference*.

[SYSTEM STAT]

Outputs the data in the V$SYSSTAT performance view every 3 seconds. For more information on the V$SYSSTAT performance view, please refer to the chapter in the *General Reference*.

[MEMORY STAT]

Outputs the data in the V$MEMSTAT performance view every 3 seconds. For more information on the V$MEMSTAT performance view, please refer to the chapter in the *General Reference*.

altiwrap

About altiwrap

altiwrap encrypts code programs written as persistent stored modules (PSMs). This utility prevents PSM code (e.g., stored procedures and stored functions) from being exposed.

```
altiwarp {--iname input_file} [--oname output_file]
```

Syntax
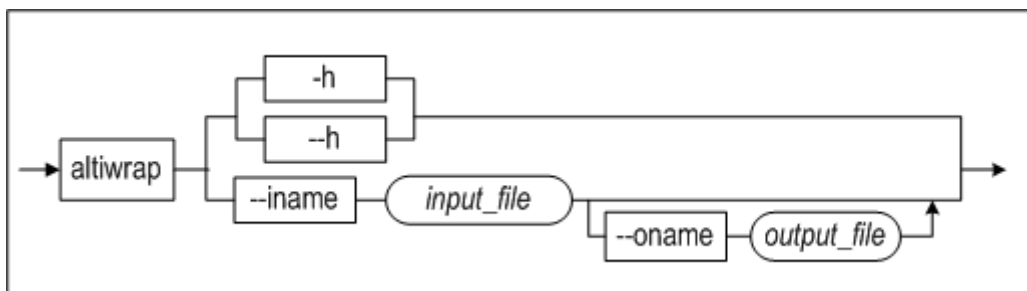
Parameters

Parameter	Description
-h/--h	Outputs help
--iname	Specifies the name of the file to encrypt. On omission of the extension, .sql is assumed.
--oname	Specifies the file name under which an encrypted code program is to be saved. On omission of the extension, it is saved as a .plb file.

Description

altiwrap encrypts the code programs of stored procedures and stored functions to prevent them from being exposed.

Altibase can encrypt the following statements.

- CREATE [OR REPLACE] PROCEDURE
- CREATE [OR REPLACE] FUNCTION
- CREATE [OR REPLACE] TYPESET
- CREATE [OR REPLACE] PACKAGE

- CREATE [OR REPLACE] PACKAGE BODY

Considerations

- Code programs cannot be modified after encryption. Changes must be made to the original code program, and then re-encrypted.

- Triggers cannot be encrypted.

- Encrypted code programs cannot be checked for syntax and semantic errors.

Example

Use altiwrap to encrypt the sample1.sql file, and then output it.

Create the sample1. sql file.

```
iSQL> create or replace procedure proc1 as
type arr1 is table of char(30) index by integer;
v1 arr1;
begin
v1[0] := 'create or replace';
v1[1] := 'typeset';
v1[2] := 'is';
v1[3] := 'success';
println( v1[0] || v1[1] || v1[2] || v1[3] || '!' );
end;
/
```

Encrypt the file

```
$ altiwrap --iname sample1.sql --oname --sample1.plb
```

Run the encrypted file in iSQL.

```
iSQL> @sample1.plb
iSQL> create or replace procedure proc1 WRAPPED
'MjQz
MTk2
AAhjcmVhdGUgb3IgcmVwbGFjZSBwcm9jZWR1cqQBAAQxIGFzCnR5cGUgYXJyMSBpcyB0YWJsQAYAAWYgY
2hhcigzMCkgaW5kZXggYnlAAQZOZWdlcjsKdjGYBQAEOwpiZWdpbgp2MVswXSA6PSAnY3JlYSu5ASdyBV
sxtAN/DHNldK0CMq4CaXO5ATO4AQNzdWNjZXNsAgZwcmludGxuKCC2C3x8YAFACaABVAegAVwGDHx8ICc
hJyApOwplbmQ7ChEAADVBRDlBRkIzMDE0MzI1Q0U0MzY1RjYxNEI2NkYwQzRDREMzMTdDQTU=
';
/
Create success.
iSQL> exec proc1;
create or replace typeset is success!
Execute success.
```

awrite

About awrite

Outputs the response time of the system call used to create the log file. The output value is used to determine the system call in the LOG_CREATE_METHOD property.

Syntax
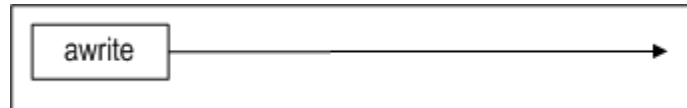
Description

Outputs the response time of write () and fallocate () system calls.

Example

awrite outputs the following:

```
$ awrite
fallocate to expand file size to 1GB
 Elapsed Time ==>               2.564 seconds
write to expand file size to 1GB
 Elapsed Time ==>               4.020 seconds
```

checkServer

About checkServer

Monitors the Altibase process and, if Altibase terminates, executes a script specified by a user.

```
checkServer [-n] {-f server-restart-script-file}
```

Syntax
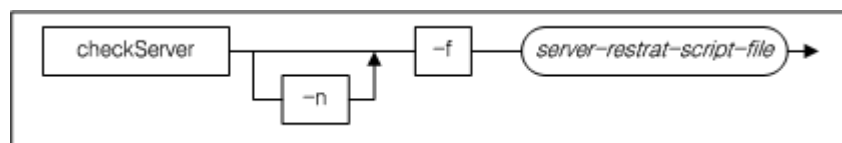
Parameters

Parameter	Description
-n	Specifies that checkServer is to be executed in the foreground. If this parameter is omitted, checkServer will be executed in the background
-f	The name of the script file to be executed when Altibase terminates

Description

checkServer periodically checks whether the Altibase process is running. If checkServer detects that Altibase has terminated, it executes the script specified by a user. It is common to set an Altibase restart script to be executed in the event of termination. Such a restart script can be written as follows:

- The Altibase starup script 'restart.sh'

```
#! /bin/sh
${ALTIBASE_HOME}/bin/server start
```

When checkServer is executed, it creates the files checkServer.pid and checkServer.log in the $ALTIBASE_HOME/trc directory. checkServer.pid is a kind of lock that prevents another instance of checkServer from being started while the current instance is running. checkServer.log is used to regularly record the status of checkServer.

If checkServer is terminated abnormally, for example by using the command kill -9, the checkServer.pid file will not be deleted from the $ALTIBASE_HOME/trc directory. As long as this file remains in that directory, it will prevent checkServer from being executed again.

To terminate checkServer normally, use the killcheckServer utility

Note

checkServer executes the specified restart script only when the Altibase server is shut down without using the server stop command. When the Altibase server is shut down normally using server stop, checkServer is also terminated, and thus does not execute the restart script. That is, checkServer only considers shutdown using the server stop command to be a normal shutdown.

Example

checkServer is executed from a shell prompt as follows:

```
$ checkServer -f restart.sh &
```

dumpbi

About dumpbi

dumpbi outputs backup information of the backupInfo file which is recorded in binary format as text format.

```
dumpbi  <backupinfo_file_name>
```

Syntax
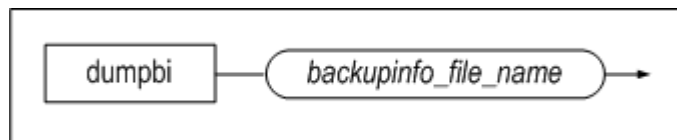
Description

Outputs contents of the backupInfo file in text format.

Example

At a shell prompt, type the following:

```
$ dumpbi backupinfo
```

Output

dumpbi outputs backup information of the backupInfo file in the following format:

[BACKUO INFO FILE HDR]

Field Name	Value (bytes)	Description
Backup info slot count	From 0(zero) to the maximum value of the unsigned int type	The number of stored backupinfo slots (=the number of files backed up until now)
Last backup LSN	FileNo, Offset	The LSN of the point in time at which the most recent backup was performed (the value necessary for determining the validity of the backupInfo file)
Database name	String	The database name

[BACKUP INFO SLOT]

Field Name	Value (bytes)	Description
Slot index	From 0(zero) to the maximum value of the unsigned int type	The slot order
Begin backup time	YYYY-MM-DD HH:MM:SS	The start time of backup
End backup time	YYYY-MM-DD HH:MM:SS	The completion time of backup

Field Name	Value (bytes)	Description
Incremental backup chunk cnt	From 0(zero) to the maximum value of the unsigned int type	The number of incremental chunks, including the pages changed from the datafile (=the number of backed up incremental chunks)
Incremental backup chunk size	From 0(zero) to the maximum value of the unsigned int type	The INCREMENTAL_BACKUP_CHUNK_SI ZE value during backup
Backup target	1: DATABASE 2: TABLESPACE	The backup target
Backup level	1: level 0 2: level 1	The backup level
Backup Type	1: full backup 2: differentail backup 3: cumulative backup	The backup type
Tablespace ID	From 0(zero) to the maximum value of the unsigned short type	The ID of the tablespace to which the backup target datafile belongs
File ID	From 0(zero) to the maximum value of the unsigned short type	The backup target datafile ID
Original file size	From 0(zero) to the maximum value of the unsigned long type	The size of the datafile when it was backed up
Backup Tag	String	The backup tag name
Backup file name	String	The path and name of the backup file

dumpct

About dumpct

Outputs information of the changeTracking file which is recorded in binary format as text format.

```
dumpct   <changeTracking_file_name>
```

Syntax
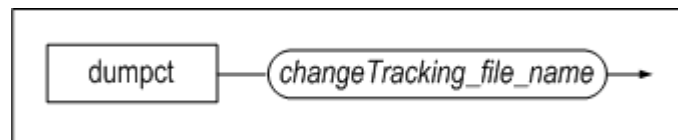
Description

Outputs contents of the changeTracking file in text format.

Example

At a shell prompt, type the following:

```
$ dumpct changeTracking
```

Output

dumpct outputs backup information of the changeTracking file in the following format:

The output is separated by [^]:

[CHANGE TRACKING FILE HDR]

Field Name	Value (bytes)	Description
Change tracking body count	From 0(zero) to the maximum value of the unsigned int type	The number of bodies of the changeTracking file. (The changeTracking file is composed of headers and bodies. The size of the body is 10Mbytes and enlarges in body units if space is insufficient)
Icremental backup chunk size	From 0(zero) to the maximum value of the unsigned int type	The value of the INCREMENTAL_BACKUP_CHUNK_SI ZE property at the time of the creation of the changeTracking file.
Last flush LSN	FileNo, Offset	The LSN at the time data changed in memory were written to files.
Database name	String	The database name

[CHANGE TRACKING FILE BODY]

Field Name	Value (bytes)	Description
Change tracking body ID	From 0(zero) to the maximum value of the unsigned int type	The body ID
Flush LSN	FileNo, Offset	The LSN at the time that the body was flushed (for file validation)

Field Name	Value (bytes)	Description
Datafile descriptor slot		
Slot ID	The LSN at the time that the body was flushed (for file validation)	The slot ID
Tracking state	0: Disables tracking 1: Enables tracking	The datafile change tracking state
Tablespace type	0: memory TBS 1: disk TBS	The tablespace type
Page size	From 0(zero) to the maximum value of the unsigned int type	The page size
Bitmap extent count	From 0(zero) to the maximum value of the unsigned short type	The number of allocated bitmap extents
Current tracking list ID	From 0(zero) to the maximum value of the unsigned short type	The ID of the bitmap extent list currently being tracked
Differential0 BmpExt list		
List	From 0(zero) to the maximum value of the unsigned int type	The block ID of the bitmap extent list
Hint	From 0(zero) to the maximum value of the unsigned int type	
Differential1 BmpExt list		
List	From 0(zero) to the maximum value of the unsigned int type	The block ID of the bitmap extent list
Hint	From 0(zero) to the maximum value of the unsigned int type	
Cumulative BmpExt list		
List	From 0(zero) to the maximum value of the unsigned int type	The block ID of the bitmap extent list
Hint	From 0(zero) to the maximum value of the unsigned int type	
Tablespace ID	From 0(zero) to the maximum value of the unsigned short type	The ID of the tablespace to which the datafile belongs
File ID	From 0(zero) to the maximum value of the unsigned short type	The datafile ID

dumpdb

About dumpdb

dumpdb outputs memory tablespace information from memory checkpoint image files, or the contents of incremental backup files of the memory tablespace in character format.

```
dumpdb {-j job_number } [-i pingpong_number] [-o] [-f file_name] [-s] [-p] [-d]
```

Syntax
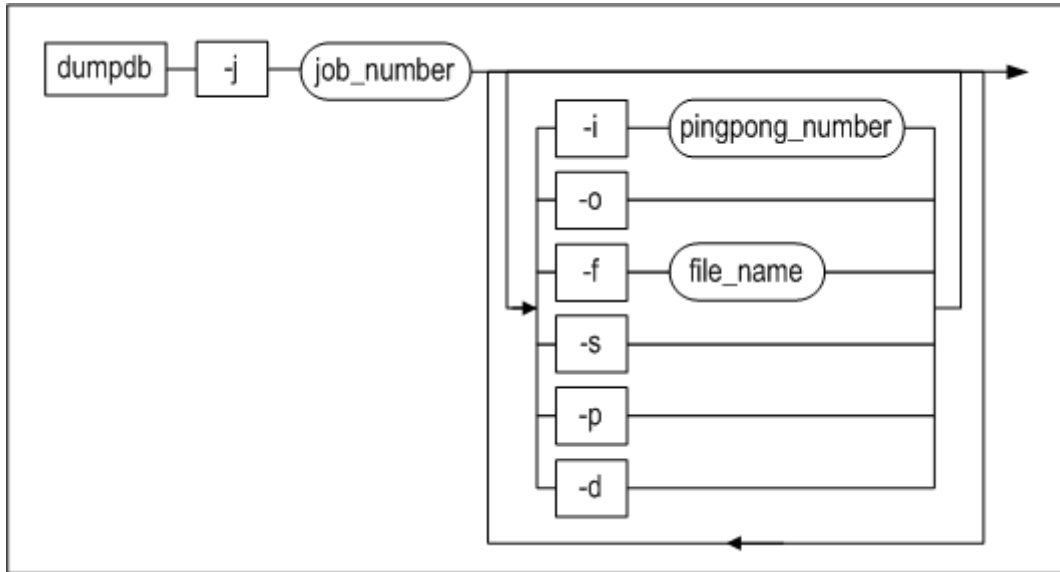
Parameters

Parameters	Description
-j job_number	Specifies which information to output. Values available for specification and suboptions available for additional specification for each value are as below: 0: META (-s -f) 1: TABLESPACE (-s -f) 2: TABLESPACE-FLI (-s -d) 3: TABLESPACE-FREE-PAGE-LIST (-s) 4: TABLE (-o -d) 5: TABLE-ALLOC-PAGE-LIST (-o) 6: PAGE (-s -p -d) 7: INCREMENTAL_BACKUP_META (-f)
-i pingpong_number	Specifies the ping pong number of the checkpoint image file. On omission, 0 is used.
-o	Specifies the ID of the object to be analyzed.
-f file_name	Specifies the name of the checkpoint image file.
-s	Specifies the ID of the tablespace to be analyzed.
-p	Specifies the ID of the page to be analyzed.

Parameters	Description
-d	Outputs detailed information.

Description

dumpdb analyzes memory checkpoint image files and outputs information of the meta header, page, etc. in text format, or outputs backup information from incremental backup files of the memory tablespace in text format.

Since this utility analyzes checkpoint image files stored on the disk, the user can view schemas created in the database, regardless of the status of the Altibase server. However, if the server is abnormally terminated after a DDL operation and this leads to the updated schema not being recorded on the disk, such information cannot be given.

Examples

At a shell prompt, type the following:

```
$ dumpdb -j 1
$ dumpdb -j 1 -s 0
$ dumpdb -j 2
$ dumpdb -j 3
$ dumpdb -j 4
$ dumpdb -j 4 -d
$ dumpdb -j 4 -o 65536
$ dumpdb -j 5 -o 65536
$ dumpdb -j 6 -s 0 -p 4
```

<Example 1> This example outputs information on the memory tablespace. By adding the -s suboption, information only regarding a certain tablespace can be output.

```
$ dumpdb -j 1
```

<Example 2> This example outputs information on the FreeListInfo(FLI) page of the memory tablespace. By adding the -s suboption, information only regarding a certain tablespace can be output; by adding the -d suboption, invalid contents of the FLI page can also be output.

```
% dumpdb -j 2
```

<Example 3> This example outputs the free pages of the memory tablespace. By adding the - s suboption, information only regarding a certain tablespace can be output.

```
% dumpdb -j 3
```

<Example 4> This example outputs information on all of the objects created in the database. By adding the -o suboption(the SelfOID in the example below), detailed information only regarding a certain object can be output; by adding the -d suboption, information on columns and indexes of the object can also be output.

```
% dumpdb -j 4
```

<Example 5> This example outputs information on all of the objects created in the database, along with information on columns and indexes.

```
% dumpdb -j 4 -d
```

<Example 6> This example outputs the schema and data of a certain table.

```
% dumpdb -j 4 -o 65568
```

<Example 7> This example outputs the list of pages that a certain table uses.

```
% dumpdb -j 5 -o 65568
```

<Example 8> This example outputs a certain page from the memory database.

```
% dumpdb -j 6 -s 0 -p 4
```

<Example 9> This example executes dumpdb on incremental backup files and outputs backup information.

```
% dumpdb -j 7 -f SYS_TBS_MEM_DATA-0-0_TAG_MONDAY.ibak
dumpdb: Release 6.3.1.0.0 - Production on Oct 31 2012 22:12:21
(c) Copyright 2001 ALTIBase Corporation.  All rights reserved.

[BEGIN CHECKPOINT IMAGE HEADER]
Binary DB Version             [ 6.2.1 ]
Redo LSN        [ 1, 5867599 ]
Create LSN      [0, 1385 ]
DataFileDescSlot ID           [ 1, 1 ]

//Incremental backup information stored in the backup file.
  [BEGIN BACKUPFILE INFORMATION]

        Begin Backup Time             [ 2012_11_06 23:18:43 ]
        End Backup Time               [ 2012_11_06 23:18:44 ]
        IBChunk Count                 [ 0 ]
        Backup Target                 [ DATABASE ]
        Backup Level                  [ LEVEL0 ]
        Backup Type                   [ FULL ]
        TableSpace ID                 [ 1 ]
        File ID                       [ 0 ]
        Backup Tag Name               [ MONDAY ]
        Backup File Name              [ /backup_dir/TAG_MONDAY/SYS_TBS_MEM_DATA-
0-0_TAG_MONDAY.ibak ]

  [END BACKUPFILE INFORMATION]

[END CHECKPOINT IMAGE HEADER]

Dump complete.
```

Output

The following table describes only the items that are output by executing the dumpdb utility on incremental backup files.

Field Name	Description
Binary DB Version	The version of the data file.
Redo LSN	The Redo LSN for media recovery. If the value of the Redo LSN of the log anchor is larger than the Redo LSN of the datafile, starting from the Redo LSN output of this item, media recovery is required.
Create LSN	The LSN at the time point of the checkpoint image creation
DataFileDescSlot ID	The DataFileDescSlot ID of the ChangeTracking bound to the memory checkpoint image.

dumpddf

About dumpddf

dumpddf outputs header information of data files or specific pages in data files. Also, if dumpddf is executed on incremental backup files, header information of the backup file and backup information are output.

```
dumpddf {-f datafile_name} {-m | -p pid}
```

Syntax
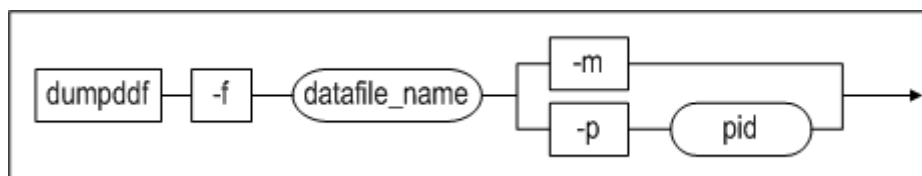
Parameters

Parameter	Description
-f	Specifies the name of the data file for which it is desired to obtain information. This option must be given. If it is omitted, dumpddf will terminate and output an error message.
-m	Outputs the data file header information.
-p	Specifies the ID of the page in the data file for which it is desired to obtain information.

Description

Outputs space and information in the data file. The menue page outputs a page from a table or socket.

Example

At a shell prompt, type the following:

```
$ dumpddf -f datafile -m
$ dumpddf -f datafile -p page_id
```

Output

The following is an example of dumpddf output:

```
[BEGIN DATABASE FILE HEADER]
Binary DB Version           [ 5.4.1 ]
Redo LSN                    [0, 734497 ]
Create LSN                  [0, 1886 ]
MustRedo LSN                [0, 0 ]
```

In the output, each field has the following meaning:

Filed Name	Description
Binary DB Version	The version of the data file
Redo LSN	The redo LSN for media recovery. If the loganchor SN is higher than the Redo LSN of the data file, it will be necessary to perform media recovery, starting from this redo LSN.
Create LSN	The LSN at the time when the specified datafile was created.
MustRedo LSN	Indicates that recovery must be performed up to this redo LSN
DataFileDescSlot ID	The ID of the DataFileDescSlot of the changeTracking file bound to the disk datafile.

The following is an example of outputting an incremental backup file by dumpddf.

```
% dumpddf -m -f system001.dbf_TAG_MONDAY.ibak
-----------------------------------------------------------------
     Altibase Client Dump Disk Database File utility.
     Release Version 6.3.1.0.0
     Copyright 2000, ALTIBASE Corporation or its subsidiaries.
     All Rights Reserved.
-----------------------------------------------------------------
[BEGIN DATABASE FILE HEADER]

Binary DB Version           [ 6.2.1 ]
Redo LSN                    [1, 5867599 ]
```

```
Create LSN                      [0, 1914 ]
MustRedo LSN                    [0, 0 ]
DataFileDescSlot ID             [ 1, 2 ]


[BEGIN BACKUPFILE INFORMATION] -->incremental backup information stored in the
backup file


        Begin Backup Time               [ 2012_11_06 23:18:44 ]
        End Backup Time                 [ 2012_11_06 23:18:46 ]
        IBChunk Count                   [ 0 ]
        Backup Target                   [ DATABASE ]
        Backup Level                    [ LEVEL0 ]
        Backup Type                     [ FULL ]
        TableSpace ID                   [ 2 ]
        File ID                         [ 0 ]
        Backup Tag Name                 [ MONDAY ]
        Backup File Name                [
/backup_dir/TAG_MONDAY/system001.dbf_TAG_MONDAY.ibak ]


    [END BACKUPFILE INFORMATION]


[END DATABASE FILE HEADER]
```

dumpla

About dumpla

ddumpla outputs the contents of loganchor files, which are saved in binary form, in the form of text.

Loganchor files contain information that is necessary in order to recover physically stored information (i.e. data files).When a database is created using the CREATE DATABASE statement, Altibase creates these files and stores them with the sequential names loganchor# (where "#" = 1, 2, or 3). Altibase stores these three files, which have the same contents, in the three respective directories specified using the LOGANCHOR_DIR property in altibase.properties. The reason that three files are maintained is to be prepared in the event that some of the files become lost or corrupt. These files contain information about all of the database's tablespaces and the data files stored in them, as well as recovery-related information. When the database is started, this information is used to load the database into memory and prepare to provide service.

```
dumpla <loganchor_file_name>
```

Syntax
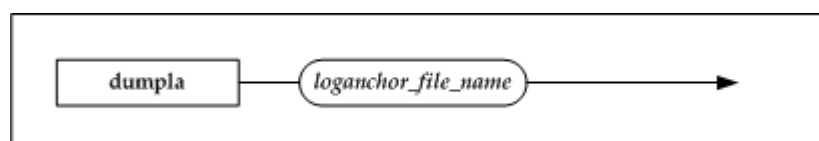
Description

Outputs the content of a loganchor file in the form of text.

Example

At a shell prompt, type the following:

```
$ dumpla loganchor0
```

Output

dumpla displays the contents of a loganchor file in the following format:

[LOGANCHOR ATTRIBUTE SIZE]

This section indicates the amount of space that is occupied by each kind of data in the loganchor file. The contents of this section are as follows:

Field Name	Value (bytes)	Description
Loganchor Static Area	From 0 (zero) to the maximum value of the unsigned int type	The size of the static information in the loganchor file. Most of this information is information that is required for recovery.
Tablespace Attribute	From 0 (zero) to the maximum value of the unsigned int type	The size of the stored tablespace attributes
Checkpoint Path Attribute	From 0 (zero) to the maximum value of the unsigned int type	The size of the stored checkpoint path attributes
Checkpoint Image Attribute	From 0 (zero) to the maximum value of the unsigned int type	The size of the stored checkpoint image attributes
Disk Datafile Attribute	From 0 (zero) to the maximum value of the unsigned int type	The size of the stored disk datafile attributes

[LOGANCHOR HEADER]

This is loganchor header information, such as the version of the database and the checkpoint Log Sequence Number (LSN). For more information about Log Sequence Numbers, please refer to "dumplf Output Items".

Field Name	Value	Description
Binary DB Version	Major.minor.patch ex) 6.2.1	The version of the database executable file with which the loganchor file was created

Field Name	Value	Description
Archivelog Mode	Archivelog |No-Archivelog	Indicates whether the database is running in archive mode
Transaction Segment Entry Count		
Begin Checkpoint LSN	FileNo, Offset	The LSN that was current when checkpointing most recently began.
End Checkpoint LSN	FileNo, Offset	The LSN that was current when checkpointing was most recently completed.
Disk Redo LSN	FileNo, Offset	The redo start point for a DRDB.
LSN for Recovery from Replication	FileNo, Offset	Recovers from replicaiton starts with this LSN.
Server Status	SERVER_SHUTDOWN |SERVER_STARTED	Logs the server state. This value is changed to SERVER_STARTED when the server is started and to SERVER_SHUTDOWN when the server is shut down normally. If this value is already set to SERVER_STARTED when the server starts, this indicates that the server was shut down abnormally, so the server will perform restart recovery.
End LSN	FileNo, Offset	The LSN of the first log that is written to when the server is started up after having shut down normally
ResetLog LSN	FileNo, Offset	The Reset LSN that was set during incomplete recovery
Last Created Logfile Num	From 0 (zero) to the maximum value of the unsigned int type	The number of the most recently created log file
Delete Logfile(s) Range	Format: [first logfile no. ~ last logfile no.] The numbers of the first and last log files that were deleted.	The range of the most recently deleted log files. When checkpointing is completed, log files that are no longer necessary are deleted. These numbers indicate the range of log files that were deleted.
Update And Flush Count	From 0 (zero) to the maximum value of the unsigned int type	The number of times that loganchor files were changed and flushed

Field Name	Value	Description
New Tablespace ID	From 0 (zero) to the maximum value of the unsigned int type	The identifier for the next new tablespace. When a tablespace is created, this value will be used as its identifier, and will then be incremented.

[TABLESPACE ATTRIBUTE]

This section provides information about the tablespace. The contents of this section are as follows:

Field Name	Value	Description
Tablespace ID	From 0 (zero) to the maximum value of the unsigned int type	The identifier of the tablespace
Tablespace Name	String Ex.) SYS_TBS_MEM_DIC	The name of the tablespace
New Database File ID	From 0 (zero) to the maximum value of the unsigned int type	The identifier that will be given to the next file to be added to the tablespace
Extent Management	FREE EXTENT BITMAP TABLESPACE	This indicates how extents are managed when a disk tablespace is created. At present, only FREE EXTENT BITMAP TABLESPACE is supported. If FREE EXTENT BITMAP TABLESPACE is enabled, bitmaps can be used to manage the free extents in a disk tablespace.
Tablespace Status	Refer to Possible Tablespace Status Values in [TABLESPACE ATTRIBUTE].	Indicates the current status of the tablespace
Tablespace Type	0 - 8 (Refer to Possible Tablespace Type Values in [TABLESPACE ATTRIBUTE])	Indicates the type of the tablespace
Checkpoint Path Count	The number of checkpoint paths	The number of checkpoint image file paths. This applies only to memory tablespaces.
Autoextend Mode	AutoExtend |Non-AutoExtend	Indicates whether the tablespace extends in size automatically. This applies only to memory tablespaces
Shared Memory Key	From 0 (zero) to the maximum value of the unsigned int type	The shared memory key for a database that resides in shared memory. This applies only to memory tablespaces

Field Name	Value	Description
Stable Checkpoint Image Num	0|1	The number corresponding to the set of checkpoint image files that is stable after checkpointing has taken place. This applies only to memory tablespaces
Init Size	From 0 (zero) to the maximum value of the unsigned int type	The initial size (MB) of the tablespace
Next Size	From 0 (zero) to the maximum value of the unsigned int type	The increment by which the tablespace automatically increases in size (MB)
Maximum Size	From 0 (zero) to the maximum value of the unsigned int type	The maximum size of the tablespace.
Split File Size	From 0 (zero) to the maximum value of the unsigned int type	When a memory tablespace is created, it consists of multiple files of this size. For example, if a tablespace 1 GB in size is to be created and the split file size is 100 MB, then 10 files will be created.

In [TABLESPACE ATTRIBUTE], Tablespace Status can have the following values:

Value	Description
OFFLINE	Currently offline
ONLINE	Currently online
INCONSISTENT	In an inconsistent state
CREATING	Being created
DROPPING	Waiting to be dropped, because the transaction that will drop the database has not been committed yet
DROP_PENDING	The transaction that will drop the database has been committed but the tablespace is still waiting to be dropped because one or more operations are still pending.
DROPPED	Deleted (dropped)
DISCARDED	Discarded
BACKUP	Being backed up
SWITCHING_TO_OFFLINE	Being brough online
SWITCHING_TO_ONLINE	Being taken offline

The possible values of Tablespace Type in [TABLESPACE ATTRIBUTE] are as follows:

Value	Description
0	MEMORY SYSTEM DICTIONARY
1	MEMORY SYSETM DATA
2	MEMORY USER DATA
3	DISK SYSTEM DATA
4	DISK USER DATA
5	DISK SYSTEM TEMP
6	DISK USER TEMP
7	DISK SYSTEM UNDO
8	VOLATILE USER DATA

[MEMORY CHECKPOINT PATH ATTRIBUTE]

This section indicates the path in which checkpoint image files are saved for a memory tablespace. The contents of this section are as follows:

Field Name	Value	Description
Tablespace ID	From 0 (zero) to the maximum value of the unsigned int type	The identifier of the tablespace
Checkpoint Path	String	The checkpoint impage file path

[MEMORY CHECKPOINT IMAGE ATTRIBUTE]

This section indicates the checkpoint image information for a memory tablespace. The contents of this section are as follows:

Field Name	Value	Description
Tablespace ID	From 0 (zero) to the maximum value of the unsigned int type	The identifier of the tablespace
File Number	From 0 (zero) to the maximum value of the unsigned int type	The file number
Create LSN	< FileNo, Offset>	The LSN that was current at the time at which the data file was created
Create On Disk (PingPong 0)	Created|None	Whether the set of checkpointing files identified by #0 has been created

Field Name	Value	Description
Create On Disk (PingPong 1)	Created|None	Whether the set of checkpointing files identified by #1 has been created
ChangeTracking DataFileDescSlot ID	From 0(zero) to the maximum value of the unsigned int type	The DataFileDescSlot ID of the changeTracking file bound to the memory checkpoint image

[DISK DATABASE FILE ATTRIBUTE]

This information indicates the path in which the data file or files for a disk tablespace are saved. The contents of this section are as follows:

Field Name	Value	Description
Tablespace ID	From 0(zero) to the maximum value of the unsigned int type	The identifier of the tablespace
Database File ID	From 0(zero) to the maximum value of the unsigned int type	The identifier of the data file
Database File Path	String	The data file path
Create LSN	< FileNo, Offset>	The LSN that was current at the time that the dat file was created
Database File Status	Refer to Database File Status Values for [DISK DATABASE FILE ATTRIBUTE]	The file state
Autoextend Mode	AutoExtend| Non-AutoExtend	Whether auto extension mode has been set
Create Mode	0|1	0: The file was reused 1: The file is a newly created file
Initialize Size	From 0 (zero) to the maximum value of the unsigned int type	The initial size (MB) of the data file
Current Size	From 0 (zero) to the maximum value of the unsigned int type	The initial size (MB) of the data file
Next Size	From 0 (zero) to the maximum value of the unsigned int type	The increment by which the data file automatically increases in size (MB)
Maximum Size	From 0 (zero) to the maximum value of the unsigned int type	The maximum size (MB) of the data file

Field Name	Value	Description
ChangeTracking DataFileDescSlot ID	From 0(zero) to the maximum value of the unsigned int type	The ID of the DataFileDescSlot of the changeTracking file bound to the disk datafile

In [DISK DATABASE FILE ATTRIBUTE], Database File Status means the following:

Value	Description
OFFLINE	Offline
ONLINE	Online
CREATING	Being created
BACKUP_BEGIN	Backup has started
BACKUP_END	Backup is being completed
DROPPING	Being dropped (deleted)
RESIZING	Being resized
DROPPED	Has been dropped

The following is an example of some of the information output by dumpla:

```
[ DISK DATABASE FILE ATTRIBUTE ]
Tablespace ID                   [ 2 ]
Database File ID                [ 0 ]
Database File Path C:\altibase_home\dbs\system001.dbf]
Create LSN                      [0, 4443 ]
Database File Status            [ ONLINE ]
Autoextend Mode                 [ Non-Autoextend ]
Create Mode                     [ 0 ]
Initialize Size                 [10 MBytes(1280 Pages)]
Current Size                    [10 MBytes(1280 Pages)]
Next Size                       [0 MBytes(0 Pages)]
Maximum Size                    [0 MBytes(0 Pages)]
```

[Change Tracking ATTRIBUTE]

This section provides information about the changeTracking file. The contents of this section are as follows:

Field Name	Value	Description
Last Flush LSN	FileNo, Offset	The LSN at the time data changed in memory were written to files
Change Tracking Manager State	String ex) CHANGE TRACKING MGR ENABLED	The page change tracking state

Field Name	Value	Description
Change Tracking File Name	String	The changeTracking file path

[Backup Info ATTRIBUTE]

This section provides information about the backupInfo file. The contents of this section are as follows:

Field Name	Value	Description
Delete Archivelog File Range	FileNo, Offset	The number of the archive log file which can be completely recovered, even after deletion
Last Backup LSN	FileNo, Offset	The LSN at the time of the most recently performed backup
Before Backup LSN	FileNo, Offset	The LSN prior to the time of the most recently performed backup
Backup Info Manager State	문자열 예) BACKUP INFO MGR INITIALIZED	The backup information file manager state
Backup Directory Path	String e.g.) /backup_dir/	The backup path
Backup Info File Name	String	The backup information file name

dumplf

ABout dumplf

When a transaction performs an operation that changes the contents of the database, such as an INSERT, DELETE or UPDATE operation, changes are made not only to the database's data buffers, but also to log files. These files are maintained for use in performing recovery if it becomes necessary. To minimize I/O, these logs are recorded in binary format. These log files are stored with the name logfile# (where "#" is the number of the log file, which continuously increments) in the directory specified in the LOG_DIR property in the altibase.properties file.

dumplf is a utility that converts and outputs the contents of these log files in text form. These logs can be used to check the types of operations that are performed on the database and determine the frequency of transactions that change the contents of the database.

```
dumplf {-f log_file_name} [-t transaction_id][-s] [-l][-S lsn [-F path] [-g]]
```

Syntax
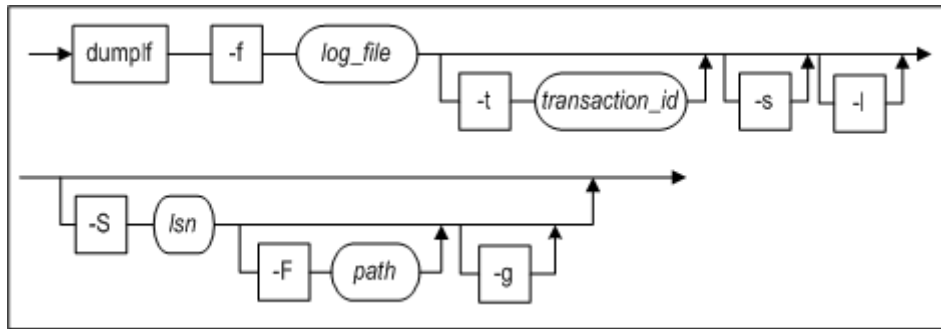
Parameters

Parameter	Description
-f	Specifies the name of the file which is to be output.
-t	Specifies the ID of the transaciton for which the logs are to be output.
-s	Specifies that only the header of the logs is to be output. If this option is omitted, both the header and the body will be output.
-l	Displays only information corresponding to log types (LT field) and sub-logtypes (OPTYPE and UTYPE fields) in the specified log file.
-S	Outputs the number of logs of INSERT, UPDATE, DELETE, COMMIT, and ROLLBACK in MMDB. If specified lsn is inserted, only the log after sn is displayed. If '0, 0' is inserted, the entire logs are displayed.
-F	Specifies the object target path, and it analyzes $ALTIBASE_HOME/logs if omitted.
-g	Outputs ID statistic of table object along with the entire statistics information. The entire statistic information is output if omitted.

Description

Converts the contents of a log file to text form and outputs it.

Example

The following is executed at a shell prompt:

```
$ dumplf -f logfile0
```

Output

The following is an example of dumplf output:

```
LSN=<0,820>, COMP:N, MAGIC:820, TID: 6400,BE: N, REP: Y, ISVP: N, ISVP_DEPTH: 0
PLSN=<0,739>, LT: SMR_LT_MEMTRANS_COMMIT, SZ: 45
```

Each field in a log file has the following meaning:다음과 같은 의미를 갖는다.

Field Name	Value	Description
LSN	Format: (FileNo, Offset) Offset range: From 0 (zero) to the maximum value of the unsigned int type	This is the log sequence number, which contains information about the physical location of the current log in a log file. The LSN consists of the identifier of the file number and an offset value.
COMP	Y|N	Indicates whether logs are compressed. Y: Compressed N: Not compressed
MAGIC	From 0 (zero) to the maximum value of the unsigned short type	This value is generated using the log file number and offset portions of the LSN to determine whether a log record is valid. When a Redo or Undo action is performed, even if a log record having garbage data is in a log file, it is possible to determine whether the log record is valid.
TID	From 0 (zero) to the maximum value of the unsigned int type	The identifier of the transaction
BE	Y|N	Y: Indicates that the Sender must check whether to send this log to the Receiver. N: This log is not used by the replication Sender.
REP	Y|N	Y: Indicates that the Sender must check whether to send this log to the Receiver. N: This log is not used by the replication Sender.
ISVP	Y|N	Y: Indicates that this log is an Implicit Savepoint Log, that is, the first log that is recorded after the start of execution of a statement. If an error occurs while the statement is executing, the transaction is partially rolled back; that is, it is rolled back only as far as this log.
ISVP_DEPTH	0 - 255	Implicit Savepoint Depth, that is, the nesting depth when a statement is nested within one or more other statements

Field Name	Value	Description
PLSN	Format: (FileNo, Offset) Offset range: From 0 (zero) to the maximum value of the unsigned int type	This value is used to connect all of the logs recorded by the same transaction in a chain.
LT	String	Indicates the Log Type (LT).
SZ	From 0 (zero) to the maximum value of the unsigned int type	Indicates the size of the log, in bytes
RdSz	From 0 (zero) to the maximum value of the unsigned int type	indicates the size of the redo log record, in bytes
DMIOff	From 0 (zero) to the maximum value of the unsigned int type	Indicates the location of the logical log that is used to undo a transaction, or is used for replication
TableOID	From 0 (zero) to the maximum value of the unsigned int type	The object identifier of the table
OID	From 0 (zero) to the maximum value of the unsigned int type	The object identifier of all objects other than tables. This includes record objects
ContType	0, 1	An itnernal vlaue that is used for replicaiton
OPTYPE	LogTypeName<LogTypeNumber>	The operation type of a Nested Top Action (NTA) log
AFTER	SZ: <size>, Value: <value>	The after name image of the log record
BEFORE	SZ: <size>, Value: <value>	The before image of the log record
UTYPE	LogTypeName<LogTypeNumber>	The operation type of an UPDATE log
UPOS	Format: (SPACEID:<SpaceID>, PID: <PageID>, OFFSET:<Offset> => OID: <OID>)	The address of the object that was updated. Additionally contains information about what happened during an update operation.
SPACEID	From 0 (zero) to the maximum value of the unsigned short type	The identifier of the tablespace containing the object that was updated
PID	From 0 (zero) to the maximum value of the unsigned int type	The identifier of the page containing the object that was updated

Field Name	Value	Description
Offset	From 0 (zero) to the maximum value of the unsigned short type	The offset from the beginning of the page containing the object that was updated
FLISlot PrevPID NextPID	Format: (<BeforePID> => <AfterPID>)	An internal value used for managing MMDB tablespaces
ESLSN	Format: (FileNo, Offset) Offset range: From 0 (zero) to the maximum value of the unsigned int type	This is the LSN from which recovery would be performed, if necessary
Lob Locator	From 0 (zero) to the maximum value of the unsigned long type	An interally used value related to the use of replication with the LOB type

The possible values of LT (Log Type) in the dumplf output are as follows:

Value	Description
SMR_LT_DUMMY	Dummy Log
SMR_LT_CHKPT_BEGIN	Checkpoint Being Log
SMR_LT_DIRTY_PAGE	Dirty Page Log
SMR_LT_CHKPT_END	Checkpoint End Log
SMR_LT_MEMTRANS_COMMIT	Memory Transaction Commit Log
SMR_LT_MEMTRANS_ABORT	Memory Transaction Abort Log
SMR_LT_DSKTRANS_COMMIT	Disk Transaction Commit Log
SMR_LT_DSKTRANS_ABORT	Disk Transaction Abort Log
SMR_LT_SAVEPOINT_SET	Savepoint Set Log
SMR_LT_SAVEPOINT_ABORT	Savepoint Abort Begin Log
SMR_LT_XA_PREPARE	XA Prepare Log
SMR_LT_TRANS_PREABORT	Abort Begin Log
SMR_LT_DDL	DDL (Data Definition Language) Log
SMR_LT_XA_SEGS	XA Prepare Transaction Segment Information
SMR_LT_LOB_FOR_REPL	LOB Log for Replication
SMR_LT_UPDATE	MMDB(Main Memory Database) Update Log
SMR_LT_NTA	MMDB NTA(Nested Top Action) Log
SMR_LT_COMPENSATION	Compensation Log

Value	Description
SMR_LT_DUMMY_COMPENSATION	Dummy Compensation Log
SMR_LT_FILE_BEGIN	File Begin Log
SMR_LT_FILE_TBS_UPDATE	Tablespace Update Log
SMR_LT_FILE_END	File End Log
SMR_DLT_READONLY	DRDB(Disk Resident Database) Redo Only Log
SMR_DLT_UNDOABLE	DRDB Undo Log
SMR_DLT_NTA	DRDB NTA Log
SMR_DLT_COMPENSATION	DRDB Compensation Log
SMR_DLT_REF_NTA	DRDB Reference NTA Log
SMR_LT_TABLE_META	Table Meta Log for Replication

Possible LogTypeName Values for OPTYPE and UTYPE

Value	Description
SMR_OP_SMM_PERS_LIST_ALLOC SMR_OP_SMC_FIXED_SLOT_ALLOC SMR_OP_SMC_VAR_SLOT_ALLOC SMR_OP_SMC_FIXED_SLOT_FREE SMR_OP_SMC_VAR_SLOT_FREE	Logs related to pages and slots in an MMDB
SMR_OP_CREATE_TABLE SMR_OP_CREATE_INDEX SMR_OP_DROP_INDEX SMR_OP_ALTER_TABLE SMR_OP_SMM_CREATE_TBS SMR_OP_INSTANT_AGING_AT_ALTER_TABLE SMR_OP_SMC_TABLEHEADER_ALLOC	Logs related to the execution of DDL statements in an MMDB
SMR_MEM_LOB_CURSOR_OPEN SMR_DISK_LOB_CURSOR_OPEN SMR_LOB_CURSOR_CLOSE SMR_PREPARE4WRITE SMR_FINISH2WRITE	Logs related to controlling LOB values in an MMDB

Value	Description
SDR_OP_SDP_CREATE_TABLE_SEGMENT SDR_OP_SDP_CREATE_LOB_SEGMENT SDR_OP_SDP_CREATE_INDEX_SEGMENT SDR_OP_SDP_ADD_LOB_PAGE_TO_AGINGLIST SDR_OP_SDC_ALLOC_UNDO_PAGE SDR_OP_SDPTB_ALLOCATE_AN_EXTENT_FROM_TBS SDR_OP_SDPTB_ALLOCATE_AN_EXTDIR_FROM_LIST SDR_OP_SDPTB_RESIZE_GG SDR_OP_SDPST_ALLOC_PAGE SDR_OP_SDPSF_ALLOC_PAGE SCT_UPDATE_MRDB_CREATE_TBS SCT_UPDATE_MRDB_CREATE_CIMAGE_FILE SCT_UPDATE_MRDB_DROP_TBS SCT_UPDATE_MRDB_ALTER_AUTOEXTEND SCT_UPDATE_MRDB_ALTER_TBS_ONLINE SCT_UPDATE_MRDB_ALTER_TBS_OFFLINE SCT_UPDATE_DRDB_CREATE_TBS SCT_UPDATE_DRDB_DROP_TBS SCT_UPDATE_DRDB_ALTER_TBS_ONLINE SCT_UPDATE_DRDB_ALTER_TBS_OFFLINE SCT_UPDATE_DRDB_CREATE_DBF SCT_UPDATE_DRDB_DROP_DBF SCT_UPDATE_DRDB_EXTEND_DBF SCT_UPDATE_DRDB_SHRINK_DBF SCT_UPDATE_DRDB_AUTOEXTEND_DBF SCT_UPDATE_DRDB_ALTER_DBF_ONLINE SCT_UPDATE_DRDB_ALTER_DBF_OFFLINE SCT_UPDATE_VRDB_CREATE_TBS SCT_UPDATE_VRDB_DROP_TBS SCT_UPDATE_VRDB_ALTER_AUTOEXTEND SCT_UPDATE_COMMON_ALTER_ATTR_FLAG	Logs related to tablespaces and segments
SDR_OP_SDPST_UPDATE_WMINFO_4DPATH SDR_OP_SDPST_UPDATE_MFNL_4DPATH SDR_OP_SDPST_UPDATE_BMP_4DPATH SDR_OP_SDPSF_ADD_PIDLIST_PVTFREEPIDLIST_4DPATH SDR_OP_SDPSF_MERGE_SEG_4DPATH SDR_OP_SDPSF_UPDATE_HWMINFO_4DPATH SDR_OP_SDP_DPATH_ADD_SEGINFOSET	Logs related to page management for Direct Page Insert in a DRDB
SDR_OP_SDN_INSERT_KEY_WITH_NTA SDR_OP_SDN_DELETE_KEY_WITH_NTA	NTA Logs for DRDB B-tree Indexes
SDR_OP_STNDR_INSERT_KEY_WITH_NTA SDR_OP_STNDR_DELETE_KEY_WITH_NTA	NTA Logs for DRDB R-tree Indexes

Value	Description
SDR_SDP_1BYTE SDR_SDP_2BYTE SDR_SDP_4BYTE SDR_SDP_8BYTE SDR_SDP_BINARY	Physical DRDB logs
SDR_SDP_PAGE_CONSISTENT SDR_SDP_INIT_PHYSICAL_PAGE SDR_SDP_INIT_LOGICAL_HDR SDR_SDP_INIT_SLOT_DIRECTORY SDR_SDP_FREE_SLOT SDR_SDP_FREE_SLOT_FOR_SID SDR_SDP_RESTORE_FREESPACE_CREDIT SDR_SDP_RESET_PAGE SDR_SDP_WRITE_PAGEIMG SDR_SDP_WRITE_DPATH_INS_PAGE	Logs related to page and slot management in a DRDB
SDR_SDPST_INIT_SEGHDR SDR_SDPST_INIT_BMP SDR_SDPST_INIT_LFBMP SDR_SDPST_INIT_EXTDIR SDR_SDPST_ADD_RANGESLOT SDR_SDPST_ADD_SLOTS SDR_SDPST_ADD_EXTDESC SDR_SDPST_ADD_EXT_TO_SEGHDR SDR_SDPST_UPDATE_WM SDR_SDPST_UPDATE_MFNL SDR_SDPST_UPDATE_PBS SDR_SDPST_UPDATE_LFBMP_4DPATH SDR_SDPSC_INIT_SEGHDR SDR_SDPSC_INIT_EXTDIR SDR_SDPSC_ADD_EXTDESC_TO_EXTDIR SDR_SDPTB_INIT_LGHDR_PAGE SDR_SDPTB_ALLOC_IN_LG SDR_SDPTB_FREE_IN_LG	The log related to segment and tablespace management for DRDB
SDR_SDC_INSERT_ROW_PIECE SDR_SDC_INSERT_ROW_PIECE_FOR_UPDATE SDR_SDC_INSERT_ROW_PIECE_FOR_DELETEUNDO SDR_SDC_UPDATE_ROW_PIECE SDR_SDC_OVERWRITE_ROW_PIECE SDR_SDC_CHANGE_ROW_PIECE_LINK SDR_SDC_DELETE_FIRST_COLUMN_PIECE SDR_SDC_ADD_FIRST_COLUMN_PIECE SDR_SDC_DELETE_ROW_PIECE_FOR_UPDATE SDR_SDC_DELETE_ROW_PIECE SDR_SDC_LOCK_ROW	Logs related to the management of rows in tables in a DRDB

Value	Description
SDR_SDC_UPDATE_LOBDESC SDR_SDC_UPDATE_LOBDESC_KEY SDR_SDC_LOB_WRITE_PIECE SDR_SDC_LOB_WRITE_PIECE4DML SDR_SDC_INIT_LOBPAGE SDR_SDC_LOB_PAGE_TO_AGING_LIST	Logs related to the use of the LOB type in a DRDB
SDR_SDC_PK_LOG	Logs related to the use of primary keys for replication in a DRDB
SDR_SDC_INIT_CTL SDR_SDC_EXTEND_CTL SDR_SDC_BIND_CTS SDR_SDC_UNBIND_CTS SDR_SDC_BIND_ROW SDR_SDC_UNBIND_ROW SDR_SDC_ROW_TIMESTAMPING SDR_SDC_DATA_SELFAGING	Logs related to MVCC for records in a DRDB
SDR_SDC_BIND_TSS SDR_SDC_UNBIND_TSS SDR_SDC_SET_INITSCN_TO_TSS SDR_SDC_INIT_TSS_PAGE SDR_SDC_INIT_UNDO_PAGE SDR_SDC_INSERT_UNDO_REC	Logs related to Transaction Status Slots (TSS) and undo records in a DRDB
SDR_SDN_INSERT_INDEX_KEY SDR_SDN_FREE_INDEX_KEY SDR_SDN_INSERT_UNIQUE_KEY SDR_SDN_INSERT_DUP_KEY SDR_SDN_DELETE_KEY_WITH_NTA SDR_SDN_FREE_KEYS SDR_SDN_COMPACT_INDEX_PAGE	Logs related to B-tree indexes in a DRDB
SDR_SDN_MAKE_CHAINED_KEYS SDR_SDN_MAKE_UNCHAINED_KEYS SDR_SDN_KEY_STAMPING SDR_SDN_INIT_CTL SDR_SDN_EXTEND_CTL SDR_SDN_FREE_CTS	Logs related to MVCC for B-tree index keys in a DRDB
SDR_STNDR_MAKE_CHAINED_KEYS SDR_STNDR_MAKE_UNCHAINED_KEYS SDR_STNDR_KEY_STAMPING	Logs related to MVCC for R-tree index keys in a DRDB
SMR_PHYSICAL	Physical logs in an MMDB

Value	Description
SMR_SMM_MEMBASE_SET_SYSTEM_SCN SMR_SMM_MEMBASE_ALLOC_PERS_LIST SMR_SMM_MEMBASE_ALLOC_EXPAND_CHUNK SMR_SMM_PERS_UPDATE_LINK SMR_SMM_PERS_UPDATE_NEXT_FREE_PAGE_LINK SMR_SMM_MEMBASE_INFO	Logs related to base information in an MMDB
SMR_SMC_TABLEHEADER_INIT SMR_SMC_TABLEHEADER_UPDATE_INDEX SMR_SMC_TABLEHEADER_UPDATE_COLUMNS SMR_SMC_TABLEHEADER_UPDATE_INFO SMR_SMC_TABLEHEADER_SET_NULLROW SMR_SMC_TABLEHEADER_UPDATE_ALL SMR_SMC_TABLEHEADER_UPDATE_ALLOCINFO SMR_SMC_TABLEHEADER_UPDATE_FLAG SMR_SMC_TABLEHEADER_SET_SEQUENCE SMR_SMC_TABLEHEADER_UPDATE_TABLE_COLUMN_COUNT SMR_SMC_TABLEHEADER_UPDATE_TABLE_SEGMENT SMR_SMC_TABLEHEADER_UPDATE_FLAG_FOR_MEDIA_RECV SMR_SMC_TABLEHEADER_SET_SEGSTOATTR SMR_SMC_TABLEHEADER_SET_INSERTLIMIT SMR_SMC_INDEX_SET_FLAG SMR_SMC_INDEX_SET_SEGATTR SMR_SMC_INDEX_SET_SEGSTOATTR SMR_SMC_INDEX_SET_DROP_FLAG	Logs related to table headers and index headers in an MMDB
SMR_SMC_PERS_INIT_FIXED_PAGE SMR_SMC_PERS_INIT_FIXED_ROW SMR_SMC_PERS_UPDATE_FIXED_ROW SMR_SMC_PERS_UPDATE_FIXED_ROW_NEXT_FREE SMR_SMC_PERS_UPDATE_FIXED_ROW_NEXT_VERSION SMR_SMC_PERS_SET_FIX_ROW_DROP_FLAG SMR_SMC_PERS_SET_FIX_ROW_DELETE_BIT SMR_SMC_PERS_INIT_VAR_PAGE SMR_SMC_PERS_UPDATE_VAR_ROW_HEAD SMR_SMC_PERS_UPDATE_VAR_ROW SMR_SMC_PERS_SET_VAR_ROW_FLAG SMR_SMC_PERS_SET_VAR_ROW_NXT_OID SMR_SMC_PERS_WRITE_LOB_PIECE SMR_SMC_PERS_INSERT_ROW SMR_SMC_PERS_UPDATE_INPLACE_ROW SMR_SMC_PERS_UPDATE_VERSION_ROW SMR_SMC_PERS_DELETE_VERSION_ROW	Logs related to rows in tables in an MMDB

Please refer to the *Atibase Administrator's Manual* for more information about MVCC.
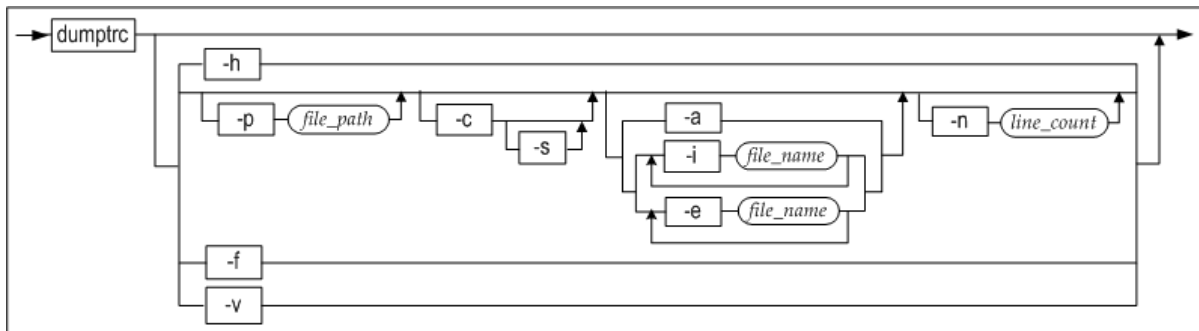
dumptrc

About dumptrc

dumptrc outputs a trace log file recorded in $ALTIBASE_HOME/trc directory by converting into a format that a user can identify when the ALTIBASE HDB server is abnormally shut down.

```
dumptrc [-h |[-p file_path][-c [-s]]
[-a|-i file_name [-i file_name]..|-e file_name [-e file_name]..] [-n file_count]
|-f |-v]
```

Syntax

Parameters

Parameter	Description
-h	This outputs help. If the parameter is omitted or used as a duplicate with other parameter, the help would be preceding.
-p	This parameter specify the path brining a trace log file. Basically, if the path is not specified, a log file in $ALTIBASE_HOME/trc is retrieved.
-c	This parameter converts ALTIBASE process call stack recorded in altibase_error.log into a function name which a user can identify and return. Use -s option if the user does not wish to convert the call stack address into a function name. Only the call stack is recorded if the parameter is used with -a, -i, -e, -n.
-s	This parameter outputs only the call stack recorded in a trace log and does not modify a function name.
-a	This parameter sorts and outputs all the trace log files.
-i	This parameter outputs specific trace log files. Multiple log files can be output. However, this parameter cannot be used with -e parameter.
-e	This parameter outputs all files except a specified trace log file. Multiple log files cannot be repeatedly removed, and this parameter cannot be used with -i parameter.
-n	This parameter specifies the number of logs which will be output at once. 1 to 127 logs can be output and 10 logs are output unless otherwise specified.

Parameter	Description
-f	This parameter outputs a log message which is added whenever a trace log file is recorded.
-v	This parameter outputs dumptc version

Description

The information of Altibase internal modules executed during Altibase termination is recorded in the process call stack. The module informaiton can be retrieved with dumptrc and retrievable trace log files can use the following files that are recorded into $ALTIBASE_HOME/trc.

- ERROR : altibase_error.log

- SERVER : altibase_boot.log

- SM : altibase_sm.log

- RP : altibase_rp.log

- QP : altibase_qp.log

- DK : altibase_dk.log

- DR : altibase_dr.log

- XA : altibase_xa.log

- MM : altibase_mm.log

- RP_CONFLICT : altibase_rp_conflict.log

- DUMP : altibase_dump.log

- TRC : altibase_trace.log

- SNMP : altibase_snmp.log

- CM : altibase_cm.log

- MISC : altibase_misc.log

- SD : altibase_sd.log

If outputting a trace log file located from a different path other than $ALTIBASE_HOME/trc, the path can be modified by using -p parameter.

If experiencing abnormal termination of Altibase server, convert the process call stack into dumptrc and send it to Altibase technical service team so that the problem can be quickly solved.

Precaution

The version of Altibase executable file and that of the dumptrc should be identical to confirm the call stack information in order to normally operate the dumptrc.

Examples

<Example 1> altibase_error.log and altibase_boot.log are bound to output

```
$ dumptrc -i server -i error
[2015/10/21 17:29:42 55C][PID:32702][Thread-2]
  ==> Initialize Disaster Recovery Manager
[2015/10/21 17:29:42 55D][PID:32702][Thread-2]
```

```
        ... [SUCCESS]
[2015/10/21 17:29:42 55E][PID:32702][Thread-2]
   ==> Initialize MMX Service
[2015/10/21 17:29:42 55F][PID:32702][Thread-2]
    ... [SUCCESS]
[2015/10/21 17:29:42 560][PID:32702][Thread-2]
   ==> Initialize Audit Service
[2015/10/21 17:29:42 561][PID:32702][Thread-2]
    ... [SUCCESS]
[2015/10/21 17:29:42 562][PID:32702][Thread-2]
   ==> Initialize Job Manager
[2015/10/21 17:29:42 563][PID:32702][Thread-2]
    ... [SUCCESS]
[2015/10/21 17:29:42 564][PID:32702][Thread-2]
--- STARTUP Process SUCCESS ---
[2015/10/21 17:30:51 57B] Dump of Stack
SIGNAL INFORMATION ============================
Signal 6(SIGABRT) caught.
     Sent by process : 2331
     Sent by user    : 1000
BEGIN-DUMP ====================================
===== SERVER =====
ALTIBASE hdb
     Product version  : 6.7.1.0.0
     CPU              : X86
     Operating System : x86_64-unknown-linux-gnu
     Process ID       : 32698
     Thread No        : 0
END-DUMP ======================================
BEGIN-STACK [CRASH] ===========================
Caller[0] 00000000011E0EBF
Caller[1] 0000000000426CFE
Caller[2] 00007FD306FD4CB0
Caller[3] 00007FD30609F763
Caller[4] 000000000042F66F
Caller[5] 00000000004213CE
Caller[6] 000000000041D1E0
Caller[7] 00007FD305FD376D
Caller[8] 00000000004206BD
END-STACK =====================================
10 logs printed.
```

<Example 2> Output except error.log.

```
$ dumptrc -e error
[2015/10/21 17:29:48 571][PID:32702][Thread-2]
[EXEC_DDL_BEGIN : DROP TABLE T1]
[2015/10/21 17:29:48 572][PID:32702][Thread-2]
[EXEC_DDL_END : SUCCESS]
[2015/10/21 17:29:49 573][PID:32702][Thread-2]
[EXEC_DDL_BEGIN : CREATE TABLE T1 ( I1 INTEGER )]
[2015/10/21 17:29:49 574][PID:32702][Thread-2]
[EXEC_DDL_END : SUCCESS]
[2015/10/21 17:29:49 575][PID:32702][Thread-2]
[EXEC_DDL_BEGIN : DROP TABLE T1]
```

```
[2015/10/21 17:29:49 576][PID:32702][Thread-2]
[EXEC_DDL_END : SUCCESS]
[2015/10/21 17:29:53 577][PID:32702][Thread-2]
[EXEC_DDL_BEGIN : CREATE TABLE T1 ( I1 INTEGER )]
[2015/10/21 17:29:53 578][PID:32702][Thread-2]
[EXEC_DDL_END : SUCCESS]
[2015/10/21 17:29:53 579][PID:32702][Thread-2]
[EXEC_DDL_BEGIN : DROP TABLE T1]
[2015/10/21 17:29:53 57A][PID:32702][Thread-2]
[EXEC_DDL_END : SUCCESS]
10 logs printed.
```

<Example 3> Output a call stack, altibase_boot.log, and altiabse_sm.log together, but output the top 20 logs.

```
$ dumptrc -c -i error -i server -i sm -n 20
================================================
= Callstack Information 0
================================================
[2015/10/21 17:29:42 551][PID:32702][Thread-2]
  ==> Initialize Security Module
[2015/10/21 17:29:42 552][PID:32702][Thread-2]
  ... [SUCCESS]
...
[2015/10/21 17:29:42 564][PID:32702][Thread-2]
--- STARTUP Process SUCCESS ---
[2015/10/21 17:30:51 57B] Dump of Stack
SIGNAL INFORMATION ==============================
Signal 6(SIGABRT) caught.
    Sent by process : 2331
    Sent by user    : 1000
BEGIN-DUMP ======================================
===== SERVER =====
ALTIBASE hdb
    Product version  : 6.7.1.0.0
    CPU              : X86
    Operating System : x86_64-unknown-linux-gnu
    Process ID       : 32698
    Thread No        : 0
END-DUMP ========================================
BEGIN-STACK [CRASH] =============================
Caller[0] 00000000011E0EBF  => iduStack::dumpStack(iduSignalDef const*, siginfo*,
ucontext*)
Caller[1] 0000000000426CFE  => mmmSignalHandler
Caller[2] 00007FD306FD4CB0  => not found
Caller[3] 00007FD30609F763  => not found
Caller[4] 000000000042F66F  => mmtSessionManager::run()
Caller[5] 00000000004213CE  => mmi::serverStart(int, int)
Caller[6] 000000000041D1E0  => main
Caller[7] 00007FD305FD376D  => not found
Caller[8] 000000000004206BD  => _start
END-STACK =======================================
20 logs printed.
```

<Example 4> Output a call stack from trc log of a different directory

```
$ dumptrc -p /home/djin/work/altidev4/trunk/altibase_home/trc -c -n 20


Path : /home/djin/work/altidev4/trunk/altibase_home/trc
===============================================
= Callstack Information 0
===============================================
[2015/10/21 17:29:45 568][PID:32702][Thread-2]
[EXEC_DDL_END : SUCCESS]
[2015/10/21 17:29:45 569][PID:32702][Thread-2]
[EXEC_DDL_BEGIN : DROP TABLE T1]
...
[2015/10/21 17:30:51 57B] Dump of Stack
SIGNAL INFORMATION ============================
Signal 6(SIGABRT) caught.
    Sent by process : 2331
    Sent by user    : 1000
BEGIN-DUMP ====================================
===== SERVER =====
ALTIBASE hdb
    Product version  : 6.7.1.0.0
    CPU              : X86
    Operating System : x86_64-unknown-linux-gnu
    Process ID       : 32698
    Thread No        : 0
END-DUMP ======================================
BEGIN-STACK [CRASH] ===========================
Caller[0] 00000000011E0EBF  => iduStack::dumpStack(iduSignalDef const*, siginfo*,
ucontext*)
Caller[1] 0000000000426CFE  => mmmSignalHandler
Caller[2] 00007FD306FD4CB0  => not found
...
Caller[8] 00000000004206BD  => _start
END-STACK =====================================
20 logs printed.
```

killCheckServer

About killCheckServer

killCheckServer terminates the checkServer utility if it is currently running.

```
killCheckServer
```

Syntax

```
killCheckServer
```

Description

killCheckServer terminates the checkServer utility if it is currently running.

If the server stop or server kill command was issued to stop the Altibase server, the server script terminates checkServer process by executing this utility before terminating the Altibase instance. In that case, the following results of killCheckServer execution will be recorded in the killCheckServer.log file under the $ALTIBASE_HOME/trc directory:

- When checkServer is running:

```
checkServer killed.
```

- When checkServer is not running:

```
ERROR CODE : -27
```

If the user executes the killCheckServer command directly, then the result will not be recorded in the file.

Example

At a shell prompt, enter the following:
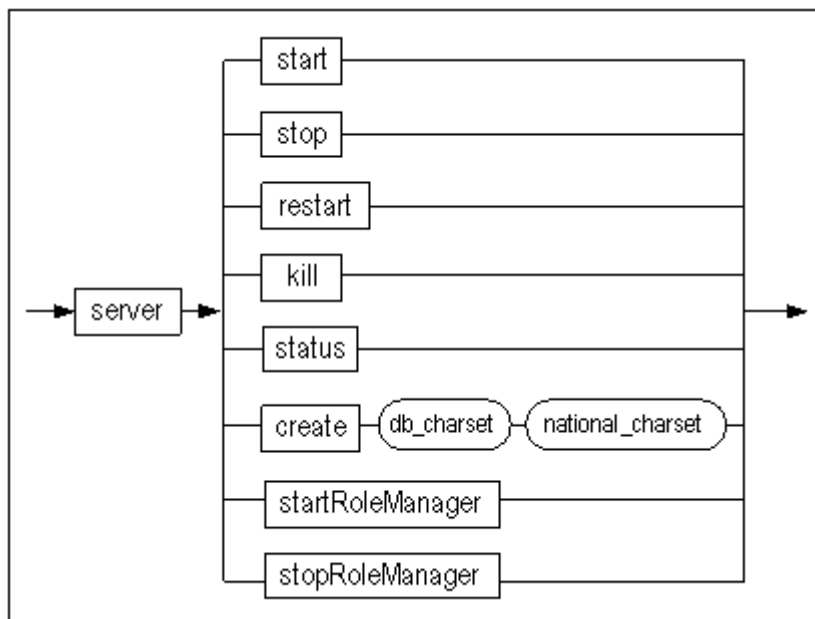
```
$ killCheckServer
```

server

About server

server is a shell script that is used to create, start up, shut down and check the status of Altibase.

```
server { start | stop | restart | kill | status | create db_charset
national_charset | startRoleManager | stopRoleManager }
```

Syntax

Parameters

Parameter	Description
start	Starts up the Altibase process
stop	Shuts down the Altibase process
restart	Restarts the Altibase process
kill	Forcibly termincates the Altibase process
status	Displays the status of the Altibase process
create	Creates a database that is 10MB in size, runs in noarchivelog mode, and uses the specified character sets
startRoleManager	Starts the Altibase process as the role manager for Disaster Recovery
stopRoleManager	Terminates the role manager

Description

iSQL is used to execute SQL statements for creating, starting up and shutting down Altibase. These frequently used commands have been combined and provided in the form of the server shell script for the convenience of DBAs.

The server script includes the following functionalities:

- Starting up the Altibase process
- Shutting down the Altibase process
- Restarting the Altibase process
- Forcibly terminating the Altibase process
- Displaying the result of querying "SELECT * from TAB;"
- Creating an Altibase

- Starting the Altibase process as the role manager

For more information about using SQL to manage Altibase databases, please refer to the *SQL Reference*.

Examples

The server shell command is used as follows:

```
$ server start
$ server restart
$ server stop
$ server status
$ server kill
$ server create ksc5601 utf16
$ server startRoleManager
$ server stopRoleManager
```

References

Please refer to the *Administrator's Manual* and *SQL Reference*.