ALTIBASE® HDB™ Tools & Utilities

Utilities Manual

Release 6.5.1

May 28, 2015

ALTIBASE

ALTIBASE HDB Tools & Utilities Utilities Manual
Release 6.5.1

Altibase Corporation
10F, Daerung PostTower II, 182-13,
Guro-dong Guro-gu Seoul, 152-847, South Korea
Telephone: +82-2-2082-1000 Fax: 82-2-2082-1099
Homepage: http://www.altibase.com

Contents

Preface

About This Manual

This manual describes how to use ALTIBASE HDB utilities.

Audience

This manual has been prepared for the following ALTIBASE HDB users:

- Database administrators
- Application developers
- Programmers

It is recommended that 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 databases and an understanding of database concepts
- Computer programming experience

Software Environment

This manual has been prepared assuming that ALTIBASE HDB 6 is used as the database server.

Organization

This manual is organized as follows:

Documentation Conventions

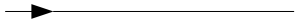
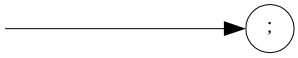
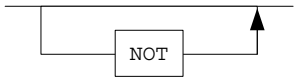
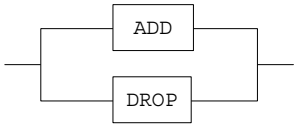
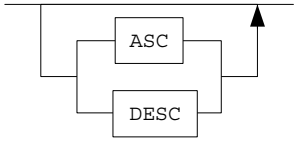
This section describes the conventions used in this manual. Understanding these conventions will make it easier to find information in this and other manuals.

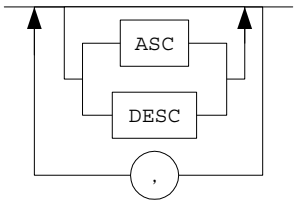
There are two sets of conventions:

- Syntax diagram conventions
- Sample code conventions

Syntax Diagram Conventions

In this manual, the syntax of commands is described using diagrams composed of the following elements:

Element	Description
	Indicates the start of a command. If a syntactic element starts with an arrow, it is not a complete command.
	Indicates that the command continues to the next line. If a syntactic element ends with this symbol, it is not a complete command.
	Indicates that the command continues from the previous line. If a syntactic element starts with this symbol, it is not a complete command.
	Indicates the end of a statement.
	Indicates a mandatory element.
	Indicates an optional element.
	Indicates a mandatory element comprised of options. One, and only one, option must be specified.
	Indicates an optional element comprised of options.

Element	Description
	Indicates an optional element in which multiple elements may be specified. A comma 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.

Convention	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 ]`

.
.
.
	Indicates that the previous argument is repeated, or that sample code has been omitted.	`iSQL> select e_lastname from employees;`
		`E_LASTNAME`
		`-----------------------`
		`Moon`
		`Davenport`
		`Kobain`
		`.`
		`.`
		`.`
		`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 Letters	Indicate program elements set by the user, such as table names, column names, file names, etc.	`SELECT e_lastname FROM employees;`

Convention	Meaning	Example
Upper Case Letters	Keywords and all elements provided by the system appear in upper case.	`DESC SYSTEM_.SYS_INDICES_;`

Related Reading

For additional technical information, please refer to the following manuals:

- ALTIBASE HDB Installation Guide
- ALTIBASE HDB Administrator's Manual
- ALTIBASE HDB Replication Manual
- ALTIBASE HDB Precompiler User's Manual
- ALTIBASE HDB ODBC Reference
- ALTIBASE HDB Application Program Interface User's Manual
- ALTIBASE HDB iSQL User's Manual

Online Manuals

Online versions of our manuals (PDF or HTML) are available from Altibase's Customer Support site (http://support.altibase.com/).

Altibase Welcomes Your Comments

Please feel free to send us your comments and suggestions regarding this manual. Your comments and suggestions are important to us, and may be used to improve future versions of the manual.

When you send your feedback, please include the following information:

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

For immediate assistance with technical issues, please contact Altibase's Customer Support site (http://support.altibase.com/).

We always appreciate your comments and suggestions.

1 aexport

1.1 Overview

1.1.1 About aexport

`aexport` supports automated data migration between ALTIBASE HDBs. 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.

Since `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.).

1.1.2 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

At execution, `aexport` creates SQL scripts to create these database elements and shell scripts to run them.

1.1.3 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 following modes are available. SQL script files are generated for each mode.

1.1.3.1 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:

- `ALL_ALT_TBL.sql`: Switches the data access mode for tables and partitions of all users.
- `ALL_CRT_DIR.sql`: Creates all directory objects.
- `ALL_CRT_FK.sql`: Creates all user-defined foreign keys.
- `ALL_CRT_INDEX.sql`: Creates all user indexes.
- `ALL_CRT_LINK.sql`: Creates all user-defined database link objects.
- `ALL_CRT_REP.sql`: Creates all replication objects.
- `ALL_CRT_SEQ.sql`: Creates all user-defined sequences.
- `ALL_CRT_SYNONYM.sql`: Creates all synonym objects.
- `ALL_CRT_TBL.sql`: Creates all user tables.
- `ALL_CRT_TBS.sql`: Creates all tablespaces.
- `ALL_CRT_TRIG.sql`: Creates all user-defined triggers.
- `ALL_CRT_USER.sql`: Creates all users and roles.
- `ALL_CRT_VIEW_PROC.sql`: Creates all views and stored procedures.
- `ALL_REFRESH_MVIEW.sql`: Refreshes all user materialized views.
- `SYS_CRT_USER.sql`: Creates all users and roles.

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

1.1.3.2 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}_ALT_TBL.sql` : Switches the data access mode for tables and partitions of the specified user.
- `{User name}_CRT_FK.sql`: Creates all foreign keys of the specified user.
- `{User name}_CRT_INDEX.sql`: Creates all indexes of the specified user.
- `{User name}_CRT_LINK.sql`: Creates all database link objects of the specified user.
- `{User name}_CRT_SEQ.sql`: Creates all sequences of the specified user.

- {User name}_CRT_TBL.sql: Creates all tables of the specified user.
- {User name}_CRT_TRIG.sql: Creates all triggers of the specified user.
- {User name}_REFRESH_MVIEW.sql: Refreshes all materialized views of the specified user.

1.1.3.3 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 file is generated for this mode:

- {User name}_{Object name}_CRT.sql: Creates the specified user object.

1.1.3.4 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_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.
- 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_refresh_mview.sh: Refreshes materialized views. 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.

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.

Note: Shell script files are not generated when aexport is executed in object mode.

1.1.3.5 aexport Properties and Script Files

This section discusses script files generated by aexport properties.

Please refer to 1.3 aexport Properties for more information.

- If 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.
- If TWO_PHASE_SCRIPT = ON, 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.

1.1.4 Setting aexport

aexport requires the following information to connect to a server:

- ALTIBASE_HOME environment variable

  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 further 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 further information about the aexport.properties file, please refer to 1.3 aexport

Properties.

It is typical for the ALTIBASE_HOME environment variable to be set automatically when the server is installed. The user is recommended to verify this setting for aexport to run properly.

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 and 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.

One exception is the NLS_USE option. On omission, the user is not prompted to enter a value, but the US7ASCII 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.

1.1.5 Environment Variables

1.1.5.1 ALTIBASE_HOME

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

1.1.5.2 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.

1.1.5.3 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.

1.1.5.4 ALTIBASE_NLS_USE

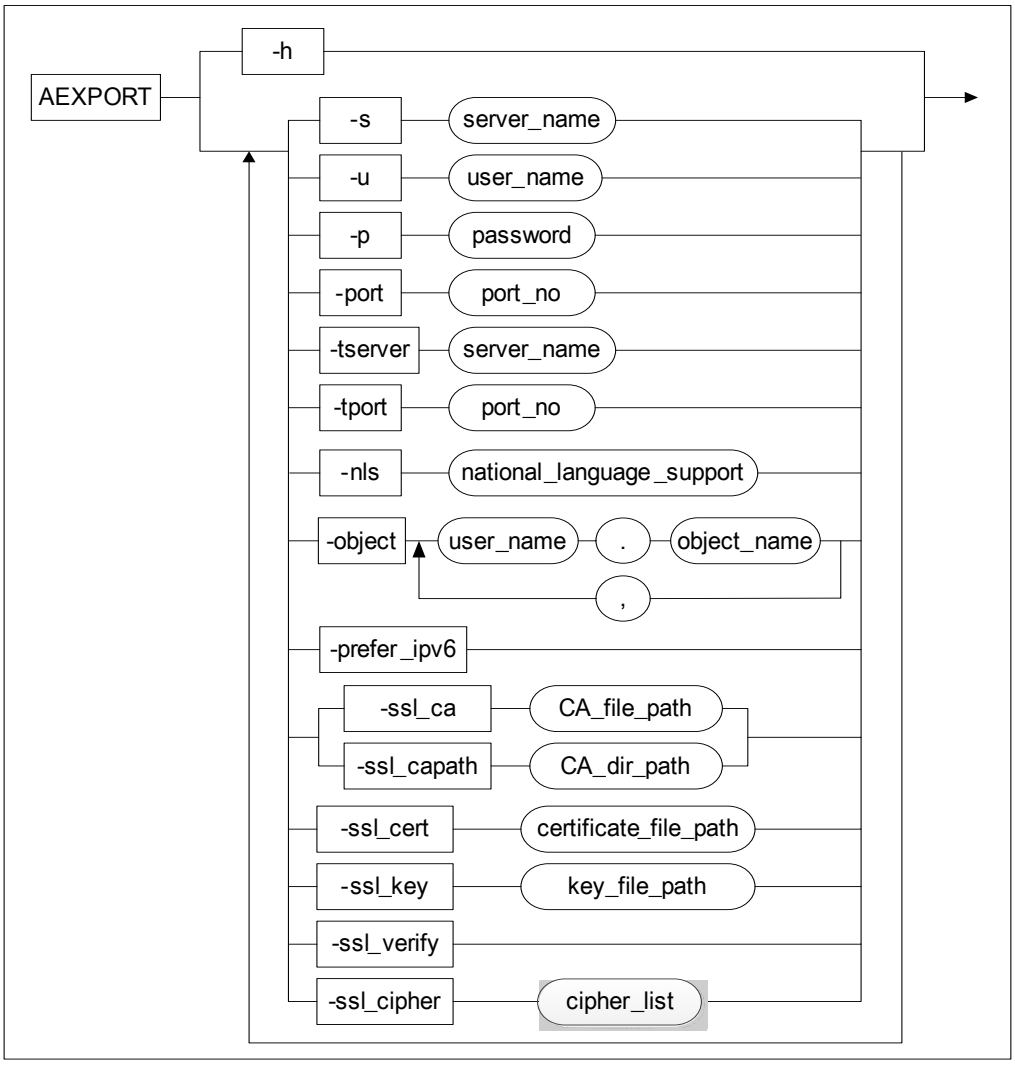
Sets the character setto use when connecting to the server. This can also be set with the `-nls` command-line option, or be set in advance in the `altibase.properties` file.

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` command-line option overrides both.

Note: On omission, the US7ASCII 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.

1.2 How to Use aexport

1.2.1 Syntax



1.2.2 Parameters

Parameter	Description
`-h`	Displays the Help menu

Parameter	Description
-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 HDB and IPv6 address notation, please refer to the *ALTIBASE HDB Administrator's Manual*.
-u	Sets the name of the ALTIBASE HDB user to access the server from which 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.
-p	Sets the password for the user. On omission, the user is prompted for a password.
-object	Sets the names of the objects to export and the object owners. If this option is specified, aexport runs in object mode.
-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.
-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	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, 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.

Parameter	Description
-ssl_ca CA_file_path	Specifies the location of the certification authority (CA) certificate in which the public key of the ALTIBASE HDB server to be connected to is incorporated.
-ssl_capath CA_dir_path	Specifies the directory under which the certification authority (CA) certificate in which the public key of the ALTIBASE HDB 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 further information about SSL connection, please refer to Chapter 2. Connecting and Disconnecting in the *iSQL User's Manual*.

1.2.3 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

1.2.3.1 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 HDB users at the prompts. This sets the password for each user that is created in the destination database.

  Note that if a password is specified using the USER_PASSWORD `aexport` property, that password is the default password for all users, and this step is skipped.

- 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 192.168.1.10 -port 21300 -u sys -p manager
  ```

1.2.3.2 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
  ```

1.2.3.3 Creating the Destination Database Structure

Create the required database objects in the destination database.

- 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.

1.2.3.4 Importing Data into the Destination Database

Import (upload) the 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*.

1.2.3.5 Creating Indexes and Foreign Keys

Create the required indexes and foreign keys in the target database.

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
```

1.2.4 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.

1.2.5 Limitations

1.2.5.1 Creating Stored Procedures using aexport

When a stored procedure is created, if any other stored procedures referenced by that stored procedure do not already exist in the database, the attempt to create the stored procedure will fail.

However, because `aexport` does not have access to information on the interdependencies between stored procedures, there is no guarantee that the stored procedures will be created in the correct order in the destination database. This means that some stored procedure creation attempts may fail.

If this happens, it will be necessary to create those stored procedures manually in the destination database.

1.2.5.2 Creating User-Defined Sequences using aexport

`aexport` only has limited access to meta data pertaining to sequences. Therefore, when `aexport` creates sequences that were originally defined by users other than the SYS user within their own schema, only the value of the INCREMENT BY property will be preserved; the other properties of user-defined sequences (namely, the START WITH, MAXVALUE, MINVALUE, CYCLE, and CACHE properties) will be set to their default values.

If this becomes an issue, it will be necessary to create user-defined sequences manually in the destination database.

1.2.5.3 SSL Connection and Script Files

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

SSL-related properties need to be set to connect to the database on SSL. For further information, please refer to SSL_ENABLE in aexport Properties.

1.2.6 Examples

1.2.6.1 Execution in Full DB Mode

```
$ aexport -s 127.0.0.1 -u sys -p manager
------------------------------------------------------------------
    Altibase Export Script Utility.
    Release Version 6.5.1.0.0
    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
```

1.2.6.2 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 6.5.1.0.0
    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
```

1.2.6.3 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 6.5.1.0.0
    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 6.5.1.0.0
    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 6.5.1.0.0
    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
```

1.3 aexport Properties

1.3.1 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 HDB 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.

1.3.2 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 destination 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 ON, `aexport` generates a script file named “INVALID.sql”, containing all of the scripts for creating the views and stored procedures that were found to be invalid.

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_CONSTRAINTS.sql, run_is.sh, and 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.

- 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 loaded 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.

- ISQL_REFRESH_MVIEW

  ```
  ISQL_REFRESH_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_REFRESH_MVIEW

  ```
  ISQL_REFRESH_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.)

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

- ILOADER_ROW_TERM

  ```
  ILOADER_ROW_TERM = row_term
  ```

  This property is used to set the delimiter between records that is used when the data in tables are saved as text. If it is not set, the default delimiter is the new line character(s).

  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.)

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

- ILOADER_PARTITION

  This property determines whether or not to create SQL scripts and shell scripts for creating partitions.

  ```
  ILOADER_PARTITION = ON/OFF
  ```

  If this property is set to ON, shell scripts for exporting data from partitions, for creating partitioned tables and all of their partitions, and for importing data into each partition are gener-

ated. In other words, enabling this property makes it possible to import data from table partitions in the source database into corresponding partitions in partitioned tables in the destination database.

If this property is set to OFF, whether tables in the source database are partitioned is ignored, and the shell script that is generated creates non-partitioned tables in the destination database and imports data from all partitions of partitioned tables in the source database into corresponding non-partitioned tables in the destination database.

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

- SSL_ENABLE

  Specifies whether to connect to the 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 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.

  ```
  Ex)
  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 discusses the `altiComp` utility and its features such as consistency control.

2.1 Overview

2.1.1 Concept

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

`altiComp` compares ALTIBASE HDB 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.

Note: *The Windows version of the ALTIBASE HDB server does not include the* `altiComp` *utility.*

2.1.2 Terminology

2.1.2.1 Master Server

This is the server whose contents are accepted as correct if a record is found to be inconsistent between two servers. Either server can be designated as the master server when `altiComp` is executed.

2.1.2.2 Master DB

The database on the master server.

2.1.2.3 Slave Server

This is the server whose contents are updated with the contents of the other server if a record is found to be inconsistent between two servers. Either server can be designated as the slave server when `altiComp` is executed.

2.1.2.4 Slave DB

The database on the slave server.

2.1.3 Inconsistent Records

An inconsistent record is a record in which a disagreement between column values is found when a designated table in the Master DB is compared with the corresponding table in the Slave DB on the basis of a primary key.

There are three types of inconsistency:

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

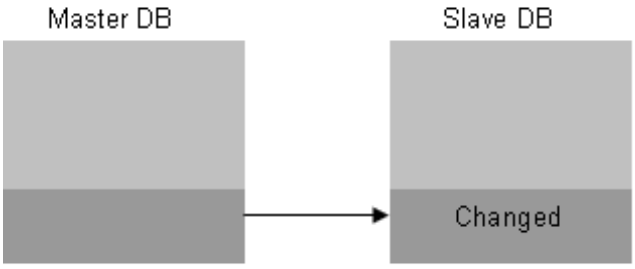
- MXSO inconsistency: When a record based on a primary key can be found in the Slave DB but not in the Master DB.

2.1.4 Synchronization Policy

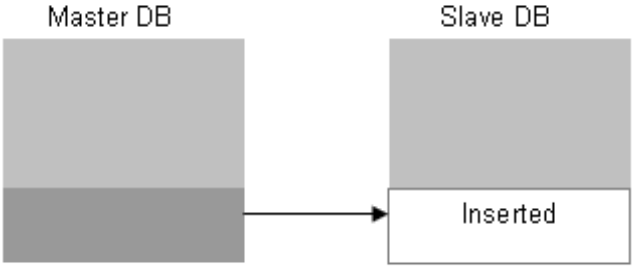
A synchronization policy is a policy that specifies how to synchronize inconsistent records. `alti-comp` usually treats the Master DB as the reference DB and synchronizes the Slave DB with it.

ALTIBASE HDB provides four synchronization policies:

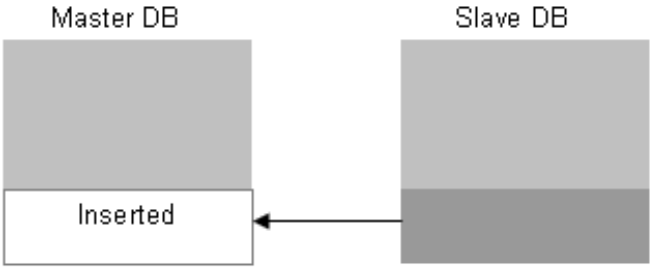
- SU Policy: This policy resolves MOSO inconsistencies by updating the Slave DB with the contents of the Master DB.



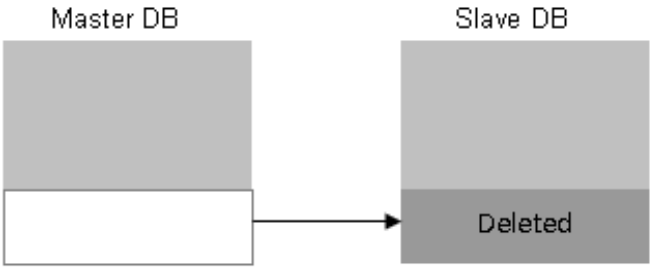
- SI Policy: This policy resolves MOSX inconsistencies by inserting records from the Master DB into the Slave DB.



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



- SD Policy: This policy resolves MXSO inconsistencies 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.

2.1.4.1 DIFF

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

2.1.4.2 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.

2.1.4.3 altiComp Environment File

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

2.2 How to Use altiComp

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

2.2.1 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 How to Use altiComp.

`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)
```

2.2.2 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.).

2.2.2.1 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:

- "**#**" 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 HDB, the following are special characters:

  ~ ! @ # $ % ^ & * ( ) _ + |

2.2.2.2 Property Names

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

2.2.2.3 Property Values

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

- An expression may include blanks. Most properties follow this format:

  ```
  e.g. 1) TABLE = EMPLOYEES
  ```

- 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.

  ```
  e.g. 2) EXCLUDE = ENO; DNO; E_FIRSTNAME
  or EXCLUDE = {ENO; DNO; E_FIRSTNAME}
  ```

- Expressions are character strings, can include spaces, and must be enclosed within "{}". The "WHERE" property is an expression.

  ```
  e.g. 3) WHERE = { ENO > '1000' and ENO < '2000' }
  ```

2.2.2.4 Data Type Support

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

```
Example 4) You should exclude a certain column from altiComp targets, if a
CLOB column exists in the EMP table.
TABLE = EMP
EXCLUDE = { CCC }
```

2.2.3 Property Options

You can 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.

2.2.3.1 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.

Set 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 HDB.

- TCP Connection

  ```
  DB_MASTER = altibase://sys:man-
  ager@DSN=192.188.1.1;PORT_NO=20300;NLS_USE=US7ASCII
  ```

- SSL Connection

```
DB_MASTER = altibase://sys:man-
ager@DSN=192.188.1.1;PORT_NO=${ALTIBASE_SSL_PORT_NO};NLS_USE=US7ASCII;CO
NNTYPE=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 further information about connection string attributes for SSL, please refer to the *SSL/TLS User's Guide*.

2.2.3.2 DB_SLAVE

This is used to set the other server.

Set 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 HDB.

Additionally, a text DB can be specified for the other server. In this case, the following format is used for this property (where "./log" is the directory containing the text DB).

```
DB_SLAVE = text://userID:PW@./log
```

2.2.3.3 OPERATION

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

2.2.3.4 INSERT_TO_SLAVE

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.

2.2.3.5 INSERT_TO_MASTER

Sets the MI policy used to resolve MXSO inconsistencies. Specifies whether to insert the 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.

2.2.3.6 DELETE_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.

2.2.3.7 UPDATE_TO_SLAVE

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.

2.2.3.8 AUTODETECT_UNIQ_INX

Specifies whether to delete the record in question from the Slave DB and repeat the insert or update action if a "Duplicate Key Values" error is raised in the Slave DB when inserting or updating data from the Master DB to the Slave DB. This property value may be set to "ON" or "OFF". "ON" signifies that the record is to be deleted, and "OFF" that it is not to be deleted.

This option can be set to "ON" only when both the INSERT_TO_SLAVE and DELETE_IN_SLAVE properties are also set to "ON".

2.2.3.9 CHECK_INTERVAL

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).

2.2.3.10 MAX_THREAD

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

2.2.3.11 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 ALTIBASE HDBs.

```
Ex) FILE_MODE_MAX_ARRAY = 1000
```

2.2.4 Table Group

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:

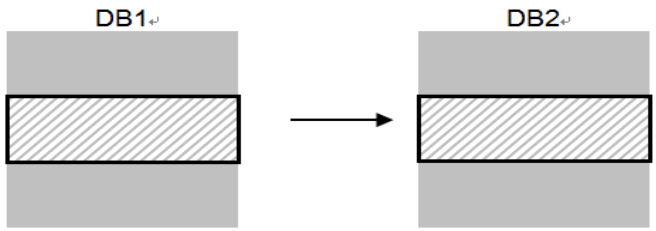
2.2.4.1 WHERE

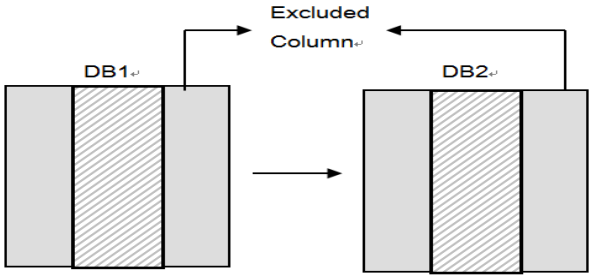
Sets conditions for selecting table records. This property is described in the same way as a WHERE clause of a SQL statement. Multiple values are permitted, but the ";" delimiter cannot be used to specify multiple values. Moreover, this property cannot be commented.

Applies to the comparison (DIFF) and synchronization (SYNC) functions.

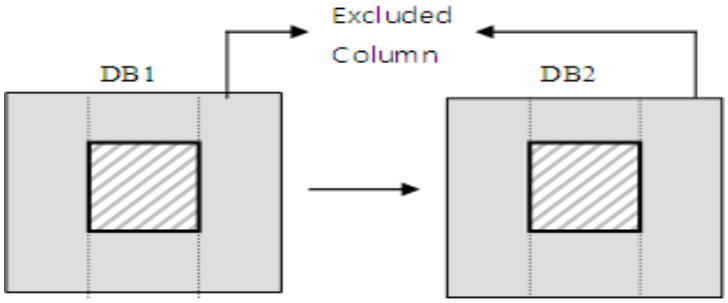
2.2.4.2 EXCLUDE

Sets conditions for the projection of table records. The property may have multiple values. The specified columns are excluded from comparison and synchronization operations.





By suitably combining WHERE and EXCLUDE properties, the result of a combined selection and projection operation can be obtained, and an `altiComp` task can be conducted on the result.



2.2.4.3 TABLE

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.

On omission, 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.

2.2.4.4 SCHEMA

Specifies the table schema in the Slave database. Table schema in ALTIBASE HDB contains information of a user's account. If the schema of a user accessing the Slave database differs from the target table's schema, this must be explicitly stated. If this is not stated, the error "Table not found" may occur when you try to execute `altiComp`. If this property is omitted, the schema of the user accessing the Slave database will be used.

2.3 Comparison (DIFF) Function

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

2.3.1 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.

2.3.2 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

2.3.3 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 you run the following `altiComp` command 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.

2.3.3.1 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
```

2.3.3.2 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.

2.3.4 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.

2.3.4.1 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:ma
ager@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
AUTODETECT_UNIQ_INX = 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.

2.3.4.2 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; GENDER}
[DEPARTMENTS]
```

2.3.4.3 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 = {GENDER}

[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.

2.4 Synchronization (SYNC) Function

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

2.4.1 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.

2.4.2 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

2.4.3 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 inconsistent records and an error log that includes information about errors which occurred during synchronization.

2.4.3.1 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.

2.4.4 Synchronization (SYNC) Examples

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

2.4.4.1 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:man-
ager@DSN=host1;PORT_NO=10111;NLS_USE=US7ASCII"
Slave Server = "altibase://sys:man-
ager@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
AUTODETECT_UNIQ_INX = 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.

2.4.4.2 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:man-
ager@DSN=host1;PORT_NO=10111;NLS_USE=US7ASCII"
Slave Server  = "altibase://sys:man-
ager@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
AUTODETECT_UNIQ_INX = 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.

2.4.4.3 SYNC Example 3

Synchronize the Master and Slave servers.

```
Master Server = "altibase://sys:man-
ager@DSN=host1;PORT_NO=10111;NLS_USE=US7ASCII"
Slave Server  = "altibase://sys:man-
ager@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
AUTODETECT_UNIQ_INX = 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”.

2.4.4.4 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.employ-
ees 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:man-
ager@DSN=127.0.0.1;PORT_NO=20582;NLS_USE=US7ASCII"
Slave Server  = "altibase://sys:man-
ager@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
AUTODETECT_UNIQ_INX = 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 SHMUTIL

3.1 Overview of SHMUTIL

3.1.1 Concepts

SHMUTIL is a utility for checking the status of shared memory that is allocated to ALTIBASE HDB and backing up and deleting databases in this shared memory.

```
shmutil {-p|w|e} [-d home_directory] [-f properties_file_path]
```

3.1.2 Features

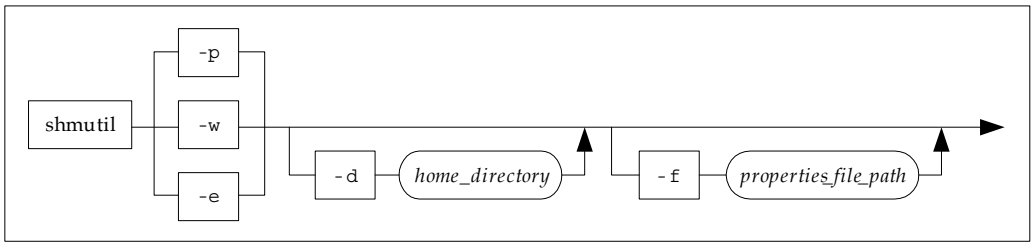
Memory for memory databases is allocated either from the process heap or from shared memory. When shared memory is allocated for storing a memory database, the database remains in the shared memory after ALTIBASE HDB has been shut down. SHMUTIL is therefore provided to check the status of and manage the data stored in shared memory.

SHMUTIL is used to perform the following tasks:

- Checking the status of shared memory
- Backing up a shared memory-resident database to disk
- Removing a database from shared memory

3.2 Using the SHMUTIL Utility

3.2.1 Syntax



3.2.2 Parameters

Parameter	Description
-p	Checks shared memory
-w	Backs up a database loaded into shared memory to disk
-e	Deletes a database from shared memory
-d	Sets the home directory path of ALTIBASE HDB. If this option is not specified, the directory specified by the $ALTIBASE_HOME environment variable is used.
-f	Specifies the location of the ALTIBASE HDB property file. If this option is not specified, the $ALTIBASE_HOME/conf/altibase.properties file is used.

3.2.3 Checking Shared Memory

Displays summarized information about the shared memory that is currently being used to store data, as well as detailed information about individual shared memory segments.

While ALTIBASE HDB is operating normally, the following message will be displayed:

```
$ shmutil -p

ShmUtil: Release 6.1.1.1 - Production on Oct  3 2010 04:52:01
(c) Copyright 2001 ALTIBase Corporation.  All rights reserved.


 ##################### IPC Information ######################
   SHM BASE KEY = 2046(0x7fe) ID = 327686(0x50006)

 ##################### Brief Information ####################
             SIZE(MB)     PAGE(#)
```

```
  TOTAL     4.0312        129
  USED      3.0000         96
  FREE      1.0312         33
  Timestamp is 1286347518(sec) / 833884(usec)


 ##################### Detail Information ######################
     ## Shared Memory Chunk ##
        Shm Key    = 2046
        Shm Id     = 327686
        Page Count = 129
        Size(MB)   = 4.0312
```

For detailed information about how to interpret this output, please refer to the *Administrator's Manual*.

3.2.4 Backing Up Shared Memory

Here is how to use SHMUTIL to back up a database that is loaded into shared memory to disk. ALTIBASE HDB must be completely shut down before the database in shared memory is backed up. In the following example, a database loaded in shared memory is backed up to a directory named *mydbkim*.

```
$ shmutil -w

ShmUtil: Release 6.1.1.1 - Production on Oct  3 2010 04:52:01
(c) Copyright 2001 ALTIBase Corporation. All rights reserved.


 !!!!!! WARNING !!!!!!
 Duplicated DB-Name will overwrite original DB-file.
 Use Different DB-Name.
 Original DB Name 1) => /home2/charlie/work/altidev4/altibase_home/dbs
 Input DB-Path => mydbkim
Tablespace File saving...SYS_TBS_MEM_DIC
Tablespace File saving...SYS_TBS_MEM_DATA
[SUCCESS] Database File Saved
```

3.2.5 Deleting a Database from Shared Memory

Here is how to use SHMUTIL to delete a database that is currently loaded into shared memory. ALTIBASE HDB must be completely shut down before the database is deleted. A database that is resident in shared memory is deleted as follows:

```
$ shmutil -e

ShmUtil: Release 6.1.1.1 - Production on Oct  3 2010 04:52:01
(c) Copyright 2001 ALTIBase Corporation.  All rights reserved.


Ready for Destroying Shared Memory? (y/N)y
Destroyed Shared Memory of Tablespace SYS_TBS_MEM_DIC.
Destroyed Shared Memory of Tablespace SYS_TBS_MEM_DATA.
[SUCCESS] All Shared Memory Removed
[SUCCESS] Database File Saved
```

3.2.6 References

ALTIBASE HDB Administrator's Manual

ALTIBASE HDB Starting User's Manual

4 Other Utilities

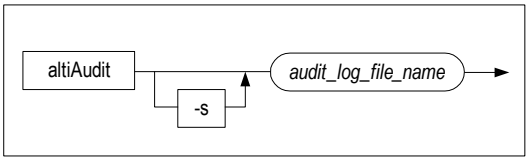
4.1 altiAudit

4.1.1 About altiAudit

When auditing is performed on the ALTIBASE HDB 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_file_name}
```

4.1.2 Syntax



4.1.3 Description

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.

4.1.4 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)"
```

4.1.5 Output

In the output, each field has the following meaning:

Field Name	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
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

Field Name	Type	Description
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
Validation time	BIGINT	The time consumed in validation
Optimization time	BIGINT	The time consumed in optimization
Execution time	BIGINT	The time consumed in execution

Field Name	Type	Description
Fetch time	BIGINT	The time consumed in fetching
SQL		
		The executed SQL statement

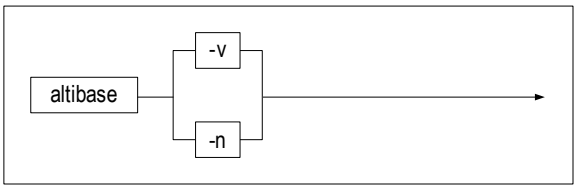
4.2 altibase

4.2.1 About altibase

`altibase` is the executable server file that controls all ALTIBASE HDB services.

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

4.2.2 Syntax



4.2.3 Parameters

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

4.2.4 Description

`altibase` is the executable server file that controls all ALTIBASE HDB services. To start up or shut down ALTIBASE HDB 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 4.18 server in this document. For a complete explanation of how to start up and shut down ALTIBASE HDB, please refer to the *ALTIBASE HDB iSQL User's Manual* or the *ALTIBASE HDB Getting Started*.

When the ALTIBASE HDB 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 HDB is executed in the foreground. This is used only for ALTIBASE HDB 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 HDB.

4.2.5 For More Information

Please refer to the *ALTIBASE HDB Getting Started*, the *ALTIBASE HDB Administrator's Manual*, and the

ALTIBASE HDB iSQL User's Manual .

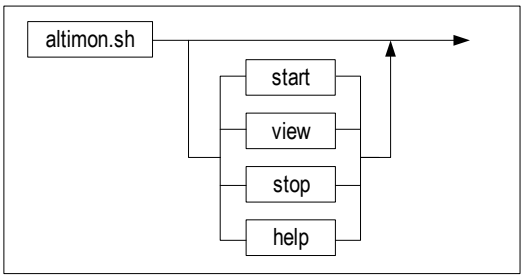
4.3 altimon.sh

4.3.1 About altimon.sh

`altimon.sh` monitors the status of the ALTIBASE HDB server process.

```
altimon.sh [ start | view | stop | help ]
```

4.3.2 Syntax



4.3.3 Parameters

Parameter	Description
start	Run `altimon` (this option can be omitted).
view	Information that is written to the log file is also echoed to the screen. This process continues until `altimon` is stopped.
stop	Terminates `altimon`
help	Outputs `altimon` help information.

4.3.4 Description

`altimon` continuously monitors the current status of the ALTIBASE HDB server process, threads, and system resources, and records related information to a log file. The log file created by `altimon` provides information that is useful when ensuring the stable operation of the system, and can be used to analyze the cause of any errors that may occur in the system.

`altimon` monitors the following items:

- The ALTIBASE HDB process
- The memory currently in use by ALTIBASE HDB
- The replication

- The sessions
- The transactions that are taking a long time to execute
- The Garbage Collector
- The log files

4.3.5 For More Information

Please refer to the *ALTIBASE HDB Administrator's Manual*.

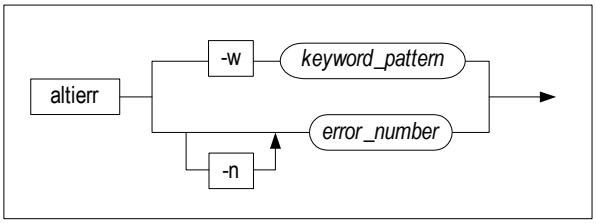
4.4 altierr

4.4.1 About altierr

`altierr` searches for and displays detailed descriptions of ALTIBASE HDB 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}
```

4.4.2 Syntax



4.4.3 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.

4.4.4 Description

`altierr` searches the ALTIBASE HDB 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 HDB 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:

```
Ex) 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:

```
Ex) 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"
```

4.4.5 For More Information

Please refer to the *ALTIBASE HDB Error Message Reference*.

4.5 altipasswd

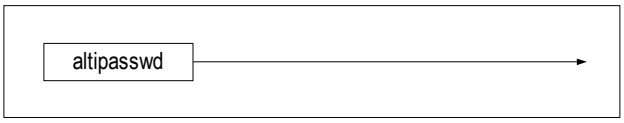
4.5.1 About altipasswd

`altipasswd` changesthe password of the SYS user, which is the user provided for accessing the database in SYSDBA mode.

The password of the SYS user must be changed using both this utility and the ALTER USER SQL statement. If the password of the SYS user is changed using only the ALTER USER statement, an error will occur when SYSDBA tasks, such as starting up and shutting down the database, are performed.

```
altipasswd
```

4.5.2 Syntax



4.5.3 Description

Changes the password of the SYS user.

4.5.4 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
```

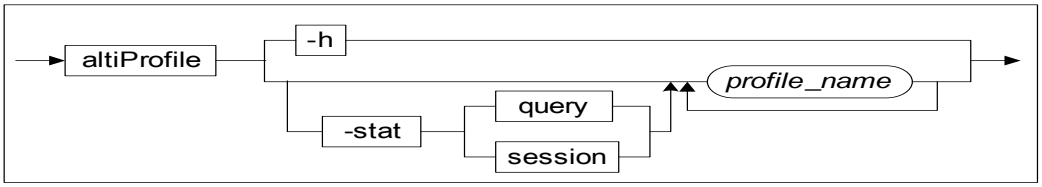
4.6 altiProfile

4.6.1 About altiProfile

ALTIBASE HDB 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] ...}
```

4.6.2 Syntax



4.6.3 Parameters

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

4.6.4 Description

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

4.6.5 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
```

4.6.6 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.
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:

```
alti-#time-#number.prof
```

The altiprofile command is used to convert this file into a form that can be analyzed.

4.6.6.1 Printing Statistics

If altiProfile is used with the –stat option, statistics are built on executed SQL statements and printed. This information helps to find SQL statements that require tuning.

The -stat query option builds statistics on the following:

- COUNT: The number of times the query was executed.

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

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 (
...
```

4.6.7 Precaution

If the QUERY_PROF_FLAG property is set so as to log information about all SQL statements that are executed on the server, and additionally to log the status of the server and information about all active sessions every 3 seconds, this can place a considerable load on the system. Furthermore, the size of the profile file will increase rapidly, which could cause the disk to become full, consequently causing problems. Therefore, care should be taken when considering whether to perform profiling.

The user should keep in mind that too many profiles can fill up the disk.

4.6.8 Output

Files are printed in the following format.

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

Each item is printed as follows.

[STATEMENT]

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

Table 4-1 [STATEMENT] Items

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 application
Elapsed Time for this SQL statement		
Total	BIGINT	The total query execution time (parsing, validating, optimizing, executing and fetching)
Parse	BIGINT	The time taken to parse the query
Valid	BIGINT	The time taken to validate the query

Field Name	Value	Description
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 Execution 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		
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

Field Name	Value	Description
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 SQL Tuning chapter in the *Administrator's Manual*.

[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* that explains performance views.

[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* that explains performance views.

[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* that explains performance views.

4.7 altiwrap

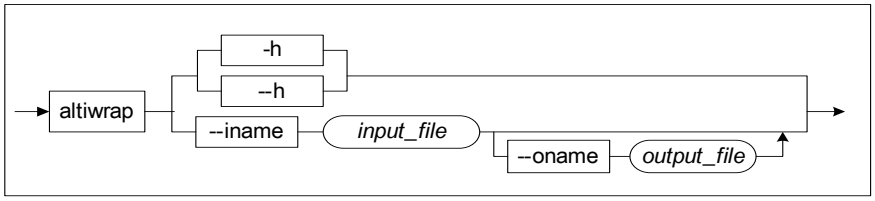
4.7.1 About altiwrap

`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]
```

4.7.2 Syntax



4.7.3 Parameters

Parameter	Description
`-h/--h`	Prints 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.

4.7.4 Description

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

ALTIBASE HDB 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`

4.7.5 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.

4.7.6 Example

Use altiwrap to encrypt the `sample1.sql` file, and then print 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
AAhjcmVhdGUgb3IgcmVwbGFjZSBwcm9jZWR1cqQBAAQxIGFzCnR5cGUgYXJyMSBpcyB0YWJs
QAYAAWYgY2hhcigzMCkgaW5kZXggYnlAAQZ0ZWdlcjsKdjGYBQAEOwpiZWdpbgp2MVswXSA6
PSAnY3JlYSu5ASdyBVsxtAN/
DHNldK0CMq4CaXO5ATO4AQNzdWNjZXNsAgZwcmludGxuKCC2C3x8YAFACaABVAegAVwGDHx8
ICchJyApOwplbmQ7ChEAADVBRDlBRkIzMDE0MzI1Q0U0MzY1RjYxNEI2NkYwQzRDREMzMTdD
QTU=
';
/
Create success.
iSQL> exec proc1;
create or replace typeset is success!
Execute success.
```

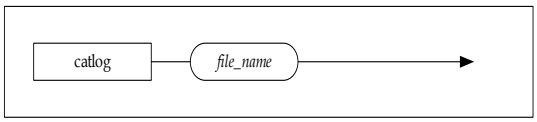
4.8 catlog

4.8.1 About catlog

`catlog` outputs an ALTIBASE HDB server trace log file to standard output (stdout).

```
catlog { file_name }
```

4.8.2 Syntax



4.8.3 Description

Outputs an ALTIBASE HDB server trace log file to standard output (stdout). Illegible parts (e.g., blanks) are excluded from the output.

A trace log file is a log file which is written to the $ALTIBASE_HOME/trc directory. The following text log files can be used with `catlog`.

- altibase_boot.log
- altibase_error.log
- altibase_mm.log
- altibase_qp.log
- altibase_rp.log
- altibase_sm.log
- altibase_dk.log
- altibase_xa.log
- altibase_ipc.log

4.8.4 Example

Enter the following command at the shell prompt:

```
$ catlog altibase_boot.log
```

The contents of the output can be manipulated with redirection operators (e.g., >, |) as shown below.

```
$ catlog altibase_boot.log > altibase_boot.txt
```

4.8.5 For More Information

Please refer to the *Administator's Manual*.

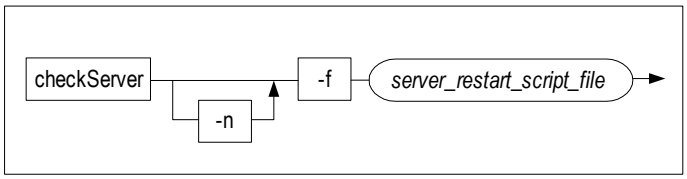
4.9 checkServer

4.9.1 About checkServer

`checkServer` monitors the ALTIBASE HDB process and, if ALTIBASE HDB terminates, executes a script specified by a user.

```
checkServer [-n] {-f server_restart_script_file}
```

4.9.2 Syntax



4.9.3 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 HDB terminates

4.9.4 Description

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

- The ALTIBASE HDB startup 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 HDB server is shut down without using the `server stop` command. When the ALTIBASE HDB 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.

4.9.5 Example

`checkServer` is executed from a shell prompt as follows:

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

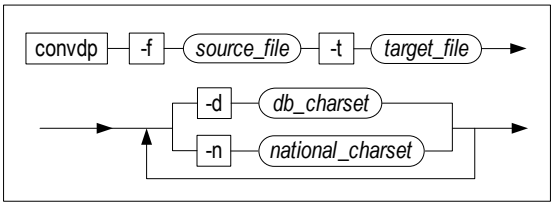
4.10 convdp

4.10.1 About convdp

convdp converts data from the DataPort file (.dpf) character set to the character set of the destination database.

```
convdp {-f source_file} {-t target_file}
       {[-d db_charset] [-n national_charset]}
```

4.10.2 Syntax



4.10.3 Parameters

Parameter	Description
-f	The name of the file containing the data to be converted
-t	The name of the file to which the converted data are to be written
-d	The database character set into which data having the CHAR and VARCHAR datatypes will be converted
-n	The national character set into which data having the NCHAR and NVARCHAR datatypes will be converted

4.10.4 Description

When the character set and/or the national character data of the data in a DataPort file (.dpf) differ from those of the destination database, the convdp utility can be used to convert the data in the DataPort file to the character set and/or the national character set of the target database.

A DataPort file contains header information, the definition of the source table or tables, and the actual data (records) in the table(s). The convdp utility converts data having the CHAR and VARCHAR data types to the specified character set, converts data having the NCHAR and NVARCHAR data types to the specified national character set, and writes the converted data to the specified target file.

When searching for the file on which to perform the conversion and determining the name of the

output file, convdp follows the DataPort naming conventions specified in the chapter of the *Stored Procedures Manual* in which DataPort is explained.

4.10.5 Example

convdp is used as follows:

```
iSQL> CREATE TABLE t1 (i1 INTEGER, i2 VARCHAR(100));
Create success.

iSQL> INSERT INTO t1 VALUES (1,'abc');
1 row inserted.

iSQL> EXEC EXPORT_TO_FILE ('SYS','T1','test');
Export - SYS_T1 1 record(s).
Execute success.

$ cd $ALTIBASE_HOME/dbs

% convdp -f test_0 -d 'UTF8' -t test_utf8

...

Convert Charset:
[0  ]INTEGER
[1  ]VARCHAR         (KSC5601->UTF8)

Convert 1 rows...
Done.
```

4.10.6 For More Information

Please refer to the *ALTIBASE HDB Stored Procedures Manual*.

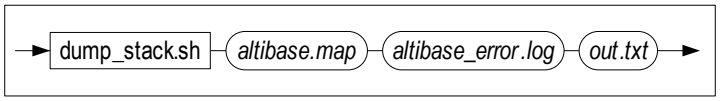
4.11 dump_stack.sh

4.11.1 About dump_stack.sh

When ALTIBASE HDB terminates abnormally, information about the ALTIBASE HDB process call stack is recorded in the `altibase_error.log` file. `dump_stack.sh` converts this information into a format that is understandable to the user.

```
dump_stack.sh altibase.map altibase_error.log out.txt
```

4.11.2 Syntax



4.11.3 Parameters

Parameter	Description
altibase.map	The symbol table queried from the altibase binary file with the `nm` command. It is located in the `${ALTIBASE_HOME)/bin` directory.
log_file_name (altibase_error.log)	The log file in which information about the call stack is recorded when ALTIBASE HDB terminates abnormally. It is located in the `${ALTIBASE_HOME}/trc` directory.
out.txt	The file to output the converted results.

4.11.4 Description

If, due to abnormal operation of the system or an undiscovered bug, an ALTIBASE HDB server becomes unable to provide service normally, it is necessary to analyze the circumstances at the time of the abnormal termination in order to resolve the issue and prevent future reoccurrences. When ALTIBASE HDB becomes unable to operate normally, it records the process call stack to `altibase_error.log` and terminates the server process. Information about the internal ALTIBASE HDB module that was active at the time that ALTIBASE HDB shut down will be stored in the process call stack.

However, this information includes only the subroutines' entries in memory address format within the process, and furthermore is recorded in a format that is not interpretable by users. Therefore, it needs to be converted into a format that can be understood. `dump_stack.sh` converts the process call stack information recorded in *altibase_booterror.log* into an interpretable format.

In the event of any unexplained abnormal shutdown, use `dump_stack.sh` to convert the process call stack information in this way and send the resultant data to the Altibase support team. This will enable us to resolve your issue as quickly as possible.

4.11.5 Notes

The `dump_stack.map`, `dump_stack.log` files are generated in the current directory when `dump_stack.sh` is executed. These are files which the `dump_stack.sh` script temporarily uses during the process of converting call stacks.

The `dump_stack.map` file contains relevant information from the *altibase.map* file and the `dump_stack.log` contains relevant information from the `altibase_error.log` file.

Since how to generate the `dump_stack.map` file differs for each OS, the user is required to manually modify the `dump_stack.sh` shell script if it does not operate normally.

Please use `gawk`, if `awk` in the `dump_stack.sh` shell script does not operate normally.

The `dump_stack.sh` shell script uses `c++filt` to convert overloaded functions in C++ into an interpretable format for the user. If `c++filt` does not operate normally, please delete `c++filt` from the `dump_stack.sh` shell script or check the path of `c++filt` with the `which` command.

4.11.6 Example

Ata shell prompt, type the following:

```
$ dump_stack.sh $ALTIBASE_HOME/bin/altibase.map $ALTIBASE_HOME/trc/
altibase_error.log out.txt
```

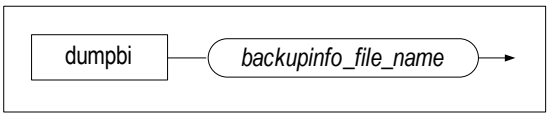
4.12 dumpbi

4.12.1 About dumpbi

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

`dumpbi <backupinfo_file_name>`

4.12.2 Syntax



4.12.3 Description

Outputs contents of the backupInfo file in text format.

4.12.4 Example

At a shell prompt, type the following:

```
$ dumpbi backupinfo
```

4.12.5 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	LFGID, 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
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_SIZE 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: differential 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

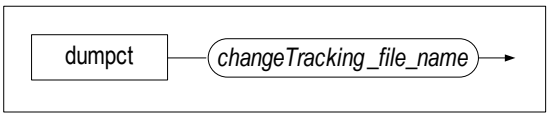
4.13 dumpct

4.13.1 About dumpct

`dumpct` outputs information of the changeTracking file which is recorded in binary format as text format.

`dumpct <changeTracking_file_name>`

4.13.2 Syntax



4.13.3 Description

Outputs contents of the changeTracking file in text format.

4.13.4 Example

At a shell prompt, type the following:

```
$ dumpct changeTracking
```

4.13.5 Output

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

[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.)
Incremental backup chunk size	From 0(zero) to the maximum value of the unsigned int type	The value of the INCREMENTAL_BACKUP_CHUNK_SIZE property at the time of the creation of the changeTracking file.

Field Name	Value (bytes)	Description
Last flush LSN	LFGID, 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	LFGID, FileNo, Offset	The LSN at the time that the body was flushed (for file validation)
Datafile descriptor slot		
Slot ID	From 0(zero) to the maximum value of the unsigned int type	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
Differential 0 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		

Field Name	Value (bytes)	Description
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

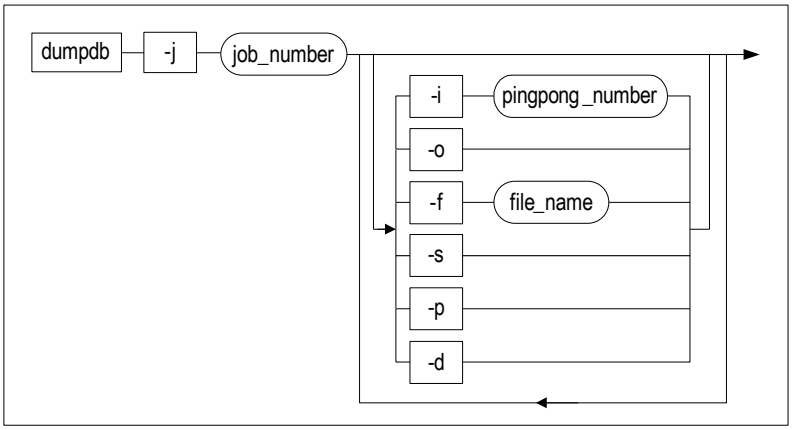
4.14 dumpdb

4.14.1 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]
```

4.14.2 Syntax



4.14.3 Parameters

Parameter	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.

Parameter	Description
-s	Specifies the ID of the tablespace to be analyzed.
-p	Specifies the ID of the page to be analyzed.
-d	Outputs detailed information.

4.14.4 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 HDB 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.

4.14.5 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 ]
LogFileGroup Count            [ 1 ]
LogFileGroup-0 Redo LSN       [ 0, 1, 5867599 ]
LogFileGroup-0 Create LSN     [ 0, 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.
```

4.14.6 Output

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

Table 4-2 Result items for dumpdb output

Field Name	Description
Binary DB Version	The version of the data file.
LogFileGroup Count	The numbers of LFG.
LogFileGroup-0 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.
LogFileGroup-0 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.

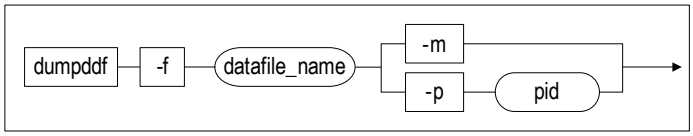
4.15 dumpddf

4.15.1 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]
```

4.15.2 Syntax



4.15.3 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.

4.15.4 Description

Outputs information about the data file in text form.

4.15.5 Examples

At a shell prompt, type the following:

```
$ dumpddf -f datafile -m
```

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

4.15.6 Output

The following is an example of `dumpddf` output:

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

In the output, each field has the following meaning:

Table 4-3 Result items for dumpddf output

Field 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                      [ 0, 1, 5867599 ]
Create LSN                    [ 0, 0, 1914 ]
MustRedo LSN                  [ 0, 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]
```

4.16 dumpla

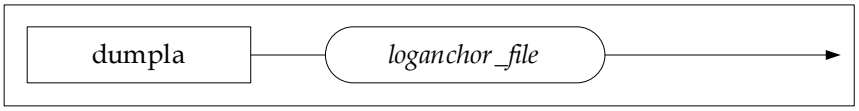
4.16.1 About dumpla

`dumpla` 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 HDB creates these files and stores them with the sequential names loganchor# (where "#" = 0, 1, or 2).

ALTIBASE HDB 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`*

4.16.2 Syntax



4.16.3 Description

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

4.16.4 Example

At a shell prompt, type the following:

```
$ dumpla loganchor0
```

4.16.5 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:

Table 4-4 [LOGANCHOR ATTRIBUTE SIZE] Items

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 Table 4-13 dumplf Output Items.

The contents of this section are as follows:

Table 4-5 [LOGANCHOR HEADER] Items

Field Name	Value	Description
Binary DB Version	Major.minor.patch Ex.) 5.4.1	The version of the database executable file with which the loganchor file was created
Archivelog Mode	Archivelog|No-Archivelog	Indicates whether the database is running in archive mode
Transaction Segment Entry Count		
Begin Checkpoint LSN	LFGID, FileNo, Offset	The LSN that was current when checkpointing most recently began.
End Checkpoint LSN	LFGID, FileNo, Offset	The LSN that was current when checkpointing was most recently completed.
Disk Redo LSN	LFGID, FileNo, Offset	The redo start point for a DRDB.

Field Name	Value	Description
Global SN	From 0 (zero) to the maximum value of the unsigned long type	Each log has a global Sequence Number (SN) in its header. When a new log is created, the value set for its SN is the SN of the previous log incremented by 1 (previous SN + 1).
SN for Recovery from Replication	From 0 (zero) to the maximum value of the unsigned long type or NULL Refer to Table 4-13 dumplf Output Items.	This information is useful when the recovery option, which is an extra feature of replication, has been enabled. It is the sequence number (SN) at which recovery using replication must begin in the event of a fault on an active server.
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.
Log File Group Count	1	The number of log file groups
Log File Group	0	The LFG identifier for the the End LSN, ResetLog LSN, Last Created Logfile Num, and Delete Logfile(s) Range fields described below.
End LSN	LFGID, 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	LFGID, 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.

Field Name	Value	Description
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
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:

Table 4-6 [TABLESPACE ATTRIBUTE] Items

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. *Note: If FREE EXTENT BITMAP TABLESPACE is enabled, bitmaps can be used to manage the free extents in a disk tablespace.*
Tablespace Status	Refer to Table 4-7 Possible Tablespace Status Values in [TABLESPACE ATTRIBUTE].	Indicates the current status of the tablespace

Field Name	Value	Description
Tablespace Type	0 - 8 (Refer to Table 4-8 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
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. This applies only to memory tablespaces
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). This applies only to memory tablespaces
Maximum Size	From 0 (zero) to the maximum value of the unsigned int type	The maximum size of the tablespace. This applies only to memory tablespaces
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. This applies only to memory tablespaces

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

Table 4-7 Possible Tablespace Status Values in [TABLESPACE ATTRIBUTE]

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 brought online
SWITCHING_TO_ONLINE	Being taken offline

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

Table 4-8 Possible Tablespace Type Values in [TABLESPACE ATTRIBUTE]

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:

Table 4-9 [MEMORY CHECKPOINT PATH ATTRIBUTE] Items

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 image 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:

Table 4-10 [MEMORY CHECKPOINT IMAGE ATTRIBUTE] Items

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	<LFGID, 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
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:

Table 4-11 [DISK DATABASE FILE ATTRIBUTE] Items

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	<LFGID, FileNo, Offset>	The LSN that was current at the time that the data file was created
Database File Status	Refer to Table 4-12 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 current 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
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:

Table 4-12 Database File Status Values for [DISK DATABASE FILE ATTRIBUTE

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, 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	LFGID, FileNo, Offset	The LSN at the time data changed in memory were written to files
Change Tracking Manager State	String e.g.) CHANGE TRACKING MGR ENABLED	The page change tracking state
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
Log File Group	LFGID	LFGID
Delete Archivelog File Range	LFGID, FileNo, Offset	The number of the archive log file which can be completely recovered, even after deletion
Last Backup LSN	LFGID, FileNo, Offset	The LSN at the time of the most recently performed backup
Before Backup LSN	LFGID, FileNo, Offset	The LSN prior to the time of the most recently performed backup
Backup Info Manager State	String e.g.) 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

4.17 dumplf

4.17.1 About dumplf

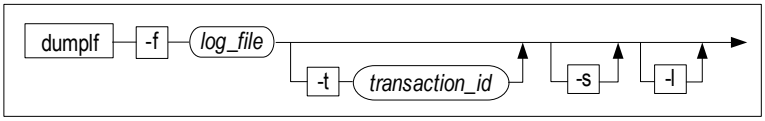
When a transaction performs an operation that changes the contents of the database, such as an INSERT, DELETE or UPDATE operation, changes are made not only to the database's data buffers, but also to log files. These files are maintained for use in performing recovery if it becomes necessary.

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} [-t transaction_id] [-s] [-l]
```

4.17.2 Syntax



4.17.3 Parameters

Parameter	Description
-f	Specifies the name of the file whose contents are to be output. This option must be given. If it is omitted, `dumplf` will terminate and output an error message.
-t	Specifies the ID of the transaction 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-log-types (OPTYPE and UTYPE fields) in the specified log file.

4.17.4 Description

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

4.17.5 Example

At a shell prompt, type the following:

```
$ dumplf -f logfile0
```

4.17.6 Output

The following is an example of `dumplf`output:

```
SN=<10>,LSN=<?,?,820>, COMP:N, MAGIC:820, TID: 6400,BE: N, REP: Y, ISVP: N,
ISVP_DEPTH: 0 PLSN=<0,0,739>, LT: SMR_LT_MEMTRANS_COMMIT, SZ: 45
```

Each field in a log file has the following meaning:

Table 4-13 dumplf Output Items

Field Name	Value	Description
SN	From 0 (zero) to the maximum value of the unsigned long type	This is the sequence number that is assigned to each log sequentially
LSN	Format: (LFGID, 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 Log File Group (LFG), the file number and an offset value. When `dumplf` cannot determine the LFG and/or FileNo, question marks (“?”) are output. Ex) `(?,?, Offset)`
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 this log is a Begin Transaction Log, which is recorded when a transaction is started.
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.

Field Name	Value	Description
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
PLSN	Format: (LFGID, 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 Refer to Table 4-14 Possible LT Values in dumplf Output.	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 internal value that is used for replication
OPTYPE	LogTypeName<LogTypeNumber> Refer to Table 4-15 Possible LogTypeName Values for OPTYPE and UTYPE.	The operation type of a Nested Top Action (NTA) log
AFTER	SZ: <size>, Value: <value>	The after image of the log record

Field Name	Value	Description
BEFORE	SZ: <size>, Value: <value>	The before image of the log record
UTYPE	LogTypeName<Log-TypeNumber> Refer to Table 4-15 Possible Log-TypeName Values for OPTYPE and UTYPE.	The operation type of an UPDATE log
UPOS	Format: (SPA-CEID:<SpaceID>, PID:<PageID>, OFF-SET:<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
Offset	From 0 (zero) to the maximum value of the unsigned int type	The offset from the beginning of the page con-taining the object that was updated
FLISlot PrevPID NextPID	Format: (<BeforePID> => <AfterPID>)	An internal value used for managing MMDB tablespaces
ESLSN OF LFG	Format: (LFGID, 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 internally 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:

Table 4-14 Possible LT Values in dumplf Output

Value	Description
SMR_LT_DUMMY	Dummy Log
SMR_LT_CHKPT_BEGIN	Checkpoint Begin Log
SMR_LT_DIRTY_PAGE	Dirty Page Log
SMR_LT_CHKPT_END	Checkpoint End Log

Value	Description
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_DDL_QUERY_STRING	DDL Query String
SMR_LT_UPDATE	MMDB (Main Memory Database) Update Log
SMR_LT_NTA	MMDB NTA (Nested Top Action) Log
SMR_LT_COMPENSATION	Compensation Log
SMR_LT_DUMMY_COMPENSATION	Dummy Compensation Log
SMR_LT_FILE_BEGIN	File Begin Log
SMR_LT_TBS_UPDATE	Tablespace Update Log
SMR_LT_FILE_END	File End Log
SMR_DLT_REDOONLY	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

Table 4-15 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

Value	Description
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
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

Value	Description
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
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.

Value	Description
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
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

Value	Description
SDR_STNDR_INSERT_INDEX_KEY SDR_STNDR_UPDATE_INDEX_KEY SDR_STNDR_FREE_INDEX_KEY SDR_STNDR_INSERT_KEY SDR_STNDR_DELETE_KEY_WITH_NTA SDR_STNDR_FREE_KEYS SDR_STNDR_COMPACT_INDEX_PAGE	Logs related to MVCC for R-tree index keys in a DRDB
SDR_STNDR_MAKE_CHAINED_KEYS SDR_STNDR_MAKE_UNCHAINED_KEYS SDR_STNDR_KEY_STAMPING	Logs related to R-tree indexes in a DRDB
SMR_PHYSICAL	Physical logs in an MMDB
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

Value	Description
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

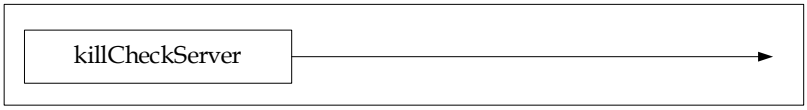
Please refer to the *Atibase Administrator's Manual* for more information about MVCC.

4.18 killCheckServer

4.18.1 About killCheckServer

`killCheckServer` terminates the `checkServer` utility if it is currently running.

4.18.2 Syntax



4.18.3 Description

`killCheckServer` terminates the checkServer utility if it is currently running.

If the `server stop` command was issued to stop the ALTIBASE HDB server, the server script terminates `checkServer` process by executing this utility before terminating the ALTIBASE HDB 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` was running:

  ```
  checkServer killed
  ```

- When `checkServer` was not running:

  ```
  ERROR CODE : -27
  ```

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

4.18.4 Example

At a shell prompt, enter the following:

```
$ killCheckServer
```

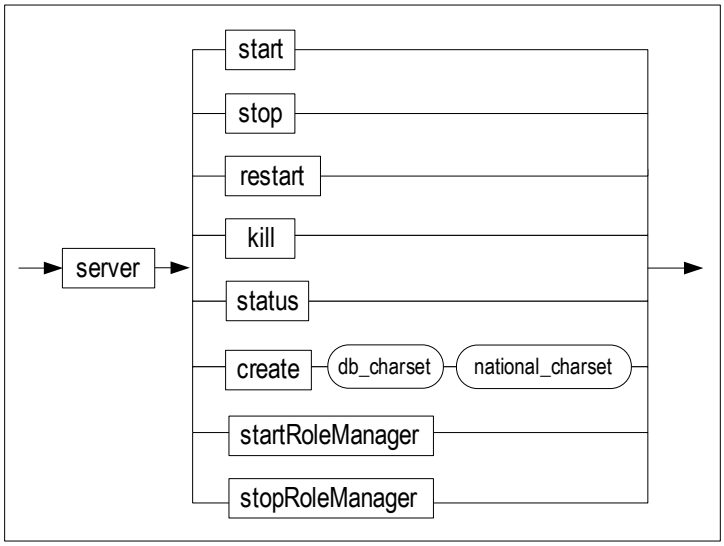
4.19 server

4.19.1 About server

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

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

4.19.2 Syntax



4.19.3 Parameters

Parameter	Description
start	Starts up the ALTIBASE HDB process
stop	Shuts down the ALTIBASE HDB process
restart	Restarts the ALTIBASE HDB process
kill	Forcibly terminates the ALTIBASE HDB process
status	Displays the status of the ALTIBASE HDB process
create	Creates a database that is 10MB in size, runs in noarchivelog mode, and uses the specified character sets

Parameter	Description
startRoleManager	Starts the ALTIBASE HDB process as the role manager for Disaster Recovery
stopRoleManager	Terminates the role manager

4.19.4 Description

Typically, iSQL is used to execute SQL statements for creating, starting up and shutting down ALTIBASE HDB. 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 HDB process
- Shutting down the ALTIBASE HDB process
- Restarting the ALTIBASE HDB process
- Forcibly terminating the ALTIBASE HDB process
- Displaying the result of querying "select * from tab;"
- Creating an ALTIBASE HDB
- Starting the ALTIBASE HDB process as the role manager
- Terminating the Role Manager

For more information about using SQL to manage ALTIBASE HDB databases, please refer to the *SQL Reference*. For further information on the role manager, please refer to the *Disaster Recovery Manual*.

4.19.5 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
```

4.19.6 For More Information

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

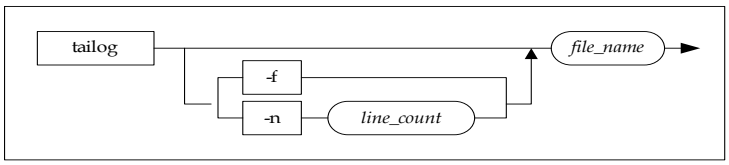
4.20 tailog

4.20.1 About tailog

`tailog ( or taillog )` outputs the last part of a trace log file that is written while the ALTIBASE HDB server is running to standard output (stdout).

```
tailog [-f | -n line_count] { file_name }
```

4.20.2 Syntax



4.20.3 Parameters

Parameter	Description
-f	Runs in Follow mode. Outputs an added log message whenever the trace log file is written.
-n	Specifies the number of lines to be output from the tail of the log. Default value: 10

4.20.4 Description

Outputs the last part of a trace log file that is written while the ALTIBASE HDB SERVER is running to standard output (stdout). Illegible parts (e.g., blanks) are excluded from the output.

The -f parameter can be used to monitor the trace log in real time. If you run the `tailog` command with the -f parameter, an added log message is output whenever the trace log file is written.

The -n parameter can be used to specify the number of lines to be output. On omission, the default value is 10 and the last 10 lines of the trace log file are output.

For further information on the trace log files that can be used with `tailog`, please refer to the description for `catlog`.

4.20.5 Example

Enter the following command at the shell prompt:

```
$ tailog -f altibase_boot.log
```

4.20.6 For More Information

Please refer to the *Administator's Manual*.

Index