SYBASE®
An SAP® Company

Users Guide

Replication Server® Data Assurance Option 15.7.1

DOCUMENT ID: DC01636-01-1571-01
LAST REVISED: April 2012

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

Conventions **1**
Replication Server Data Assurance Option **3**
System Architecture 3
Data Assurance Agents 6
Connection Profiles 7
Comparesets 7
Jobs and Comparisons 7
Comparison and Reconciliation Strategies 8
Core Options 9
Job History 12
Data Assurance System Database 12
Integration with Replication Server 13
Getting Started **15**
Single-Server Deployment 15
Distributed Deployment 22
Database Table Reconciliation 24
Administrative Tasks **29**
Creating a Job 29
Creating a Schema Job 29
Importing a Job from Replication Server 30
Setting Server Configuration Parameters 31
Backing Up and Restoring the DASD 31
Deleting Data and Log Files 31
Data Assurance Server Command Reference **33**
Agent Commands 33
alter agent 33
create agent 34
depend agent 34
drop agent 35
show agent 35
show agent connection 36

show agent dts 36
show agent jvm 37
show agent system 38
show agent task 38
test agent 39
Connection Profile Commands 39
alter connection 40
create connection 41
depend connection 43
drop connection 44
show connection 44
test connection 45
Compareset Commands 45
alter compareset 46
create compareset 48
depend compareset 51
drop compareset 51
show compareset 52
Row Comparison Job Commands 53
alter job 53
create job 59
drop job 64
show job 65
Schema Comparison Job Commands 65
alter schema job 65
create schema job 68
drop schema job 70
show schema job 70
Managing Job Commands 71
abort job 71
disable job 71
drop history 72
enable job 72
monitor job 73
run job 74

show history 75
show reconcile 76
show report 77
truncate history 77
Import Job Command 78
import job 78
Data Assurance System Database (DASD) Commands 83
create backup 83
drop backup 83
restore backup 84
show backup 84
truncate backup 84
Other Commands 85
config 85
password 95
role 95
show jvm 96
show system 97
sslconfig 97
trace 100
version 100
Reserved Words for Data Assurance Server 101
Remote Data Assurance Agent Command Reference ...103
config 103
password 104
role 105
show connection 106
show dts 106
show jvm 107
show system 107
show task 108
sslconfig 108
trace 110
version 111

Reserved Words for Data Assurance Agent 112
Security and Access Control 113
Kerberos Security 113
Configuring DA Agent for Kerberos 113
LDAP Authentication 114
DA Administrator Role 114
Configuring DA for LDAP Authentication 115
SSL Security 116
SSL Overview 117
Enabling SSL 117
Password Administration 120
Password Policy 120
Resetting a Lost or Forgotten Password 121
Password Encryption 122
Performance and Tuning 123
General Settings 123
Row Comparison Optimization 124
Troubleshooting 127
Approximate Numeric Datatypes Comparison 127
DA Server Out of Memory Errors 127
External Sort Option Configuration 128
Comparison Fails to Record an Inconsistency 128
Job Comparison Stops Responding 129
Comparison Fails with Stack Space Error 129
Comparisons Against Compressed Tables Fail 130
Obtaining Help and Additional Information 131
Technical Support 131
Downloading Sybase EBFs and Maintenance Reports 131
Sybase Product and Component Certifications 132
Creating a MySybase Profile 132
Accessibility Features 132
Glossary 135
Index 137

Conventions

Learn about the style and syntax conventions used in Sybase® documentation.

Style Conventions

Key	Definition
`monospaced(fixed-width)`	• SQL and program code • Commands to be entered exactly as shown • File names • Directory names
`italic monospaced`	In SQL or program code snippets, placeholders for user-specified values (see example below).
italic	• File and variable names • Cross-references to other topics or documents • In text, placeholders for user-specified values (see example below) • Glossary terms in text
bold san serif	• Command, function, stored procedure, utility, class, and method names • Glossary entries (in the Glossary) • Menu option paths • In numbered task or procedure steps, user-interface (UI) elements that you click, such as buttons, check boxes, icons, and so on

If necessary, an explanation for a placeholder (system- or setup-specific values) follows in text. For example:

Run:

```
installation directory\start.bat
```

where *installation directory* is where the application is installed.

Syntax Conventions and Command Rules

Key	Definition
{ }	Curly braces indicate that you must choose at least one of the enclosed options. Do not type the braces when you enter the command.
[ ]	Brackets mean that choosing one or more of the enclosed options is optional. Do not type the brackets when you enter the command.
( )	Parentheses are to be typed as part of the command.
|	The vertical bar means you can select only one of the options shown.
,	The comma means you can choose as many of the options shown as you like, separating your choices with commas that you type as part of the command.
...	An ellipsis (three dots) means you may repeat the last unit as many times as you need. Do not include ellipses in the command.

- Valid characters for input values are `_` , `a-z`, `A-Z`, `0-9`, `-`, and `:`. All other characters must be within quotes. Any input value string that contains a blank space must be within quotes (single or double).
- The keyword **go** is treated as a command terminator.
- Use **reset** to clear the command buffer.
- Use "--" or "#" to comment out a single line in the script.

Case-Sensitivity

- All command syntax and command examples are shown in lowercase. However, Replication Server® Data Assurance Option command names are not case-sensitive. For example, **CONFIG**, **Config**, and **config** are equivalent.
- Names of configuration parameters are not case-sensitive. For example, **MAX_CONCURRENT_COMPARISONS** is the same as **max_concurrent_comparisons**.
- User connection properties are case-sensitive. For example:

```
alter connection myconn
 with properties
  set you=sybase and set YOU=sybase
```

- Adaptive Server® database objects are case-sensitive. Use the correct case for table names when you specify database objects in your DA configuration.

Replication Server Data Assurance Option

Replication Server Data Assurance Option compares row data and schema between two or more Adaptive Server® Enterprise databases, and reports discrepancies.

Replication Server Data Assurance Option is a scalable, high-volume, and configurable data comparison product. It allows you to run comparison jobs even during replication by using a "wait and retry" strategy that eliminates any down time.

Each comparison job lets you check for data discrepancies using a number of settings that determine which data is being compared and in what way. Replication Server Data Assurance Option includes a command line tool (CLT) that allows users to perform all comparison and reporting jobs. Users can monitor and abort jobs, as well as generate detailed comparison reports. Replication Server Data Assurance Option is licensed through SySAM license manager and is available on multiple platforms.

For more information about SySAM, see the installation guide for your platform, or the SySAM Web site: *http://www.sybase.com/sysam*.

System Architecture

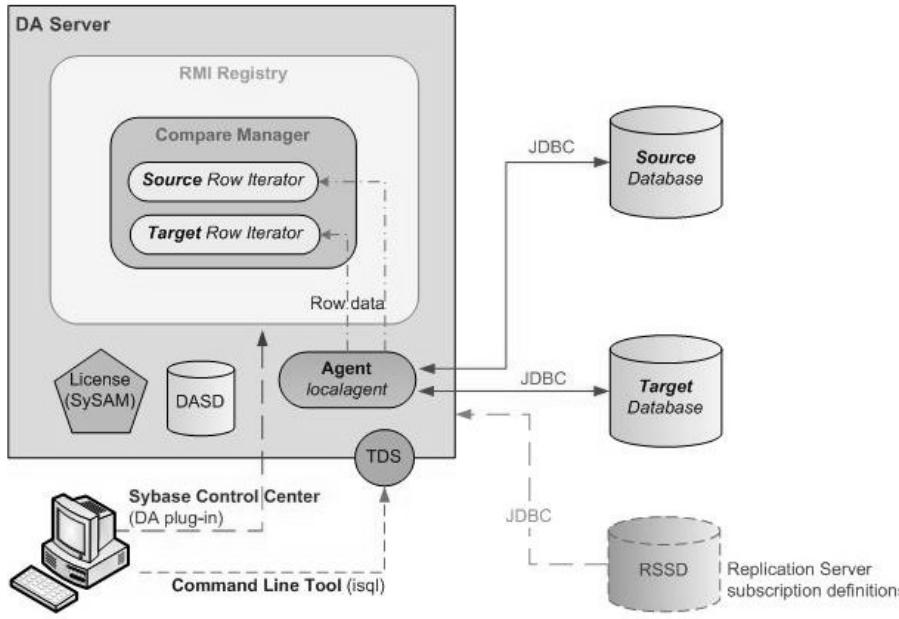
A Replication Server Data Assurance Option system has a central Data Assurance (DA) server component with zero or more satellite DA agents. The exact number of servers and agents depends on your deployment type: single-server or distributed.

Single-Server Deployment

The example architecture shown here illustrates a single-server deployment with:

- One DA server (with embedded agent)
- One primary (source) database
- One replicate (target) database
- Data Assurance System Database (DASD)
- Command line tool (CLT)
- Replication Server System Database (RSSD)
- Protocols used between components

Figure 1: Single-Server Deployment

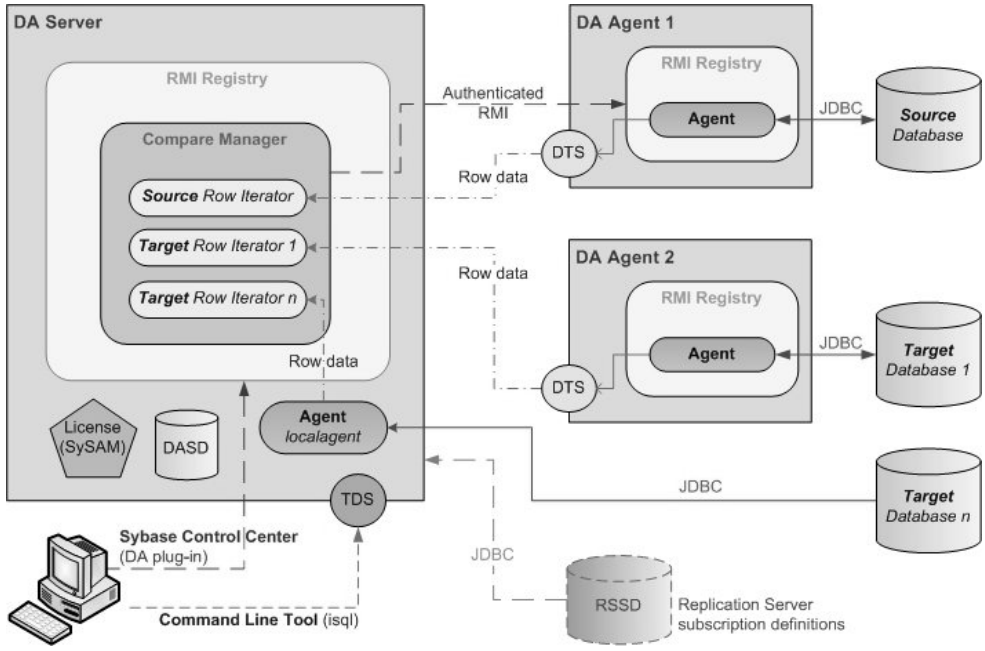


Distributed Deployment

The example architecture shown here illustrates a distributed deployment with:

- One DA server (with embedded agent)
- Two remote DA agents
- One primary (source) database
- Two replicate (target) databases
- Data Assurance System Database (DASD)
- Command line tool (CLT)
- Replication Server System Database (RSSD)
- Protocols used between components

Figure 2: Distributed Deployment



DA server is the core component and includes the DASD and an embedded DA agent, which is called the local agent. The remote DA agent stands alone, and is used by a DA server to enhance comparison job performance. Sybase recommends that you install a remote DA agent on a machine that shares a fast Ethernet connection with your database.

DA server:

- Compares rows
- Compares schemas
- Creates jobs
- Creates schedules for job
- Creates script for reconciliation
- Creates Data Manipulation Language (DML) commands for automatic reconciliation
- Checks job status
- Generates reports

Determine the number of DA agents to install based on your database or replication environment requirements. For example, you may choose to install one remote DA agent for your primary (source) database and one remote DA agent for your replicate (target) database. If you have different jobs that connect to different databases, you may decide to install multiple remote DA agents. Or you may have a single primary database and 10 replicate databases, and yet choose to perform the entire comparison task using the local agent.

To send commands to the DA server or the DA agent, use either:

- The interactive SQL (**isql**) tool. The **isql** parser lets you run multiple commands sequentially through an **isql** options file. See *Adaptive Server Enterprise > Utility Guide > Using Interactive isql from the Command Line*.
- The Sybase Control Center Data Assurance plug-in. See *Sybase Control Center 3.2.6 for Data Assurance Documentation*.

DA server integrates with Replication Server to automatically generate DA server jobs that are based on the information in the RSSD. Replication Server can import jobs only from table replication definitions; it cannot import jobs from database replication definitions.

DA server supports only one login, `da_admin`, which is assigned all administrator privileges.

See also

- *Data Assurance Agents* on page 6
- *Connection Profiles* on page 7
- *Data Assurance System Database* on page 12
- *Integration with Replication Server* on page 13
- *import job* on page 78

Data Assurance Agents

Data Assurance (DA) agents fetch and compress data from databases into the DA server.

There are two types of agents: the embedded local agent in DA server, and standalone (remote) agents, which DA server can use to improve job performance.

A DA agent opens a JDBC connection to one or more databases, and reads the row data used for comparison. If there are no standalone DA agents, you must use the embedded local agent.

Based on the DA server request, a DA agent:

- Compresses rows for precomparison, if configured to do so
- Fetches rows for comparison
- Hashes rows for precomparison, if configured to do so
- Sorts rows for precomparison, if configured to do so
- Runs **insert**/**delete**/**update** statements on databases, if configured to do so

See also

- *create agent* on page 34

Connection Profiles

A connection profile contains the information required to establish a database connection.

In DA server, create a connection profile to compare data between the source and target databases. The connection profiles, which contain login credentials, are stored in the Data Assurance System Database (DASD). Each database connection is owned by a single DA agent, and can be used any number of times by comparesets and schema jobs.

Note: You must set up connection profiles before you can create comparesets and schema jobs.

See also

- *create connection* on page 41

Comparesets

Comparesets, which consist of sets of tables and columns, define the data being compared in a particular job.

A compareset includes:

- Tables to compare
- Key columns that uniquely define a row
- Columns to compare
- A **WHERE** clause that defines which rows to compare

Source and target tables and columns need not use the same name in a compareset. Each compareset must have one source and one or more targets, and can be used by any number of jobs.

See also

- *create compareset* on page 48

Jobs and Comparisons

A job is a collection of one or more row or schema comparison tasks. You can create jobs manually, or automatically, based on the information in a Replication Server System Database (RSSD). You can run jobs manually, or schedule them to run at a specific time or interval.

A row comparison sorts rows by a primary or a unique key, from all participating database tables and compares them one by one. A comparison summary is stored in the Data Assurance

System Database (DASD). Detailed text or XML reports showing the row differences are stored in the data directory.

Schema comparison lets you compare the schema of one primary database against one or more replicate databases. You can compare an entire database schema using automatic table name mapping, or you can compare specific tables using table name mapping.

See also

- *create job* on page 59
- *create schema job* on page 68
- *Creating a Job* on page 29
- *Creating a Schema Job* on page 29

Comparison and Reconciliation Strategies

DA server comparison strategies and reconciliation help you plan and manage your row and schema comparison jobs.

The comparison and reconciliation phases in DA server include:

- Initial comparison
- In-flight data option
- Verify differences
- Reconciliation

Initial Comparison

During an initial comparison, which is mandatory for all jobs, the DA agent fetches rows from source and target databases using a query. You can specify row comparison in DA server by specifying any of these options:

- Column hash (`column_hash`) – each column value gets its own hash.
- Row hash (`row_hash`) – hashes multiple column values into a single hash.
- Literal compare – compares the full column data (value-to-value).
- Mixed-compare mode – compares some columns by hash, and others by literal comparison.

Note: Some, such as `column_hash` and `row_hash`, apply only to row comparison jobs.

In-Flight Data Option

Row differences may arise during comparison, due to data being in flight during replication. DA server lets you recheck row differences, by selecting row data only from the target database; you need not run a full table check.

Row differences are classified into three types:

- Missing – a row in the primary table is not present in the replicate table.

- Orphaned – a row in the replicate table is not present in the primary table.
- Inconsistent – a row is present in both tables, but the column data is different.

If DA server identified row differences in the initial comparison, an in-flight data comparison rechecks those rows to verify whether the differences have been reconciled. This is important, especially in replication environments where there are time lags in updating target databases.

In-flight data comparisons, which are optional, apply a “wait and retry” technique to any number of rows that shows data discrepancy during the initial comparison. For example, if an initial comparison at 8:00 p.m. reveals an out-of-sync row, and the wait period is 120 seconds, the recomparison is not started until 8.02 p.m to allow replication to apply any in-flight changes to that row.

Note: In-flight comparisons do not impact the source database. All source rows which differed are cached for recomparison against rows that are reselected from the target database.

Verify Differences

DA server fetches the literal data of all rows that differ between the source and target databases, and writes it to a column log. When you create a job, enable this option by setting **CREATE_COL_LOG** to true. A column log lists all the missing, orphaned, and inconsistent row values (keys and columns).

Reconciliation

Based on your job settings, you can reconcile the data differences—either automatically or by generating a reconciliation script. DA server verifies the differences and generates a SQL statement that ensures the target table is in the same state as the source table. Based on the row difference type, DA server runs:

- **insert** statements on the target table for missing rows.
- **delete** statements on the target table for orphaned rows.
- **update** statements on the target table for inconsistent rows.

See also

- *Row Comparison Optimization* on page 124

Core Options

DA server provides various comparison and job options, which you can use to optimize row and schema comparison queries.

Compressed Data Transfer (CDT)

During CDT, the row data between remote DA agent (excluding local DA agent) and DA server gets compressed thereby improving overall comparison time in distributed environments that have high network latency. Compressed data includes:

- All row data transmitted during the initial row or key comparison

- All retries of row or key comparison
- Verified differences of row or key comparison

CDT is optional for row comparison and not supported for schema comparison. Hashes do not compress well; the initial comparison and the in-flight data comparisons do not see much benefit in using CDT when the columns use the "column_hash" or "row_hash" option. However, literal data compresses well. CDT is beneficial to the initial comparison and in-flight data comparisons when the columns use the "literal" column option. The verify differences phase always benefits from using CDT.

Consider CDT when:

- Local area network (LAN) or wide area network (WAN) is a bottleneck
- Performing literal comparisons
- You expect hundreds or thousands (or more) of differences
- Your primary key column is large (as key columns are never hashed)

Do not choose CDT when:

- You use a local DA agent.
- LAN or WAN performance is not an issue.
- You always use hashing and either never use the “verify differences” option or you use it, but expect few or no differences.

To use this option, set **compress_data_transfer** to true.

External Sort Option

An **order by** clause specifies that a **select** statement returns a result set with the rows being sorted by the values of the key columns. DA server requires source and target table rows to be sorted before they can be compared. For very large tables, this sorting may have a large negative impact on the Adaptive Server temporary database space. To reduce the impact of processing **order by** clause in the databases, use the external sort option, which:

- Omits the **order by** clause and receives unsorted rows from the database.
- Sorts rows as they are written to flat files on your system.
- Reads simultaneously from all flat files and returns the sorted rows for comparison.

You can control and configure the external sort option by tuning the associated configuration parameters for best possible results.

To use this option, set **external_sort** to true.

Database Hash Comparison

Note: Adaptive Server supports the **hashbytes** function in version 15.0.2 and later. If your source or target database is earlier than 15.0.2, you cannot use the **database_hash** option.

The `database_hash` column comparison option does not support large objects (LOB), such as `text`, `image`, and `unitext`. This is because the Adaptive Server **hashbytes**

function does not accept LOB types as a parameter. All LOB types use a “first N bytes” parameter, where *N* is a configurable parameter. If the number of bytes in the LOB column is less than “N”, the entire column value is used.

Adaptive Server Hashbytes Null Handling

The Adaptive Server Transact-SQL® **hashbytes** function ignores null values.

For example, if a source table has `column_a=34` and `column_b=NULL` and a target table has `column_a=NULL` and `column_b=34`, the equality test is:

`hashbytes(34, NULL) = hashbytes(NULL, 34)`,

which computes as:

`hashbytes(34) = hashbytes(34)`, (a “false positive” match).

To manage the Adaptive Server **hashbytes** limitation, DA server provides a configuration parameter, **db_hash_ase_ignore_null**, to help reduce the chances of a “false positive” row match. Setting **db_hash_ase_ignore_null** to false eliminates this issue by adding an extra value to denote the “is null” state of a column. The above example becomes:

`hashbytes(0, 34, 1, NULL) = hashbytes(1, NULL, 0, 34)`,

which computes as:

`hashbytes(0, 34, 1) = hashbytes(1, 0, 34)`.

Data Reconciliation Option

DA server can fix data differences between your source and target databases. When creating a new job, DA server provides these two comparison options:

- **create_recon_script** – generates a script that includes **insert**, **update**, and **delete** statements when you set this option to true.
- **auto_reconcile** – generates and executes the **insert**, **update**, and **delete** statements on the database that requires reconciliation when you set this option to true.

Note: Set **create_col_log** to true for the reconciliation option to work.

Scheduling Option

When you create a comparison job, you can assign specific schedules to it. You can schedule a job based on days, weeks, or months. You can also set it using the UNIX clock daemon cron, which executes commands at a specified date and time.

Note: Although the schedule format is based on the cron, it does not use the UNIX cron command. DA server manages the scheduling.

See also

- *create job* on page 59
- *Creating a Job* on page 29

Job History

Each finished job generates a job report that includes information about the parameters used to compare source and target databases, and the results of the comparison.

The report is stored in text and XML file formats, in the `data` subdirectory under the Replication Server Data Assurance Option installation directory. The `data` directory is further classified by job name and timestamp for each job . For example:

```
C:\Sybase\DA-15_5\server\instance\data
\job2\2010-10-13\14.38.11.762\report.txt
```

where:

- *job2* – specifies the job name.
- *2010-10-13* – specifies the date when the job was submitted.
- *14.38.11.762* – specifies the time, in hours, minutes, seconds and milliseconds, when the job was submitted.

Each report file provide detailed information, which includes:

- Comparison options used
- Number of rows compared

The report files are generated when you run:

```
show history jobname historyid
```

If **create_col_log** is set to true, the XML and text report files contain the details of every difference.

See also

- *show history* on page 75

Data Assurance System Database

The Data Assurance System Database (DASD) stores all comparison information.

The DASD stores:

- System and configuration settings
- Agent connection, database connection, compareset, and job (including comparisons and schedules) configuration settings
- Task run history for reporting purposes

The DASD is located in `$SYBASE/DA-15_5/DA/server/instance/dasd/dasd.db`

See also

- *create backup* on page 83
- *Backing Up and Restoring the DASD* on page 31

Integration with Replication Server

Replication Server Data Assurance Option integrates with Replication Server to automatically import DA server jobs that are based on the information in the Replication Server System Database (RSSD). The DA server directly connects to the RSSD and retrieves the information about table replication definitions and subscriptions to determine the tables and columns that are defined for replication between the primary and replicate databases.

Note: Replication Server Data Assurance Option does not support database replication definitions.

Jobs that are imported from Replication Server includes:

- Comparison task name
- Primary (source) and replicate (target) databases
- Tables and columns – for the comparison SQL statement
- Schedule

Note: DA server does not automatically create a schedule for an imported job.

See also

- *import job* on page 78
- *Importing a Job from Replication Server* on page 30

Getting Started

Set up a single-server or distributed deployment for Replication Server Data Assurance Option.

These examples use the pubs2 Adaptive Server Enterprise sample database. You must install pubs2 on both Adaptive Server Enterprise servers. See *Adaptive Server Enterprise Installation Guide for Windows > Post Installation Tasks > Installing Sample Databases*.

Single-Server Deployment

Sybase recommends a single-server deployment when there is low network latency between the DA server and the database servers and when few concurrent comparisons are required. A single-server deployment is also easier to deploy and maintain than a distributed deployment.

Before You Begin

This example uses a single Data Assurance (DA) server with the local embedded agent. No remote DA agents are used.

Table 1. Deployment Summary

Component Name	Machine Name	Port Numbers
DA server	mars	• 4500 – RMI • 4501 – TDS • 4503 – DASD
Adaptive Server Enterprise	venus	5000 – server
Adaptive Server Enterprise	pluto	5000 – server

1. Start your DA server instance:

   ```
   $SYBASE/DA-15_5/server/instance/RUN_instance_64.sh
   ```

   Where $SYBASE is the directory in which you installed the Data Assurance Option, *instance* is the name of your DA server instance, and `RUN_instance_64.sh` is the start-up script.

   Note: On Windows, the start-up script file is named `RUN_instance_32.bat` or `RUN_instance_64.bat`, where *instance* is your DA server instance name. On UNIX or Linux platforms, the file is named `RUN_instance_64.sh`.

2. From **isql**, log in to DA server as an administrator:

```
$SYBASE/OCS-15_0/bin/isql -S mars:4501 -U da_admin -P password -w 250
```

Note: 4501 is the default TDS port number for DA server. The TDS port is required when the command line tool connects to the DA server using **isql**.

3. Create the database connections for the local DA agent:

```
create connection conn_venus
    set agent=localagent
    and set host=venus
    and set port=5000
    and set database=pubs2
    and set user=sa
    and set password=''
go
```

```
create connection conn_pluto
    set agent=localagent
    and set host=pluto
    and set port=5000
    and set database=pubs2
    and set user=sa
    and set password=''
go
```

Note: In this example, the embedded DA agent *localagent* connects to the Adaptive Server databases installed on *venus* and *pluto*. The user sa and null password are defaults for Adaptive Server Enterprise. For a null password, you can also omit the password parameter.

4. View the database connections defined within DA:

```
show connection
go

 NAME       TYPE AGENT      HOST  PORT DATABASE USER DESCRIPTION
 ---------- ---- ---------- ----- ---- -------- ---- -----------
 conn_venus ASE  localagent venus 5000 pubs2    sa
 conn_pluto ASE  localagent pluto 5000 pubs2    sa
(0 rows affected)
```

5. Test the database connections:

```
test connection conn_venus
go

RESULT
------
Succeeded
(0 rows affected)
```

```
test connection conn_pluto
go
RESULT
------
Succeeded
(0 rows affected)
```

6. Create a simple compareset for the authors table, comparing only the au_id, au_lname, and au_fname columns:

```
create compareset authors_demo1
with
source conn_venus dbo authors s1
target conn_pluto dbo authors t1
map
s1.au_id = t1.au_id set key=true
and s1.au_lname = t1.au_lname
and s1.au_fname = t1.au_fname
go
```

7. Create a more complex compareset using the **where** clause and comparing all columns in the authors table. Exclude from comparison all rows with a state column value of CA:

```
create compareset authors_demo2
with
source conn_venus dbo authors s1
where "state != 'CA'"
target conn_pluto dbo authors t2
where "state != 'CA'"
map
s1.au_id = t2.au_id set key=true
and s1.au_lname = t2.au_lname
and s1.au_fname = t2.au_fname
and s1.phone = t2.phone
and s1.address = t2.address
and s1.city = t2.city
and s1.state = t2.state
and s1.country = t2.country
and s1.postalcode = t2.postalcode
go
```

8. View the comparesets:

```
show compareset authors_demo1
go

 TYPE CONNECTION  OWNER TABLE   WHERE CONSTRAINT
 ---- ----------  ----- ------- ----------------
 S    conn_venus  dbo   authors
 T    conn_pluto  dbo   authors
(0 rows affected)
```

```
show compareset authors_demo2
go

 TYPE CONNECTION  OWNER TABLE   WHERE CONSTRAINT
 ---- ----------  ----- ------- ----------------
 S    conn_venus  dbo   authors state != 'CA'
 T    conn_pluto  dbo   authors state != 'CA'
(0 rows affected)
```

Note: To see compareset column mappings, use the **columns** option with the **show compareset** command. For example:

```
show compareset authors_demo1 columns
```

---

9. Create a row comparison job with default options using the authors_demo1 compareset:

```
create job authors_job1
 add comparison cmp_authors1
 set compareset=authors_demo1
go
```

10. Create another job using the authors_demo2 compareset, and set comparison options explicitly:

```
create job authors_job2
   set max_concurrent_comparisons = 10
   add comparison cmp_authors2
      set compareset=authors_demo2
      and set abort_diff_max to 1000
      and set abort_diff_row_count to true
      and set auto_reconcile to false
      and set compare_mode to row_compare
      and set compress_data_transfer to false
      and set create_col_log to false
      and set create_recon_script to false
      and set enable_row_count to true
      and set external_sort to false
      and set hash_type to database_hash
      and set priority to normal
      and set retry_delay_sec to 10
      and set retry_diff to wait_and_retry
      and set retry_max to 3
      with column option
         set city = literal
         and set postalcode to column_hash
go
```

Note: To change the job or comparison options, use **alter job**.

11. View the newly created job authors_job2:

```
show job authors_job2
go

OPTION                      VALUE
--------------------------- -----
MAX_CONCURRENT_COMPARISONS 10
(0 rows affected)

COMPARISON   ACTIVE COMPARESET    PRIORITY COMPARE MODE RETRY
------------ ------ ------------- -------- ------------
--------------
cmp_authors2 true   authors_demo2 NORMAL   ROW_COMPARE  WAIT_AND_RETRY

DESCRIPTION
-----------
(0 rows affected)

SCHEDULE ACTIVE TYPE EVERY START DATE TIME KEEP KEEP UNIT CRON
-------- ------ ---- ----- ---------- ---- ---- --------- ----
DESCRIPTION
-----------
```

```
(0 rows affected)
```

12. View the comparison cmp_authors1 for the newly created job authors_job1:

```
show job authors_job1 cmp_authors1
go

 OPTION                  VALUE
 ----------------------- -------------
 ABORT_DIFF_MAX          1000
 ABORT_DIFF_ROW_COUNT    true
 AUTO_RECONCILE          false
 COMPARE_MODE            ROW_COMPARE
 COMPRESS_DATA_TRANSFER false
 CREATE_COL_LOG          false
 CREATE_RECON_SCRIPT     false
 ENABLE_ROW_COUNT        true
 EXTERNAL_SORT           false
 HASH_TYPE               DATABASE_HASH
 RETRY_DELAY_SEC         10
 RETRY_DIFF              NEVER
 RETRY_MAX               3
(0 rows affected)

 COLUMN   COMPARE MODE
 -------- ------------
 au_fname ROW_HASH
 au_id    LITERAL
 au_lname ROW_HASH
(0 rows affected)
```

13. Execute the jobs:

```
run job authors_job1
go
(1 row affected)
```

```
run job authors_job2
go
(1 row affected)
```

14. Monitor the progress of the jobs:

```
monitor job authors_job1
go

COMPARISON   STATUS   SUBMIT TIME         END TIME            RUN PROGRESS
------------ -------- ------------------- ------------------- ---
--------
cmp_authors1 FINISHED 2011-11-15 21:26:20 2011-11-15 21:26:26 1   100%

NEXT RETRY ERROR
---- ----- -----
```

```
monitor job authors_job2
go

COMPARISON   STATUS   SUBMIT TIME         END TIME            RUN PROGRESS
------------ -------- ------------------- ------------------- ---
```

```
--------
cmp_authors1 FINISHED 2011-11-15 21:26:35 2011-11-15 21:26:36 1   100%

NEXT RETRY ERROR
---- ----- -----
```

15. Monitor the individual comparisons within each job:

```
monitor job authors_job1 cmp_authors1
go

COMPARISON   SUBMIT TIME         END TIME
------------ ------------------- -------------------
cmp_authors1 2011-11-15 21:33:28 2011-11-15 21:33:29

(0 rows affected)

 RUN PHASE       TYPE SUMMARY               START TIME
        END TIME          COUNT READ M O I R PROGRESS ESTIMATE END
ERROR
 --- ----------- ---- -------------------- -------------------

 1   COMPARE_ALL S    conn_venus/dbo.authors 2011-11-15 21:33:28
         2011-11-15 21:33:29 23    23           100%
                T    conn_pluto/dbo.authors 2011-11-15 21:33:28
         2011-11-15 21:33:29 23    23   0 0 0   100%

(0 rows affected)
```

```
monitor job authors_job2 cmp_authors2
go

COMPARISON   SUBMIT TIME         END TIME
------------ ------------------- -------------------
cmp_authors2 2011-11-15 21:35:46 2011-11-15 21:35:50

(0 rows affected)

RUN PHASE       TYPE SUMMARY               START TIME
--- ----------- ---- -------             --------------------
        END TIME            COUNT READ M O I R PROGRESS ESTIMATE
        ------------------  ----- ---- -------
 END ERROR
 --- -----
1   COMPARE_ALL S    conn_venus/dbo.authors 2011-11-15 21:35:46
         2011-11-15 21:35:46 8     8            100%
                T    conn_pluto/dbo.authors 2011-11-15 21:35:46
         2011-11-15 21:35:47 8     8    0 0 0   100%
(0 rows affected)
```

16. View a job history list:

```
show history authors_job1
go
 HISTORY ID SUBMIT TIME         FINISH TIME
 ---------- ------------------- -------------------
 3          2011-11-15 21:33:28 2011-11-15 21:33:29
 1          2011-11-15 21:26:19 2011-11-15 21:26:23
```

```
(0 rows affected)
```

17. To view an individual job history, specify the HISTORY_ID number for a job:

```
show history authors_job1 3
go

COMPARISON  RUN  PHASE        TYPE   SUMMARY
----------  ---  -----        ----   -------
  START TIME          END TIME               COUNT  READ M O I R
  ----------          --------               ------ ------------
ERROR
-----
cmp_authors1 1   COMPARE_ALL   S      venus:5000/pubs2.dbo.authors
 2011-11-15 21:33:28  2011-11-15 21:33:29     23     23
                               T      pluto:5000/pubs2.dbo.authors
 2011-11-15 21:33:28 2011-11-15 21:33:29      23     23  0 0 0
(0 rows affected)



 FILE        SERVER PATH
 ----------- ------------------------------------------------
---------------------------------
 Text report /Sybase/DA-15_5/server/instance/data/authors_job
/2011-11-15/21.33.28.795/report.txt
 XML report  /Sybase/DA-15_5/server/instance/data/authors_job
/2011-11-15/21.33.28.795/report.xml

(0 rows affected)
```

Note the path to the report files specified at the end of the output here.

This is an excerpt from the text report file:

```
source venus:5000/pubs2.dbo.authors
starttime 2011-11-15 21:33:00
endtime 2011-11-15 21:33:00

target pluto:5000/pubs2.dbo.authors
starttime 2011-11-15 21:33:00
endtime 2011-11-15 21:33:00
missing 0 orphaned 0 inconsistent 0
```

Note: A number of server configuration parameters may impact job performance. Use **config** to modify the default values for the configuration parameters.

Distributed Deployment

Sybase recommends a distributed deployment when there is high network latency between the DA server and the database servers, when many concurrent comparisons are required, or when performance requirements are more important than ease of deployment and maintenance.

Before You Begin

This example uses a single Data Assurance (DA) server and two remote DA agents. The local DA agent is not used.

Table 2. Deployment Summary

Component Name	Machine Name	Port Numbers
DA server	mars	• 4500 – RMI • 4501 – TDS • 4503 – DASD
DA agent	venus	• 4510 – RMI • 4511 – TDS • 4512 – DTS
DA agent	pluto	• 4510 – RMI • 4511 – TDS • 4512 – DTS
Adaptive Server Enterprise	venus	5000 – server
Adaptive Server Enterprise	pluto	5000 – server

1. Start your DA server instance:
   ```
   $SYBASE/DA-15_5/server/instance/RUN_instance_64.sh
   ```
   Where $SYBASE is the directory in which you installed the Data Assurance Option, *instance* is the name of your DA server instance, and `RUN_instance_64.sh` is the start-up script.

   Note: On Windows, the start-up script file is named `RUN_instance_32.bat` or `RUN_instance_64.bat`, where *instance* is your DA server instance name. On UNIX or Linux platforms, the file is named `RUN_instance_64.sh`.
2. Start your DA agent instances on the machines named venus and pluto:
   ```
   $SYBASE/DA-15_5/agent/instance/RUN_instance_64.sh
   ```

Where $SYBASE is the directory in which you installed the Data Assurance agent, *instance* is the name of your DA agent instance, and `RUN_instance_64.sh` is the start-up script.

Note: On Windows, the start-up script file is named `RUN_instance_32.bat` or `RUN_instance_64.bat`, where *instance* is your DA server instance name. On UNIX or Linux platforms, the file is named `RUN_instance_64.sh`.

3. From **isql**, log in to DA server as an administrator:

```
$SYBASE/OCS-15_0/bin/isql -S mars:4501 -U da_admin -P password -w 250
```

Note: 4501 is the default TDS port number for DA server. The TDS port is required when the command line tool connects to the DA server using **isql**.

You can also log in to your DA agent instances in the same way. For example:

```
$SYBASE/OCS-15_0/bin/isql -S venus:4511 -U da_admin -P password -w 250
```

4. Create DA agent profiles to connect to the DA server:

```
create agent agent_venus
    set host=venus
    and set port=4510
    and set user=da_admin
    and set password=password
go
```

```
create agent agent_pluto
    set host=pluto
    and set port=4510
    and set user=da_admin
    and set password=password
go
```

5. View the newly created DA agents:

```
show agent
go
```

6. Test connection settings for the DA agents:

```
test agent agent_venus
go

test agent agent_pluto
go
```

7. Create database connections for the new DA agents:

```
create connection conn_venus
    set agent=agent_venus
    and set host=venus
    and set port=5000
    and set database=pubs2
    and set user=sa
    and set password=''
go
```

```
create connection conn_pluto
    set agent=agent_pluto
```

```
    and set host=pluto
    and set port=5000
    and set database=pubs2
    and set user=sa
    and set password=''
go
```

Note: In this example, *agent_venus* connects to the Adaptive Server database installed on *venus*, and *agent_pluto* connects to the Adaptive Server installed on *pluto*.

8. View the newly created database connections:

```
show connection
go
```

9. Continue from step 5 in the previous example for a single-server deployment.

Database Table Reconciliation

DA can automatically reconcile differences between the source and target databases, or create a SQL script that enables a database administrator to manually reconcile the target database. You can configure DA server to do both tasks simultaneously.

This example uses a single DA server with the local embedded agent, and shows you how to generate a script to reconcile a target table that differs from the source table with one missing row, one inconsistent row, and one orphaned row.

Table 3. Deployment Summary

Component Name	Machine Name	Port Numbers
DA server	mars	• 4500 – RMI • 4501 – TDS • 4503 – DASD
Adaptive Server Enterprise	venus	5000 – server
Adaptive Server Enterprise	pluto	5000 – server

1. Follow step 1 to step 5 of the single-server deployment example to start your DA server instance and connect to your databases.
2. Create a new compareset to map the entire source table:

```
create compareset authors_demo3
  with
    source conn_venus dbo authors s
    target conn_pluto dbo authors t
  map
    s.au_id = t.au_id set key=true
    and s.au_lname = t.au_lname
    and s.au_fname = t.au_fname
    and s.phone = t.phone
```

```
    and s.address = t.address
    and s.city = t.city
    and s.state = t.state
    and s.country = t.country
    and s.postalcode = t.postalcode
go
```

Warning! DA server reconciles only columns that are mapped in the compareset. Using a compareset that partially maps a table for reconciliation may lead to automatic reconciliation errors and defective reconciliation scripts.

3. Create a new job:

```
create job authors_job3
    add comparison cmp_authors3
      set compareset = authors_demo3
      and set create_col_log = true
      and set create_recon_script = true
go
```

Note: You can set the job comparison auto_reconcile option to true to automatically reconcile data differences.

4. Execute the new job:

```
run job authors_job3
go
(1 row affected)
```

5. Monitor the job:

```
monitor job authors_job3 cmp_authors3
go
```

```


COMPARISON    SUBMIT TIME          END TIME
------------  -------------------  -------------------
cmp_authors3  2012-03-30 10:31:36  2012-03-30 10:31:42

(0 rows affected)

RUN PHASE             TYPE SUMMARY                START TIME          END
TIME             COUNT READ M O I R
--- ----------------- ---- ---------------------- -------------------
------------------- ---- ---- -------
   PROGRESS ESTIMATE END
ERROR
   -------- ------------------
 1  COMPARE_ALL        S   conn_venus/dbo.authors 2012-03-30 10:31:39
2012-03-30 10:31:39 23   23
   100%
                      T   conn_pluto/dbo.authors 2012-03-30 10:31:39
2012-03-30 10:31:39 23   23  1 1 1
   100%
 2  VERIFY_DIFFERENCES S                           2012-03-30 10:31:40
2012-03-30 10:31:41 2
   100%
```

```
                    T                          2012-03-30 10:31:40 2012-03-30
10:31:41 2        1 1 1
   100%
```

6. Obtain the job history ID:

```
show history authors_job3
go

HISTORY ID   SUBMIT TIME             FINISH TIME
----------   -------------------    -------------------
1            2012-03-30 10:31:36    2012-03-30 10:31:42

(0 rows affected)
```

7. Use the history ID to view the job history:

```
show history authors_job3 1
go
```

```
COMPARISON  RUN PHASE                      TYPE SUMMARY
START TIME         END TIME
----------- --- --------------------------  ----
-------------------------  -----------------  ------------------
   COUNT READ M O I R ERROR
   ----- ---- ------- -----
cmp_authors3 1  COMPARE_ALL                S  venus:5000/pubs2.dbo.authors
2012-03-30 10:31:39 2012-03-30 10:31:39
    23  23
                                          T  pluto:5000/pubs2.dbo.authors
2012-03-30 10:31:39 2012-03-30 10:31:39
    23  23  1 1 1
            2  VERIFY_DIFFERENCES          S
2012-03-30 10:31:40 2012-03-30 10:31:41
         2
                                           T
2012-03-30 10:31:40 2012-03-30 10:31:41
        2   1 1 1
            3  CREATE_RECONCILIATION_SCRIPT T
2012-03-30 10:31:41 2012-03-30 10:31:42
        3

(0 rows affected)


 FILE         SERVER PATH
 ------------ ------------------------------------------------------
-----------------------------------------------
 Recon script C:\Sybase\DA-15_5\server\instance\data\authors_job3\
2012-03-30\10.31.36.099\cmp_authors3_T_recon.sql
 Text report  C:\Sybase\DA-15_5\server\instance\data\authors_job3\
2012-03-30\10.31.36.099\report.txt
 XML report   C:\Sybase\DA-15_5\server\instance\data\authors_job3\
2012-03-30\10.31.36.099\report.xml
```

```
(0 rows affected)
```

This is an excerpt from the text report file:

```
source venus:5000/pubs2.dbo.authors
starttime 2012-03-30 10:31:39
endtime 2012-03-30 10:31:39

target pluto:5000/pubs2.dbo.authors
starttime 2012-03-30 10:31:40
endtime 2012-03-30 10:31:41
missing 1 orphaned 1 inconsistent 1
```

```
diff   |au_id           |au_lname    |au_fname   |phone          |
address             |city
----   --------         ----------   ---------   -------------
---------------     -----------
| state   |country  |postalcode
-------   --------  -----------
      |172-32-1176    |Roberts    |Alex      |408 496-7223   |10932 Bigge
Rd.     |Menlo Park
|CA       |USA      |94025
      |172-32-1176    |White      |Johnson   |408 496-7223   |10932 Bigge
Rd.     |Menlo Park
|CA       |USA      |94025
|                       |^^^^^^^     |^^^^^^^    |
|                    |
|         |
O     |213-46-8915    |Green      |Marjorie   |415 986-7020   |309 63rd
St. #411   |Oakland
|CA       |USA      |94618
M     |321-78-9087    |Jones      |Steve     |412 555-6434   |48 Barnaby
Close    |Walnut Creek
|CA       |USA      |94592

reconciliation script
    starttime 2012-03-30 10:31:41
    endtime 2012-03-30 10:31:42
    reconciled 3
(0 rows affected)
```

8. Execute the reconciliation script against the target database:

```
C:\>isql -S pluto:5000 -U sa -i "C:\Sybase\DA-15_5\server\myserver\data\
authors_job3\2012-03-30\10.31.36.099\cmp_authors3_T_recon.sql"

Password:
(1 row affected)
(1 row affected)
(1 row affected)
```

An example of the reconciliation script:

```
--
```

```
-- Replication Server Data Assurance Option/15.7.1/DA Server/P/generic/
generic/da157x/121/VM: Sun Microsystems Inc. 1.6.0_24/OPT/Tue 24 Apr
2012 09:24:31 GMT
-- Reconciliation Script (Auto-generated); fixes 3 difference(s).
-- Missing: 1 (insert)
-- Inconsistent: 1 (update)
-- Orphaned: 1 (delete)
--
-- Date Created: 2012-03-30 10:31:42
-- File encoding: UTF-8
--
-- Source: dbo.authors on venus:5000/pubs2
-- Target: dbo.authors on pluto:5000/pubs2
--
use pubs2
go
--
-- Missing: 1 rows
--
begin tran
insert into dbo.authors
au_id,au_lname,au_fname,phone,address,city,state,country,postalcode)
values ('321-78-9087','Jones','Steve','412 555-6434','48 Barnaby
Close','Walnut Creek','CA','USA','94592     ')
commit tran
go
--
-- Inconsistent: 1 rows
--
begin tran
update dbo.authors set au_lname = 'Roberts', au_fname = 'Alex' where au_id
= '172-32-1176'
commit tran
go
--
-- Orphaned: 1 rows
--
begin tran
delete from dbo.authors where au_id = '213-46-8915'
commit tran
go
```

Administrative Tasks

Create comparison jobs, import jobs from Replication Server, back up the Data Assurance System Database, and configure server parameters.

Creating a Job

Perform row comparison in DA server.

1. Either:
   - Use the local agent.
   - Choose the remote DA agent, which you created as part of your installation.
2. Create source and target database connection profiles.
3. Create a compareset.
4. Create a job.
5. Run the job.
6. Monitor your running job.
7. (Optional) Show job history.

See also

- *create agent* on page 34
- *create connection* on page 41
- *create compareset* on page 48
- *create job* on page 59
- *run job* on page 74
- *monitor job* on page 73
- *show history* on page 75

Creating a Schema Job

Compare database object schemas in DA server.

1. Either:
   - Use the local agent.
   - Choose the remote DA agent, which you created as part of your installation.

2. Create source and target database connection profiles.
3. Create a schema job.
4. Run the job.
5. (Optional) Show job history.

See also

- *create agent* on page 34
- *create connection* on page 41
- *create schema job* on page 68
- *run job* on page 74
- *show history* on page 75

Importing a Job from Replication Server

Import a job based on predefined table replication definitions and subscriptions from Replication Server System Database to DA server.

1. Either:
   - Use the local agent.
   - Choose the remote DA agent, which you created as part of your installation.
2. Create source and target database connection profiles.
3. Create a RSSD database connection profile.
4. Import a job from Replication Server.
5. (Optional) Alter the comparison options in the imported job.
6. (Optional) Alter or add a schedule to the imported job.
7. Run the job.
8. Monitor your running job.

See also

- *create agent* on page 34
- *create connection* on page 41
- *import job* on page 78
- *alter job* on page 53
- *run job* on page 74
- *monitor job* on page 73

Setting Server Configuration Parameters

Tune the server configuration parameters, which define how the DA server executes jobs, to improve system performance.

1. View default server configuration parameters.
2. Modify the default values for the appropriate server configuration parameter.

See also

- *config* on page 85

Backing Up and Restoring the DASD

Create a backup of the current DASD, then restore the DASD from the backup copy.

1. Create backup of the current DASD.
2. View the backup copy.
3. Restore the DASD from the backup copy.

   After restoring the DASD, the DA server shuts down.
4. Restart the DA server.

   See *Replication Server Data Assurance Option Installation Guide > Getting Started After Installing*.

See also

- *create backup* on page 83
- *show backup* on page 84
- *restore backup* on page 84

Deleting Data and Log Files

Delete job history and DASD backup.

- To delete job history, use one:
  - **drop history**
  - **truncate history**

  Note: drop history deletes history records by history ID.
- To delete backup (DASD copy), use one:

- **drop backup**
- **truncate backup**

See also

- *drop history* on page 72
- *truncate history* on page 77
- *drop backup* on page 83
- *truncate backup* on page 84

Data Assurance Server Command Reference

You can execute DA server commands using **isql** or the Sybase Control Center Data Assurance plug-in.

Note: You must have “da_admin” permission to execute all DA server commands.

Agent Commands

Commands for creating and managing agents in DA server.

alter agent

Changes the attributes of an existing agent. You can modify one or more attributes for an agent.

Syntax

```
alter agent agent_name
[set host [{to|=}] host]
[and set port [{to|=}] port]
[and set user [{to|=}] user]
[and set password [{to|=}] password]
[and set desc [{to|=}] description]
```

Parameters

- **agent_name –** the name of the agent.
- **host –** the host name of the machine on which the agent is installed.
- **port –** the port number of the machine on which the agent is installed.
- **user –** the administrator login name.
- **password –** the password associated with the login name.
- **description –** description of the agent. Use double quotes if you are using a reserved word or blank spaces.

Examples

- **Example 1 –** changes the myagent user name and password:

```
alter agent myagent
and set user=youruser
and set password=yourpwd
go
```

create agent

Creates an agent profile.

Syntax

```
create agent agent_name
set host [{to|=}] host
and set port [{to|=}] port
and set user [{to|=}] user
[and set password [{to|=}] password]
[and set desc [{to|=}] description]
```

Parameters

- **agent_name –** the name of the agent to be created.
- **host –** the host name of the machine on which the agent is installed.
- **port –** the port number of the machine on which the agent is installed.
- **user –** the login name of the Replication Server Data Assurance Option administrator for accessing the agent.
- **password –** (Optional) the password for the administration user login name to connect to the agent and the database.
- **description –** (Optional) description of the agent. Use double quotes if you are using a reserved word or blank spaces.

Examples

- **Example 1 –** creates a new agent:

```
create agent myagent
set host=myhost
and set port=1111
and set user=myuser
and set password=mypwd
go
```

See also

- *create connection* on page 41
- *alter connection* on page 40
- *test agent* on page 39

depend agent

Shows a list of connection names that depend on the named agent.

Syntax

```
depend agent agent_name
```

Parameters

- **agent_name –** the name of the agent.

Examples

- **Example 1 –** shows connection dependency for “myagent”:

```
depend agent myagent
go
```

The returned result is:

```
CONNECTION
----------
connection1
connection2
```

See also

- *show connection* on page 44

drop agent

Deletes an existing agent.

Syntax

drop agent *agent_name*

Parameters

- **agent_name –** the name of the agent to be dropped.

Examples

- **Example 1 –** removes “myagent” from the system:

```
drop agent myagent
go
```

show agent

Shows details of one, or all, agent.

Syntax

show agent [*agent_name*]

Parameters

- **agent_name –** (Optional) the name of the agent for which to show details. If this parameter is not provided, the details of all the agents are provided.

Examples

- **Example 1 –** shows information about all agents:

```
show agent
go
```

The returned result is:

```
NAME         HOST      PORT USER      DESCRIPTION
------------ --------- ---- --------- --------------
localagent   localhost 0    localuser
ragent       myuser    4510 da_admin  remote agent 1
```

show agent connection

Shows the database connections for an agent.

Syntax

show agent connection *agent_name*

Parameters

- **agent_name –** the name of the agent.

Examples

- **Example 1 –** shows the database connections for “localagent”:

```
show agent connection localagent
go
```

The returned result is:

```
NAME          TYPE  CONNECTED
------------- ----- ---------
conn_soka3    ASE   0
conn_soka2    ASE   0
```

show agent dts

Shows the data transfer stream (DTS) information that is running on a specified agent.

Syntax

show agent dts *agent_name*

Parameters

- **agent_name –** the name of the agent.

Examples

- **Example 1 –** shows all the DTS information for “agent1”:

```
show agent dts agent1
go
```

The returned result is:

```
SERVER              TASK ID CONNECTION OBJECT       STAGE OBJ PROCESSED
PREDICATE SQL PROCESSED
------------------- ------- ---------- -----------  ----- -------------
--------- --- ---------
myserver:4500@soka2  4      conn1      dbo.da1_10m        0
0             1
myserver:4500@soka2  2      conn2      dbo.da1_10m        0
0             1
myserver:4500@soka2  6      conn3      dbo.da1_10m        0
0             1

(0 rows affected)

 TASK ID ESTIMATE COUNT QUEUE TAKEN    ESTIMATE SECONDS LEFT
 ------- -------------- ----- -------- --------------------
 4       10000000       0     10000000 0
 2       10000000       0     10000000 0
 6       10000000       0     10000000 0
```

show agent jvm

Shows some of the important Java Virtual Machine (JVM) details of a remote DA agent.

Syntax

show agent jvm *agent_name*

Parameters

- **agent_name –** the name of the remote agent.

Examples

- **Example 1 –** shows JVM details for “myagent”:

```
show agent jvm myagent
go
```

The returned result is:

```
JVM NAME                    JVM INFO   JVM VENDOR
         JVM VERSION
------------------------- ---------- -----------
         -------------------------
Java HotSpot(TM) Server VM mixed mode Sun Microsystems Inc.
         Java 1.6.0_24, VM 19.1-b02

(0 rows affected)

JVM TOTAL MEM JVM FREE MEM JVM MAX MEM
------------- ------------ -----------
31.8 MB       27.1  MB     455.1 MB
```

show agent system

Shows some of the important system properties of a remote DA agent.

Syntax

show agent system *agent_name*

Parameters

- **agent_name –** the name of the agent.

Examples

- **Example 1 –** shows system details for “myagent”:

  ```
  show agent system myagent
  go
  ```

  The returned result is:

```
NAME      HOST         LOCALE  TIME ZONE            DATE       TIME
-------   -----------  ------  -------------------- ---------- --------
myagent   10.65.0.111  en_GB   Greenwich Mean Time  2011-06-10 16:05:44

OS NAME     OS VERSION   OS ARCH   OS LOAD AVG
----------  ----------   -------   -----------
Windows XP  5.1          x86       14.897%
```

show agent task

Shows the task information for an agent.

Syntax

show agent task *agent_name*

Parameters

- **agent_name –** the name of the agent.

Examples

- **Example 1 –** shows all tasks for “localagent”:

```
show agent task localagent
go
```

The returned result is:

```
TASK ID CONNECTION OBJECT      PREDICATE SQL PROCESSED
------- ---------- ----------- --------- --- ---------
3       conn_soka2 dbo.da1_10m
```

test agent

Validates whether or not an existing agent is available. If the agent is available, establishes and authenticates a connection to it.

Syntax

test agent *agent_name*

Parameters

- **agent_name –** the name of the agent to be tested.

Examples

- **Example 1 –** tests the agent “MyAgent”:

```
test agent MyAgent
go
```

The returned result is:

```
RESULT
----------
Succeeded
```

Connection Profile Commands

Commands for creating and managing source and target database connection profiles.

alter connection

Changes the attributes of an existing connection profile.

Syntax

```
alter connection connection_name
[set agent [{to|=}]agent_name]
[and set host  [{to|=}] host]
[and set port [{to|=}] port]
[and set database [{to|=}] database_name]
[and set user [{to|=}] username]
[and set password [{to|=}] password]
[and set desc [{to|=}] description]
```

To alter node information for a cluster connection:

```
alter connection connection_name
with node
set host [{to|=}] host and set port [{to|=}] port
[and node set host [{to|=}] host and set port [{to|=}] port] ...
```

To alter information properties:

```
alter connection connection_name
with properties
set property_name [{to|=}] property_value
[and set property_name [{to|=}] property_value ]…]
```

To drop information about properties:

```
alter connection connection_name
with properties drop property_name | ALL
```

Parameters

- **connection_name –** the name of the connection to be altered.
- **agent_name –** the name of the agent that establishes the connection to the database.
- **database_name –** the name of the target database.
- **host –** the host name of the machine on which the target database is installed.
- **port –** the port number of the machine on which the target database is installed.
- **user –** the database login name. The user must have select permission. To automatically reconcile inconsistent, missing, and orphaned rows in the target database, the user must have the update, insert, and delete permissions.
- **password –** (Optional) the password for the user login name.

  Note: If your Adaptive Server instance has a system administrator (sa) login with a null password, you must also state a null password in DA server, either by setting a blank password or by not setting a password at all.
- **description –** description of the agent. Use double quotes if you are using a reserved word or blank spaces.

- **property_name –** the connection properties for the Sybase JDBC driver, jConnect™ for JDBC™, which establishes the connection to the database.

  See *jConnect for JDBC 7.0 > jConnect for JDBC 7.0 Programmers Reference > Programming Information > Establishing a Connection > Connection Properties*.
- **property_value –** specifies the value of the property to be set.

Examples

- **Example 1 –** resets the user name and password for connection "MyConnPDB":

```
alter connection MyConnPDB
set user=myuser2
and set password=mypwd2
go
```

  The returned result is:

```
Connection "MyConnPDB" was altered successfully
```

create connection

Creates a database connection profile, which includes the agent and the JDBC parameters.

Syntax

```
create connection connection_name
set type [{to|=}] ASE | ASE_CLUSTER | RSSD
and set agent [{to|=}]agent_name
and set host  [{to|=}] host
and set port [{to|=}] port
and set database [{to|=}] database_name
and set user [{to|=}] user
[and set password [{to|=}] password]
[and set desc [{to|=}] description]
[with node set host [{to|=}] host and set port [{to|=}] port
[and node set host [{to|=}] host
and set port [{to|=}] port]…]
[with properties set property_name [{to|=}] property_value
[and set property_name [{to|=}] property_value ]…]
```

Parameters

- **connection_name –** the name of the connection to be created.
- **type –** (Optional) the connection type supported by Replication Server Data Assurance Option: Adaptive Server Enterprise (ASE), ASE_CLUSTER, and Replication Server System Database (RSSD). If type is not provided, but the node parameter is provided; the value defaults to ASE_CLUSTER; otherwise, it will default to ASE.

  You cannot set the host name and port number for an ASE_CLUSTER connection; you must set these values inside a node definition. **agent_name** is invalid, if connection type is

Replication Server System Database (RSSD). An RSSD connection cannot define an agent.

- **agent_name –** the name of the agent that establishes the connection to the database.
- **database_name –** the name of the target database.
- **host –** the host name of the machine on which the target database is installed.
- **port –** the port number of the machine on which the target database is installed.
- **user –** the database login name. The user must have select permission. To automatically reconcile inconsistent, missing, and orphaned rows in the target database, the user must have the update, insert, and delete permissions.
- **password –** (Optional) the password for the user login name.

  Note: If your Adaptive Server instance has a system administrator (sa) login with a null password, you must also state a null password in DA server, either by setting a blank password or by not setting a password at all.
- **description –** (Optional) description of the agent. Use double quotes if you are using a reserved word or blank spaces.
- **property_name –** the connection properties for the Sybase JDBC driver, jConnect for JDBC, which establishes the connection to the database.

  See *jConnect for JDBC 7.0 > jConnect for JDBC 7.0 Programmers Reference > Programming Information > Establishing a Connection > Connection Properties*.
- **property_value –** the value of the property to be set.

Examples

- **Example 1 –** creates an Adaptive Server connection named “MyConnPDB”:

```
create connection MyConnPDB
set type=ASE
and set agent=pdbAgt
and set database=mypdb
and set host=myhost
and set port=1111
and set user=myuser
and set password=mypwd
go
```

  The returned result is:

```
Connection “MyConnPDB” was created successfully
```

- **Example 2 –** creates an RSSD connection:

```
create connection MyRSSDConn
set type=RSSD
and set host=rshost
and set database=RSSD_database_name
and set port=2222
and set user=rsuser
and set password=rspwd
go
```

The returned result:

```
Connection “MyRSSDConn” was created successfully.
```

- **Example 3 –** creates an Adaptive Server connection that authenticates the user with a Kerberos authentication server:

```
create connection MyConnPDB
set type=ASE
and set agent=pdbAgt
and set database=mypdb
and set host=myhost
and set port=1111
and set user=myuser
and set password=mypwd
with properties
set REQUEST_KERBEROS_SESSION=true
and set SERVICE_PRINCIPAL_NAME=myserver
go
```

The returned result:

```
Connection “MyConnPDB” was created successfully.
```

- **Example 4 –** creates an Adaptive Server connection that encrypts the login password:

```
create connection MyEncryptedPasswordConn
set type=ASE
and set agent=pdbAgt
and set database=mypdb
and set host=myhost
and set port=4901
and set user=myuser
and set password=mypwd
with properties
set ENCRYPT_PASSWORD=true
and set JCE_PROVIDER_CLASS="com.certicom.ecc.jcae.Certicom"
go
```

See also

- *create compareset* on page 48
- *create schema job* on page 68
- *show agent connection* on page 36
- *test connection* on page 45

depend connection

Shows a list compareset names and a list of schema job comparisons that depend on the named connection. The list also indicates whether it is the source or target database using the named connection.

Syntax

depend connection *connection_name*

Parameters

- **connection_name –** the name of the connection.

Examples

- **Example 1 –** shows compareset dependency for “MyConnPDB”:

```
depend connection MyConnPDB
go
```

The returned result is:

```
COMPARESET      TYPE
----------      ----
compareset1     S
compareset2     T

SCHEMA JOB   COMPARISON    TYPE
-----------  ----------    ----
schema_job2   comparison2  S
schema_job3   comparison2  T
```

drop connection

Deletes an existing connection profile.

Syntax

drop connection *connection_name*

Parameters

- **connection_name –** the name of the connection to be dropped.

Examples

- **Example 1 –** drops the connection “MyConnPDB”:

```
drop connection MyConnPDB
go
```

The returned result is:

```
Connection “MyConnPDB” was dropped successfully
```

show connection

Shows zero or more existing connection profiles, which include the agent and the JDBC parameters.

Syntax

show connection [connection_name]

Parameters

- **connection_name –** (Optional) the name of the connection to be shown. If you do not provide this parameter, all connections are shown.

Examples

- **Example 1 –** shows all existing database connection profiles:

```
show connection
go
```

The returned result is:

```
NAME  TYPE AGENT      HOST  PORT DATABASE USER DESCRIPTION
----- ---- ---------- ----- ---- -------- ---- ------------------
pdb   ASE  localagent users 5010 pdb      sa   primary database
rdb   ASE  localagent users 5010 rdb      sa   replicate database

(0 rows affected)
```

test connection

Establishes a connection to the database, authenticates, and performs a simple query.

Syntax

test connection *connection_name*

Parameters

- **connection_name –** the name of the connection to be tested.

Examples

- **Example 1 –** tests the connection “MyConnPDB”:

```
test connection MyConnPDB
go
```

The returned result is:

```
RESULT
----------
Succeeded
(0 rows affected)
```

Compareset Commands

Commands for creating and managing comparesets.

alter compareset

Changes the attributes of an existing compareset. **alter compareset** fails if the compareset you are modifying is being used in an executing job.

Syntax

To alter description:

```
alter compareset compareset_name [force]
set desc [{to|=}] description
```

To drop a target table:

```
alter compareset compareset_name [force]
drop target target_connection_name owner_name
target_table_name
[and target_connection_name owner_name
target_table_name …]
```

To add a target connection:

```
alter compareset compareset_name  [force]
    add target target_connection_name owner_name
target_table_name target_alias [where “constraint”]
[and target target_connection_name owner_name
target_table_name target_alias2 where “constraint”]...]
map
    source.column_name = target_alias.column_name
[ = target_alias2.column_name ...],
[and source.column_name = target_alias2.column_name
[ = target_alias2.column_name ...]…]
```

To drop a column mapping:

```
alter compareset compareset_name [force]
drop map source_column_name [and source_column_name…]
```

To add a column mapping:

```
alter compareset compareset_name  [force]
    with target target_connection_name owner_name
target_table_name target_alias
[and target target_connection_name owner_name
target_table_name target_alias2]]
add map
    source.column_name = {target1}target_alias.column_name
[ = {target2}target_alias2.column_name] [set key=true],
[and source.column_name = {target1}target_alias2.column_name
[ = {target2}target_alias2.column_name] [set key=true]…]
```

To alter a **where constraint**:

```
alter compareset compareset_name  [force]
[alter source where constraint]
[alter target target_connection_name owner_name
target_table_name where constraint]
```

Note: You cannot modify the source **where constraint** and the target **where constraint** at the same time.

To drop a **where constraint**:

```
alter compareset compareset_name  [force]
[alter source where “”]
[alter target target_connection_name owner_name
target_table_name where “”]
```

To alter key columns:

```
alter compareset compareset_name  [force]
alter map
source.column_name set key {false | true}
[source.column_name set key {false | true}...]
```

Parameters

- **compareset_name –** the name of the compareset.
- **description –** description of the agent. Use double quotes if you are using a reserved word or blank spaces.
- **source_connection_name –** the name of the source connection.
- **target_connection_name –** the name of the target connection.
- **owner_name –** the name of the owner for the source or target tables.
- **source_table_name –** the name of the source table.
- **target_table_name –** the name of the target table.
- **source_alias –** the alias that refers to source connection, owner, and table.
- **target_alias –** the alias that refers to target connection, owner, and table.
- **column_name –** the name of the column.
- **force –** alters a compareset that points to one or more comparisons. You must use **force** to modify such comparisons.

Examples

- **Example 1 –** drops the target connection “conn_bak1” from the existing compareset “cust_orders”:

  ```
  alter compareset cust_orders
  drop target conn_bak1 cust_owner cust_orders
  go
  ```

  The returned result is:

  ```
  Compareset “cust_orders” was altered successfully.
  ```

- **Example 2 –** drops some column mappings from the existing compareset “cust_orders”:

  ```
  alter compareset cust_orders
  drop map
  id and cust_id
  go
  ```

The returned result is:

```
Compareset “cust_orders” was altered successfully.
```

Usage

- If you do not redefine the target alias, you can use the keywords “target1” and “target2” to refer to the target tables; the sequence must be consistent with the definition used when you created the compareset.
- When using the **add map** clause in **alter compareset**, you must include all the target connections.

create compareset

Creates a compareset, which includes the database connection profile, and the source and target tables to be compared.

Syntax

```
create compareset compareset_name
[set desc [{to|=}] description]
with source source_connection_name  owner_name
source_table_name source_alias [where “constraint”]
target target_connection_name owner_name
target_table_name target_alias [where “constraint”]
[and target target_connection_name owner_name
target_table_name target_alias [where “constraint”]...]
map {all [set strict_column=true | false]
[set strict_type=true | false]
[set keep_computed=true | false]
[set keep_encrypted=true | false]|
source_table_alias.column_name = target_table_alias.column_name
[ = target_table_alias.column_name ...]
[set key=true ],
[and source_table_alias.column_name =
target_table_alias.column_name [ =
target_table_alias.column_name ...]
[set key=true]…] }
```

Parameters

- **compareset_name –** the name of the compareset to be created.
- **description –** (Optional) description of the agent. Use double quotes if you are using a reserved word or blank spaces.
- **source_connection_name –** the name of the source connection.
- **target_connection_name –** the name of the target connection.
- **owner_name –** the name of the owner for the source or target tables.
- **source_table_name –** the name of the source table.
- **target_table_name –** the name of the target table.

- **source_alias –** the alias that refers to source connection, owner, and table.
- **target_alias –** the alias that refers to target connection, owner and table.
- **column_name –** the name of the column.
- **key –** specifies whether the column is a primary key or an unique key.

Note: You must define at least one key column in your compareset. When you use the **map all** statement, both the mapping and the key property come from the database.

- **strict_column –** if set to true, this parameter throws an exception if one or more source columns do not exist in target tables. By default, **strict_column** is false.
- **strict_type –** if set to true, this parameter checks the type name, scale, and precision. By default, **strict_type** is false.

Note: The **strict_type**, **strict_column**, **keep_computed**, and **keep_encrypted** are used only when you use the **map all** statement, which requires the source and target databases to be online; while for explicit mappings, databases need not be online.

Examples

- **Example 1 –** creates a new compareset named “cust_orders”:

```
create compareset cust_orders
with source conn_prod cust_owner cust_orders t1
where “id>100”
target conn_bak1 cust_owner cust_orders t2
where “id>100”
and target conn_bak2 cust_owner cust_orders t3
where “id>100”
map
t1.id = t2.id = t3.id set key=true
and t1.cust_id = t2.cust_id = t3.cust_id
and t1.sku = t2.sku = t3.sku
and t1.date= t2.date = t3.date
go
```

  The returned result is:

```
Compareset “cust_orders” was created successfully
```

- **Example 2 –** creates a new compareset named “cust_orders” and automatically maps all the target and source columns:

```
create compareset cust_orders
with source conn_prod cust_owner cust_orders t1
where “id>100”
target conn_bak1 cust_owner cust_orders t2
where “id>100”
and target conn_bak2 cust_owner cust_orders t3
where “id>100”
map all
go
```

  The returned result is:

```
Compareset “cust_orders” was created successfully
```

Usage

- All targets must have same number of map columns.
- At least one key column is required. Multiple key columns are allowed.

See also

- *create connection* on page 41
- *create job* on page 59

Limitations

The **map all** clause of the **create compareset** has several limitations.

- Your source and target database tables must have at least one primary key column or one identity column defined when using the **map all** statement in **create compareset**. For example, **map all** statement works if you create your table with a primary key or identity column:

  ```
  CREATE TABLE orders (
      order_num INTEGER NOT NULL PRIMARY KEY,
      date_ordered DATE,
      name CHAR(80)
    )
  ```

  ```
  CREATE TABLE orders (
      order_num INTEGER IDENTITY,
      date_ordered DATE,
      name CHAR(80)
   )
  ```

- If your table does not have a primary key or identity column, enforcing a primary key using **sp_primarykey** stored procedure does not enable **map all** to work.

  For example, map all cannot map a table defined like this:

  ```
  CREATE TABLE orders (
      order_num INTEGER NOT NULL,
      date_ordered DATE,
      name CHAR(80)
    )
    sp_primarykey orders, order_num
  go
    create unique clustered index ordernumidx on orders(order_num)
  go
  ```

  The **map all** clause cannot be used here because a table with a primary key defined using the **sp_primarykey** system procedure still lacks a primary key integrity constraint. In such a case, you can alter the table to enforce a primary key integrity constraint. For example:

  ```
  drop index orders.order_num
  go
  alter table orders
  ```

```
add constraint order_num_pk
primary key (order_num)
go
```

You can verify that a table has a primary-key constraint by using the **sp_helpconstraint** system procedure. See *Adaptive Server Enterprise Transact-SQL Users Guide > Creating Databases and Tables > Defining Integrity Constraints for Tables > Specifying Unique and Primary Key Constraints*.

- When using the **map all** statement in **create compareset**, the source and target table columns must have identical names.

depend compareset

Shows a list of (non-schema) job comparisons that have a dependency on the named compareset.

Syntax

depend compareset *compareset_name*

Parameters

- **compareset_name –** the name of the compareset.

Examples

- **Example 1 –** shows job comparison dependency for “cust_orders”:

```
depend compareset cust_orders
go
```

The returned result is:

```
JOB COMPARISON
--------------
job4/comparison3
job5/comparison1
```

See also

- *show job* on page 65

drop compareset

Deletes an existing compareset. **drop compareset** fails if the compareset is being used in an existing job.

Syntax

drop compareset *compareset_name*

Parameters

- **compareset_name –** the name of the compareset to be deleted.

Examples

- **Example 1 –** drops the compareset “cust_orders”:

```
drop compareset cust_orders
go
```

The returned result is:

```
Compareset “cust_orders” was dropped successfully
```

show compareset

Shows zero or more comparesets, which includes the database connection profile, the tables that are compared, and the column mappings.

Syntax

show compareset [*compareset_name*[columns]]

Parameters

- **compareset_name –** (Optional) the name of the compareset to be shown.

Examples

- **Example 1 –** shows the compareset “cust_orders” :

```
show compareset cust_orders
go
```

The returned result is:

```
TYPE CONNECTION OWNER TABLE        WHERE CONSTRAINT
---- ---------- ----- ------------ ----------------
S    conn_prod  dbo   cust_orders
T    conn_bak1  dbo   cust_orders
T    conn_bak2  dbo   cust_orders

(0 rows affected)
```

- **Example 2 –** shows all the comparesets :

```
show compareset
go
```

The returned result is:

```
NAME            DESCRIPTION
----            -----------
```

```
cust            The customer tables.
cust_orders     The customer orders tables.
```

- **Example 3 –** shows the column mappings in compareset “cust_orders”’:

```
show compareset cust_orders columns
go
```

The returned result is:

```
TYPE  CONN       TABLE        MAP_ID  COL_NAME     COL_TYPE      P_KEY
----  ---------  -----------  ------  ----------   -----------   -----
 S    conn_prod  cust_orders       1    id         numeric(11)     Y
 T    conn_bak1  cust_orders       1    id         numeric(11)     Y
 T    conn_bak2  cust_orders       1    id         numeric(11)     Y
 S    conn_prod  cust_orders       2    cust_id    numeric(9)      N
 T    conn_bak1  cust_orders       2    cust_id    numeric(9)      N
 T    conn_bak2  cust_orders       2    cust_id    numeric(9)      N
 S    conn_prod  cust_orders       3    sku        varchar(50)     N
 T    conn_bak1  cust_orders       3    sku        varchar(50)     N
 T    conn_bak2  cust_orders       3    sku        varchar(50)     N
 S    conn_prod  cust_orders       4    date       datetime        N
 T    conn_bak1  cust_orders       4    date       datetime        N
 T    conn_bak2  cust_orders       4    date       datetime        N
```

Row Comparison Job Commands

Commands for creating and managing row comparison jobs.

alter job

Changes the attributes of an existing job.

Syntax

To drop a job comparison:

```
alter job job_name
    drop comparison comparison_name [and comparison_name2 […]]
```

To add a job comparison:

```
alter job job_name
add comparison comparison_name
set COMPARESET =compareset_name
[and set ABORT_DIFF_MAX [{to|=}] number_of_differences
[and set ABORT_DIFF_ROW_COUNT [{to|=}] {true|false}
[and set AUTO_RECONCILE [{to|=}] {true|false}
[and set COMPARE_MODE [{to|=}] {row_compare | key_compare |
row_checksum | table_checksum | row_count}
[and set COMPRESS_DATA_TRANSFER [{to|=}] {true|false}
[and set CREATE_COL_LOG [{to|=}] {true|false}
[and set CREATE_RECON_SCRIPT [{to|=}] {true|false}
[and set DESC [{to|=}] description
[and set ENABLE_ROW_COUNT [{to|=}] {true|false}
```

```
[and set EXTERNAL_SORT [{to|=}] {true|false}
[and set HASH_TYPE [{to|=}] {database_hash | agent_hash}
[and set PRIORITY [{to|=}] {highest | high | normal | low}
[and set RETRY_DELAY_SEC [{to|=}] number_delay_second
[and set RETRY_DIFF [{to|=}] {never | wait_and_retry }
[and set RETRY_MAX [{to|=}] number_of_retries
]]]]]]]]]]]]]]]
[with column option
set column_name [{to|=}] {literal | column_hash | row_hash}
[set column_name [{to|=}] {literal | column_hash | row_hash}]
 […]]
 [and comparison comparison_name2
…]
```

To alter the job comparisons:

```
alter job job_name
alter comparison comparison_name
[and set ABORT_DIFF_MAX [{to|=}] number_of_differences
[and set ABORT_DIFF_ROW_COUNT [{to|=}] {true|false}
[and set AUTO_RECONCILE [{to|=}] {true|false}
[and set COMPARE_MODE [{to|=}] {row_compare | key_compare |
row_checksum | table_checksum | row_count}
[set COMPARESET =compareset_name
[and set COMPRESS_DATA_TRANSFER [{to|=}] {true|false}
[and set CREATE_COL_LOG [{to|=}] {true|false}
[and set CREATE_RECON_SCRIPT [{to|=}] {true|false}
[and set DESC [{to|=}] description
[and set ENABLE_ROW_COUNT  [{to|=}] {true|false}
[and set EXTERNAL_SORT [{to|=}] {true|false}
[and set HASH_TYPE [{to|=}] {database_hash | agent_hash}
[and set PRIORITY [{to|=}] {highest | high | normal | low}
[and set RETRY_DELAY_SEC [{to|=}] number_delay_second
[and set RETRY_DIFF [{to|=}] {never | wait_and_retry }
[and set RETRY_MAX [{to|=}] number_of_retries
]]]]]]]]]]]]]]]
```

To alter the column options in a job comparison:

```
alter job job_name
alter comparison comparison_name
with column option
set column_name [{to|=}] {literal | column_hash | row_hash}
[set column_name [{to|=}] {literal | column_hash | row_hash}]
 […]
```

To change the job options:

```
alter job job_name
[[set MAX_CONCURRENT_COMPARISONS [{to|=}]
number_of_max_concurrent_comparisons
[and set DESC [{to|=}] description]]
```

To change all the comparison options:

```
alter job job_name
alter  comparison option
set parameter [{to|=}] value
```

To enable a comparison:

```
alter job job_name enable comparison comparison_name
[ and comparison_name2[…]]
```

To disable a comparison:

```
alter job job_name disable comparison comparison_name
[ and comparison_name2[…]]
```

To add or alter the scheduling options:

```
alter job job_name
{add | alter} schedule schedule_name
[set TYPE [{to|=}] {once | cron | every_day | every_week |
every_month}
[and set EVERY [{to|=}] n
[and set DATE [{to|=}] date_value
[and set TIME [{to|=}] time_value
[and set KEEP [{to|=}] keep_value
[and set KEEP_UNIT [{to|=}] {day | week | month | forever}
[and set CRON [{to|=}] cron_value
[and set DESC [{to|=}] description
]]]]]]]]
     [and schedule schedule_name2
[set TYPE [{to|=}] {once | cron | every_day | every_week |
every_month}
[and set EVERY [{to|=}] n
[and set DATE [{to|=}] date_value
[and set TIME [{to|=}] time_value
[and set KEEP [{to|=}] keep_value
[and set KEEP_UNIT [{to|=}] {day | week | month | forever}
[and set CRON [{to|=}] cron_value
[and set DESC [{to|=}] description
]]]]]]]]……]}
```

To drop the scheduling option:

```
alter job job_name
drop schedule schedule_name[and schedule_name2[……]]
```

Parameters

- **job_name –** the name of the job.
- **comparison_name –** the name of the comparison to be added to the job.
- **compareset_name –** the name of the compareset to be added into the comparison.
- **schedule_name –** the name of the schedule to be added.
- **max_concurrent_comparisons –** the number of the comparisons that can be run concurrently with a job.
- **description –** description of the agent. Use double quotes if you are using a reserved word or blank spaces.

Table 4. Comparison Options

Parameter	Value
ABORT_DIFF_MAX	Aborts row comparison if the difference count exceeds the specified value. Valid values: 1 to 9223372036854775807. Default value:1000.
ABORT_DIFF_ROW_COUNT	Determines whether to abort row comparison if table row counts do not match. Valid values: true or false. Default value: false.
AUTO_RECONCILE	Indicates whether to automatically apply the reconciliation script. Valid values: true or false. Default value: false. **Note:** To enable **AUTO_RECONCILE**, set **CREATE_COL_LOG** to true.
COMPARE_MODE	Specifies the row comparison mode. • `row_compare` – compares all table rows. • `key_compare` – compares the primary key columns. • `row_count` – compress row count. Default value: row_compare.
COMPRESS_DATA_TRANSFER	Compresses the row data between the agent and the server. Valid values: true or false. Default value: false.
CREATE_COL_LOG	Generates a column differences log, which lists all missing, orphaned, and inconsistent row values (keys and columns). Create a column log if you want to: • Generate a reconciliation script • Perform automatic reconciliation • Generate a detailed report Valid values: true or false. Default value: false.

Parameter	Value
CREATE_RECON_SCRIPT	Generates a reconciliation script. To use this parameter, you must also set **CREATE_COL_LOG** to true. Valid values: true or false. Default value: false.
ENABLE_ROW_COUNT	Determines whether or not to count source and target table rows before they are compared. DA server uses the row count to estimate the comparison progress and end time. **Note:** DA server counts rows if **COMPARE_MODE** is row_count. Use **ENABLE_ROW_COUNT** only if **COMPARE_MODE** is a value other than row_count. Valid values: true or false. Default value: true.
EXTERNAL_SORT	Sorts rows on the agent, thereby reducing the impact of processing the **ORDER BY** clause in the databases. Valid values: true or false. Default value: false.
HASH_TYPE	Specifies the hash type for the comparison. • `database_hash` – use the hash function provided by the database. • `agent_hash` – use the hash function provided by Replication Server Data Assurance Option. Default value: database_hash.
PRIORITY	Specifies the job comparison order in the comparison queue. Valid values are: • `highest` • `high` • `normal` • `low` Default value: normal.

Parameter	Value
RETRY_DIFF	Specifies the retry option. • `never` – no recompare. • `wait_and_retry` – run recompare based on **RETRY_MAX** and **RETRY_DELAY_SEC** settings. Default value: never.
RETRY_DELAY_SEC	Specifies the number of seconds delay for each recomparison. Valid values: 0 to 86400. Default value: 10.
RETRY_MAX	Specifies the total number of recomparison for rows that have differences resulting from a previous comparison. Valid values: 0 to 100. Default value: 3.

Table 5. Column Comparison Option

Column Option	Value
COMPARE_MODE	Specifies for how each column is compared. • column_hash – compares using column hash value. • row_hash – compares all columns with this option together with a whole hash value. • literal – compares using column literal value.

Table 6. Scheduling Options

Parameter	Value
date_value	Specifies a date in the scheduler.
time_value	Specifies a time in the scheduler.
keep_value	Specifies the number of keep units for which this schedule remains active.
cron_value	Specifies the cron option value in the scheduler.

Examples

- **Example 1 –** alters “myjob_1” to remove its comparison:

```
alter job myjob_1
drop comparison mycomparison_1
go
```

The returned result is:

```
Job “myjob_1” was altered successfully.
```

- **Example 2 –** alters “myjob_1” to add comparison:

```
alter job myjob_1
add comparison mycomparison_2
set compareset=mycompareset_2
and set priority = high
go
```

The returned result is:

```
Job “myjob_1” was altered successfully.
```

create job

Creates a new job from one or more comparesets, schedules, and comparison options.

Syntax

create job *job_name*
[set MAX_CONCURRENT_COMPARISONS [{to|=}]
number_of_max_concurrent_comparisons]
[and set DESC [{to|=}] *description*]
add comparison *comparison_name*
set COMPARESET=*compareset_name*
[and set ABORT_DIFF_MAX [{to|=}] *number_of_differences*
[and set ABORT_DIFF_ROW_COUNT [{to|=}] {true|false}
[and set AUTO_RECONCILE [{to|=}] {true|false}
[and set COMPARE_MODE [{to|=}] {row_compare | key_compare |
row_checksum | table_checksum | row_count}
[and set COMPRESS_DATA_TRANSFER [{to|=}] {true|false}
[and set CREATE_COL_LOG [{to|=}] {true|false}
[and set CREATE_RECON_SCRIPT [{to|=}] {true|false}
[and set DESC [{to|=}] *description*
[and set ENABLE_ROW_COUNT [{to|=}] {true|false}
[and set EXTERNAL_SORT [{to|=}] {true|false}
[and set HASH_TYPE [{to|=}] {database_hash | agent_hash}
[and set PRIORITY [{to|=}] {highest | high | normal | low}
[and set RETRY_DELAY_SEC [{to|=}] *number_delay_second*
[and set RETRY_DIFF [{to|=}] {never | wait_and_retry }
[and set RETRY_MAX [{to|=}] *number_of_retries*
]]]]]]]]]]]]]]]
[with column option
set column_name [{to|=}] {literal | column_hash | row_hash}
[set column_name [{to|=}] {literal | column_hash | row_hash}]
 […]]

```
[add schedule schedule_name
[set TYPE [{to|=}] {once | cron | every_day | every_week |
every_month}
[and set EVERY [{to|=}] n
[and set DATE [{to|=}] date_value
[and set TIME [{to|=}] time_value
[and set KEEP [{to|=}] keep_value
[and set KEEP_UNIT [{to|=}] {day | week | month | forever}
[and set CRON [{to|=}] cron_value
[and set DESC [{to|=}] description
]]]]]]]]]
```

To clone an existing job:

```
create job job_name with exist_job_name
```

Note: When you clone a job with schedules, the new job includes the cloned schedules but they will always be inactive.

Parameters

- **job_name –** the name of the job to be created.
- **exist_job_name –** the name of an existing job to be cloned.
- **comparison_name –** the name of the comparison to be added to the job.
- **compareset_name –** the name of the compareset to be added into the comparison.
- **schedule_name –** the name of the schedule to be added.
- **max_concurrent_comparisons –** (Optional) the number of the comparisons that can be run concurrently with a job. The default value is 5
- **description –** (Optional) description of the agent. Use double quotes if you are using a reserved word or blank spaces.

Table 7. Comparison Options

Parameter	Value
ABORT_DIFF_MAX	Aborts row comparison if the difference count exceeds the specified value. Valid values: 1 to 9223372036854775807. Default value:1000.
ABORT_DIFF_ROW_COUNT	Determines whether to abort row comparison if table row counts do not match. Valid values: true or false. Default value: false.

Parameter	Value
AUTO_RECONCILE	Indicates whether to automatically apply the reconciliation script. Valid values: true or false. Default value: false. **Note:** To enable **AUTO_RECONCILE**, set **CREATE_COL_LOG** to true.
COMPARE_MODE	Specifies the row comparison mode. • `row_compare` – compares all table rows. • `key_compare` – compares the primary key columns. • `row_count` – compress row count. Default value: row_compare.
COMPRESS_DATA_TRANSFER	Compresses the row data between the agent and the server. Valid values: true or false. Default value: false.
CREATE_COL_LOG	Generates a column differences log, which lists all missing, orphaned, and inconsistent row values (keys and columns). Create a column log if you want to: • Generate a reconciliation script • Perform automatic reconciliation • Generate a detailed report Valid values: true or false. Default value: false.
CREATE_RECON_SCRIPT	Generates a reconciliation script. To use this parameter, you must also set **CREATE_COL_LOG** to true. Valid values: true or false. Default value: false.

Parameter	Value
ENABLE_ROW_COUNT	Determines whether or not to count source and target table rows before they are compared. DA server uses the row count to estimate the comparison progress and end time. **Note:** DA server counts rows if **COMPARE_MODE** is row_count. Use **ENABLE_ROW_COUNT** only if **COMPARE_MODE** is a value other than row_count. Valid values: true or false. Default value: true.
EXTERNAL_SORT	Sorts rows on the agent, thereby reducing the impact of processing the **ORDER BY** clause in the databases. Valid values: true or false. Default value: false.
HASH_TYPE	Specifies the hash type for the comparison. • `database_hash` – use the hash function provided by the database. • `agent_hash` – use the hash function provided by Replication Server Data Assurance Option. Default value: database_hash.
PRIORITY	Specifies the job comparison order in the comparison queue. Valid values are: • `highest` • `high` • `normal` • `low` Default value: normal.
RETRY_DIFF	Specifies the retry option. • `never` – no recompare. • `wait_and_retry` – run recompare based on **RETRY_MAX** and **RETRY_DELAY_SEC** settings. Default value: never.

Parameter	Value
RETRY_DELAY_SEC	Specifies the number of seconds delay for each re-comparison. Valid values: 0 to 86400. Default value: 10.
RETRY_MAX	Specifies the total number of recomparison for rows that have differences resulting from a previous comparison. Valid values: 0 to 100. Default value: 3.

Table 8. Column Comparison Option

Column Option	Value
COMPARE_MODE	Specifies for how each column is compared. • column_hash – compares using column hash value. • row_hash – compares all columns with this option together with a whole hash value. • literal – compares using column literal value.

Table 9. Scheduling Options

Parameter	Value
date_value	Specifies a date in the scheduler.
time_value	Specifies a time in the scheduler.
keep_value	Specifies the number of keep units for which this schedule remains active.
cron_value	Specifies the cron option value in the scheduler.

Examples

- **Example 1 –** creates a new job named “myjob_1”:

```
create job myjob_1
set max_concurrent_comparisons = 3
add comparison mycomparison_1
set compareset=mycompareset_1
and set priority = high
with column option
and set a = literal
```

```
set b = hash
and comparison mycomparison_2
set compareset=mycomparset_2
and set priority = normal
with schedule myschedule_1
set type=every_day
and set every=2
and set time=10:00
and set keep=1
and set keep_unit=month
and set date=2011-05-05
go
```

The returned result is:

```
Job “myjob_1” was created successfully.
```

- **Example 2 –** clones “myjob_1” to a new job “myjob_2”:

```
create job myjob_2 with myjob_1
go
```

The returned result is:

```
Job “myjob_2” was created successfully.
```

Usage

- The names of the comparisons in a cloned job are generated automatically (if not explicitly specified). The rule is *job_name_cloned_comparison*_sequence_number, where sequence_number starts from one.
- When cloning jobs, you can redefine only the scheduling options. The comparison options are automatically imported from the existing job.

drop job

Deletes an existing job.

Syntax

```
drop job job_name
```

Parameters

- **job_name –** the name of the job to be deleted.

Examples

- **Example 1 –** drops “myjob_1”:

```
drop job myjob_1
go
```

The returned result is:

```
Job ‘myjob_1’ was dropped successfully.
```

show job

Shows zero or more existing jobs, each of which includes one or more comparisons.

Syntax

show job *[job_name [,comparison_name]]*

Parameters

- **job_name –** the name of the job to be shown .
- **comparison_name –** the name of the comparison to be shown.

Examples

- **Example 1 –** shows existing jobs, with their status:

```
show job
go
```

The returned result is:

```
Name       Active   Description
------     -------  -----------
job_1      true     my_job1
job_2      true     my_job2
job_3      true     my_job3
```

Schema Comparison Job Commands

Commands for creating schema comparison jobs.

alter schema job

Changes the attributes of an existing schema job.

Syntax

To add a comparison to a schema job:

alter schema job *sc_job_name*
add comparison *comparison_name*
With source *source_connection_name source_connection_name_alias*
target *target_connection_name target_connection_name_alias*
[and target *target2_connection_name target2_connection_name_alias*]…
[include all tables]
[map tables

```
source_connection_name_alias.source_object_schema.source_object_name=

target_connection_name_alias.target_object_schema.target_object_name[=

target2_connection_name_alias.target2_object_schema.target2_object_name]
…]
     [and
source_connection_name_alias.source_object2_schema.source_object2_name=

target_connection_name_alias.target_object2_schema.target_object2_name[=

target2_connection_name_alias.target2_object2_schema.target2_object2_nam
e]…]
     ]
          [exclude tables
     source_object_schema.source_object_name
     [and source_object2_schema.source_object2_name]…]
```

To drop a comparison from a schema job:

```
alter schema job sc_job_name
drop comparison comparison_name
```

To alter a schema comparison that alters its target connection:

```
alter schema job sc_job_name
alter comparison comparison_name
drop target target_connection_name
           [and target2_connection_name]…
```

To alter a schema comparison to add a target connection:

```
alter schema job sc_job_name
alter comparison comparison_name
        add target new_target_connection_name
new_target_connection_name_alias
           [and target new_target2_connection_name
new_target2_connection_name_alias]…
[map tables
source.source_object_schema.source_object_name=
new_target_connection_name_alias.new_target_object_schema.new_targe
t_object_name[=
new_target2_connection_name_alias.new_target2_object_schema.new_tar
get2_object_name]…
[and source.source_object2_schema.source_object2_name=
new_target_connection_name_alias.new_target_object2_schema.new_targ
et_object2_name[=
new_target2_connection_name_alias.new_target2_object2_schema.new_ta
rget2_object2_name]…]]
```

To alter a schema comparison to drop table mappings:

```
alter schema job sc_job_name
alter comparison comparison_name
drop map  tables
     source.source_object_schema.source_object_name
           [and
source_connection_name.source_object2_schema.source_object2_name]…
```

To alter a schema comparison that adds table mappings:

```
alter schema job sc_job_name
alter comparison comparison_name
add map tables
source_connection_name.source_object_schema.source_object_name=[
new_target_connection_name.new_target_object_schema.new_target_obje
ct_name[=
new_target_connection_name2.new_target_object_schema2.new_target_ob
ject_name2]…]]…
```

Note: The added map entry overrides the existing one, if the key of the new added map entry is included in the existing map.

To alter a schema comparison to add table exclusions:

```
alter schema job sc_job_name
alter comparison comparison_name
add map tables
source_connection_name.source_object_schema.source_object_name=[
new_target_connection_name.new_target_object_schema.new_target_obje
ct_name[=
new_target_connection_name2.new_target_object_schema2.new_target_ob
ject_name2]…]]…
```

To add an **all tables** clause to a schema comparison:

```
alter schema job sc_job_name
alter comparison comparison_name
add include all tables
```

To alter a schema comparison to remove the **include all tables** clause:

```
alter schema job sc_job_name
alter comparison comparison_name
drop include all tables
```

To alter schema comparison job options:

```
alter schema job sc_job_name
set max_concurrent_comparisons [{to|=}]
number_of_max_concurrent_comparisons
[and set desc [{to|=}] description]
```

Parameters

- **sc_job_name –** the name of the schema comparison job.
- **comparison_name –** the name of the schema comparison job.
- **source_connection_name –** the name of the source connection.
- **source_connection_name_alias –** the alias name of the source connection.
- **target_connection_name –** the name of the target connection.
- **target_connection_name_alias –** the alias name of the target connection.
- **source_object_schema –** the schema name of the source object.
- **source_object_name –** the name of the source object.

- **new_source_connection_name –** the new name of the source connection.
- **new_source_object_name –** the new name of the source object.
- **new_source_object_schema –** the new schema name of the source object.
- **target_object_schema –** the schema name of the target object.
- **target_object_name –** the name of the target object.
- **new_target_connection_name –** the new name of the target connection.
- **new_target_object_name –** the new name of the target object.
- **new_target_object_schema –** the new schema name of the target object.
- **max_concurrent_comparisons –** the number of maximum concurrent comparisons.
- **description –** description of the agent. Use double quotes if you are using a reserved word or blank spaces.

Examples

- **Example 1 –** disables the **include all tables** clause in "schema_job":

```
alter schema job schema_job
alter comparison sj_cmp
drop include all tables
go
```

  The returned result is:

```
Job "schema_job" was altered successfully.
```

- **Example 2 –** alters the job description for "schema_job":

```
alter schema job schema_job
set desc="my schema job"
go
```

  The returned result is:

```
Job "schema_job" was altered successfully.
```

create schema job

Creates a new schema job for comparing database object schemas.

Syntax

```
create schema job sc_job_name
set max_concurrent_comparisons = 100
[and set desc [{to|=}] description]
  add comparison comparison_name
       With source source_connection_name source_alias
   target target_connection_name target_alias
      [and target target2_connection_name target2_connection_name_alias]…
[include all tables]
         [map tables
     source_connection_name_alias.source_schema.source_object_name=
     target_connection_name_alias.target_schema.target_object_name[=
     target2_connection_name_alias.target2_schema.target2_object_name]…]
```

```
     [and
source_connection_name_alias.source_object2_schema.source_object2_name=
     target_connection_name_alias.target_schema.target_object2_name[=
    target2_connection_name_alias.target2_schema.target2_object2_name]…]
     ]
          [exclude tables
     source_schema.source_object_name
     [and source_schema.source_object2_name]…]
```

Parameters

- **sc_job_name –** the name of the schema comparison job.
- **comparison_name –** the name of the schema comparison.
- **max_concurrent_comparisons –** (Optional) the number of maximum concurrent comparisons. The default value is 5.
- **description –** (Optional) description of the agent. Use double quotes if you are using a reserved word or blank spaces.
- **source_connection_name –** the name of the source connection.
- **source_alias –** the alias name of the source connection.
- **target_connection_name –** the name of the target connection.
- **target_alias –** the alias name of the target connection.
- **source_schema –** the schema name of the source object.
- **source_object_name –** the name of the source object.
- **target_schema –** the schema name of the target object.
- **target_object_name –** the name of the target object.

Examples

- **Example 1 –** creates a schema comparison job “sc_job_test”:

```
create schema job sc_job_test
  set max_concurrebt_comparisons = 100
  add comparison cmp1
    with source s1_con  s1
           target t1_con t1
            and target t2_con t2
      include all tables
      map tables
            s1.tab_a = t1.tab_b
            and s1.tab_b = t2.tab_c
      exclude tables
            s1.tab_a2
```

The returned result is:

```
Schema job “sc_job_test” was created successfully.
```

Usage

- The **include all tables** clause specifies that all tables in the source database are in the schema comparison object lists, and use automatic name mapping between source database and target database to compare table schemas. The **exclude table** clause specifies the tables you want to exclude in the source database after you have set **include all tables** for a schema job.
- The **map tables** clause specifies the object mapping. A source object cannot be in the **map tables** and **exclude tables** simultaneously; the object mapping overrides the **map tables** clause. Object mapping for the current release is limited to tables.

drop schema job

Deletes an existing schema job.

Syntax

drop schema job *sc_job_name*

Parameters

- **sc_job_name –** the name of the schema comparison job to be dropped.

Examples

- **Example 1 –** deletes schema job "sc_job_test" :

```
drop schema job sc_job_test
go
```

The returned result is:

```
Schema job "sc_job_test" was dropped successfully.
```

show schema job

Shows zero or more existing schema comparison jobs, each of which consists of one or more comparisons.

Syntax

show schema job *[schema_job_name [schema_job_comparison_name]]*

Parameters

- **schema_job_name –** the name of the schema comparison job to be shown.
- **schema_job_comparison_name –** the name of the schema comparison to be shown.

Examples

- **Example 1 –** shows existing schema jobs, with their status:

```
show schema job
go
```

The returned result is:

```
Name         Active   Description
------      --------  ------------
scjob_1      true     my_schemajob1
scjob_2      true     my_schemajob2
scjob_3      true     my_schemajob3
```

Managing Job Commands

Commands for creating job execution and history.

abort job

Aborts a running job.

Syntax

```
abort job job_name
```

Parameters

- **job_name –** the name of the job to be aborted.

Examples

- **Example 1 –** aborts “myjob_1”:

```
abort job myjob_1
go
```

The returned result is:

```
Job ‘myjob_1’ was aborted successfully.
```

disable job

Disables a specified job. Disabled jobs are excluded from schedules, nor can you run a disabled job.

Syntax

```
disable job job_name
```

Parameters

- **job_name –** the name of the job to be disabled.

Examples

- **Example 1 –** disables “myjob_1”:

```
disable job myjob_1
go
```

The returned result is:

```
Job ‘myjob_1’ was disabled successfully.
```

drop history

Deletes an existing job history, including report and reconciliation script files.

Syntax

```
drop history job_name n
```

Parameters

- **job_name –** the name of the job for which to delete history.
- **n –** the job history sequence ID of the history to be deleted.

Examples

- **Example 1 –** deletes “myjob_1” history with job history ID 1:

```
drop history myjob_1 1
go
```

enable job

Enables a specified job.

Syntax

```
enable job job_name
```

Parameters

- **job_name –** the name of the job to be enabled.

Examples

- **Example 1 –** enables “myjob_1”:

```
enable job myjob_1
go
```

The returned result is:

```
Job ‘myjob_1’ was enabled successfully.
```

monitor job

Shows runtime status information about running jobs, or jobs that have just finished.

Syntax

```
monitor job [job_name [comparison_name]]
```

Parameters

- **job_name –** the name of the job to be shown.
- **comparison_name –** the name of the comparison to be shown.

Examples

- **Example 1 –** shows runtime information for all jobs:

```
monitor job
go
```

The returned result is:

```
NAME  TYPE            STATUS  SUBMIT TIME  FINISH TIME  ERROR
----  --------------- ------- ------------ -----------  --------
job2  ROW_COMPARE_JOB RUNNING 2010-10-18   09:14:53.358
job6  ROW_COMPARE_JOB RUNNING 2010-10-18   09:14:57.093
```

- **Example 2 –** shows comparison information for job “j1” :

```
monitor job j1 c1
go
```

The returned result is:

```
COMPARISON SUBMIT TIME         END TIME
---------- ------------------- -------------------
 c1         2011-03-01 16:49:34 2011-03-01 16:49:47

(0 rows affected)

 RUN PHASE              TYPE SUMMARY       START TIME         END
TIME
 --- -------------------- ---- -------------- ------------------
------------------
 COUNT READ  M    O I R   PROGRESS ESTIMATE END ERROR
 ----- ---- ---- - - --   -------- ------------ -----
 1   COMPARE_ALL          S    c1/dbo.person  2011-03-01 16:49:34
2011-03-01 16:49:34
```

```
 3072  3072                100%
                           T    c1/dbo.person2 2011-03-01 16:49:34
2011-03-01 16:49:34
 1109  1109 1963 0 0       100%
                           T2   c1/dbo.person3 2011-03-01 16:49:34
2011-03-01 16:49:34
 1109  1109 1963 0 0       100%
 2   RECHECK_DIFFERENCES   T                   2011-03-01 16:49:45
2011-03-01 16:49:45
       0    1963 0 0       100%
                           T2                  2011-03-01 16:49:45
2011-03-01 16:49:45
       0    1963 0 0       100%
 3   VERIFY_DIFFERENCES    S                   2011-03-01 16:49:45
2011-03-01 16:49:46
      1963                100%
                           T                   2011-03-01 16:49:45
2011-03-01 16:49:46
      0    1963 0 0       100%
                           T2                  2011-03-01 16:49:45
2011-03-01 16:49:46
      0    1963 0 0       100%
 4   APPLY_RECONCILIATION  T                   2011-03-01 16:49:46
2011-03-01 16:49:47
                     1963 100%
                           T2                  2011-03-01 16:49:46
2011-03-01 16:49:47
                     1963  100%
```

run job

Starts a specified job.

Syntax

```
run job job_name
```

run job immediately executes the job, regardless of any existing job schedule.

Parameters

- **job_name –** the name of the job to be started.

Examples

- **Example 1 –** executes “myjob_1”:

  ```
  run job myjob_1
  go
  ```

  The returned result is:

  ```
  Job ‘myjob_1’ was started successfully.
  ```

show history

Shows the history, including the report file and reconciliation file path, for a single job.

Syntax

```
show history job_name[historyid]
```

Parameters

- **job_name –** the name of the job for which to show history.
- **historyid –** the job history sequence ID.

Examples

- **Example 1 –** shows “job2” history:

```
show history job2
go
```

The returned result is:

```
HISTORY ID SUBMIT TIME             FINISH TIME
---------- ----------------------- -----------------------
12         2010-10-13 14:38:11.783 2010-10-13 14:38:19.41
```

- **Example 2 –** shows history for “job2” with history ID 12:

```
show history job2 12
go
```

The returned result is:

```
COMPARISON RUN  PHASE       TYPE   SUMMARY
START TIME
---------- ---- ---------   ---  ------------------------------
------------
END TIME           COUNT READ   M O I   RECON ERROR
--------            ----------  -- -- -- ---------------

 c1        1   COMPARE_ALL  S    MACHINEXP1:5000/test.dbo.mycash
2011-02-22 16:09:54
2011-02-22 16:09:54 3     3
                           T    MACHINEXP1:5000/test.dbo.mycash2
2011-02-22 16:09:54
2011-02-22 16:09:54 3     3    0 0 3
           2   RECHECK_DIFFERENCES  T
2011-02-22 16:09:59
2011-02-22 16:09:59       3    0 0 3
           3   VERIFY_DIFFERENCES          S
2011-02-22 16:10:00
2011-02-22 16:10:00       3
                                         T
2011-02-22 16:10:00
```

```
2011-02-22 16:10:00       3    0 0 3
           4   CREATE_RECONCILIATION_SCRIPT T
2011-02-22 16:10:00
2011-02-22 16:10:00                  0
           5   APPLY_RECONCILIATION        T
2011-02-22 16:10:00
2011-02-22 16:10:00                 0
(0 rows affected)

FILE         SERVER PATH
------------ ---------------------------------------------
Recon script C:\Sybase\DA-15_5\server\instance\data\job2\
2011-02-22\14.38.11\c1_T_recon.sql
Text report  C:\Sybase\DA-15_5\server\instance\data\job2\
2011-02-22\14.38.11\report.txt
```

Note: Executing **show history** generates the text and XML reports, if they have not already been generated, before showing the output. The more the differences, **show history** takes a longer time to generate the reports, which causes a delay in showing the results.

show reconcile

Shows the reconciliation script path for a job with a specified history ID.

Syntax

```
show reconcile job_name historyid
```

show reconcile is applicable only to row comparison jobs; it does not work with schema comparison jobs.

Parameters

- **job_name –** the name of the job for which to show the reconciliation script.
- **historyid –** the job history sequence ID of the reconcile script to be shown.

Examples

- **Example 1 –** shows the reconciliation script for "job6" with history ID 29:

  ```
  show reconcile job6 29
  go
  ```

  The returned result is:

```
COMPARISON TARGET                        RUN  SUBMIT TIME
---------- ----------------------------- ---  -----------
    FINISH TIME              RECONCILE SCRIPT
    -----------              ----------------
cmp6       myhost:5000/dadb.dbo.da1_10m 1    2010-10-15 14:04:04.573
    2010-10-15 14:04:04.573   N/A
                                         2    2010-10-15 14:04:26.26
```

```
    2010-10-15 14:04:28.73    C:\Sybase\DA-15_5\server\instance\data
\job6\2012-05-10\14.04.03\cmp6_0_recon.sql
(0 rows affected)
```

show report

Generates and shows the report file path of the job with a specified history ID.

Syntax

```
show report job_name historyid
```

Parameters

- **job_name –** the name of the job for which to show the report.
- **historyid –** the job history sequence ID of the report to be shown.

Examples

- **Example 1 –** shows the report file path for "job6" with history ID 29:

  ```
  show report job6 29
  go
  ```

  The returned result is:

```
REPORT TYPE  SERVER PATH
-----------  ------------------------------------------------------------
TEXT         C:\Sybase\DA-15_5\server\instance\data\job6\
2010-10-15\14.04.03\report.txt
XML          C:\Sybase\DA-15_5\server\instance\data\job6\
2010-10-15\14.04.03\report.xml
(0 row affected)
```

truncate history

Deletes existing job history records or the history records belonging to a single job, including reports and reconciliation scripts.

Syntax

```
truncate history all |(job_name all|historyid)
```

Parameters

- **all –** truncates all job history records.
- **job_name –** the name of the job for which to truncate history.
- **historyid –** the job history sequence ID.

Examples

- **Example 1 –** deletes the history records for all jobs:

```
truncate history all
go
```

- **Example 2 –** deletes the history records up to and including the particular job history ID:

```
truncate history job_name 1
go
```

Import Job Command

Use **import job** to create and configure a job based on table replication definitions and subscriptions in the RSSD.

import job

Creates a job from predefined table replication definitions and subscriptions in Replication Server.

Syntax

```
import job rs_job_name
with connection rssd_connection_name
with map da_connection_name repdefs_ds repdef_db
[and da_connection_name repdefs_ds repdefs_db[...]]
  [set MAX_CONCURRENT_COMPARISONS [{to|=}]
number_of_max_concurrent_comparisons]
  [and set DESC [{to|=}] description]
[with comparison option
  set ABORT_DIFF_MAX [{to|=}] number_of_differences
  [and set ABORT_DIFF_ROW_COUNT [{to|=}] {true|false}
  [and set AUTO_RECONCILE [{to|=}] {true|false}
  [and set COMPARE_MODE [{to|=}] {row_compare | key_compare |
row_checksum | table_checksum | row_count}
  [and set COMPRESS_DATA_TRANSFER [{to|=}] {true|false}
  [and set CREATE_COL_LOG [{to|=}] {true|false}
  [and set CREATE_RECON_SCRIPT [{to|=}] {true|false}
  [and set DESC [{to|=}] description
  [and set ENABLE_ROW_COUNT [{to|=}] {true|false}
  [and set EXTERNAL_SORT [{to|=}] {true|false}
  [and set HASH_TYPE [{to|=}] {database_hash | agent_hash}
  [and set PRIORITY [{to|=}] {highest | high | normal | low}
  [and set RETRY_DELAY_SEC [{to|=}] number_delay_second
  [and set RETRY_DIFF [{to|=}] {never | wait_and_retry }
  [and set RETRY_MAX [{to|=}] number_of_retries
]]]]]]]]]]]]]]]
[add schedule schedule_name
[set TYPE [{to|=}] {once | cron | every_day | every_week |
every_month}
[and set EVERY [{to|=}] n
[and set DATE [{to|=}] date_value
```

```
[and set TIME [{to|=}] time_value
[and set KEEP [{to|=}] keep_value
[and set KEEP_UNIT [{to|=}] {day | week | month | forever}
[and set CRON [{to|=}] cron_value
[and set DESC [{to|=}] description
]]]]]]]]]
```

Parameters

- **rs_job_name –** the name of the Replication Server job to be created.
- **rssd_connection_name –** the name of the existing RSSD connection.
- **da_connection_name –** the name of the Data Assurance (DA) server connection.
- **repdef_ds –** the name of the datasource defined in the replication definition.
- **repdef_db –** the name of the database defined in the replication definition.
- **schedule_name –** the name of the schedule to be added.
- **max_concurrent_comparisons –** (Optional) the number of the comparisons that can be run concurrently with a job. The default value is 5.
- **description –** (Optional) description of the agent. Use double quotes if you are using a reserved word or blank spaces.

Table 10. Comparison Options

Parameter	Value
ABORT_DIFF_MAX	Aborts row comparison if the difference count exceeds the specified value. Valid values: 1 to 9223372036854775807. Default value:1000.
ABORT_DIFF_ROW_COUNT	Determines whether to abort row comparison if table row counts do not match. Valid values: true or false. Default value: false.
AUTO_RECONCILE	Indicates whether to automatically apply the reconciliation script. Valid values: true or false. Default value: false. **Note:** To enable **AUTO_RECONCILE**, set **CREATE_COL_LOG** to true.

Parameter	Value
COMPARE_MODE	Specifies the row comparison mode. • `row_compare` – compares all table rows. • `key_compare` – compares the primary key columns. • `row_count` – compress row count. Default value: row_compare.
COMPRESS_DATA_TRANSFER	Compresses the row data between the agent and the server. Valid values: true or false. Default value: false.
CREATE_COL_LOG	Generates a column differences log, which lists all missing, orphaned, and inconsistent row values (keys and columns). Create a column log if you want to: • Generate a reconciliation script • Perform automatic reconciliation • Generate a detailed report Valid values: true or false. Default value: false.
CREATE_RECON_SCRIPT	Generates a reconciliation script. To use this parameter, you must also set **CREATE_COL_LOG** to true. Valid values: true or false. Default value: false.
ENABLE_ROW_COUNT	Determines whether or not to count source and target table rows before they are compared. DA server uses the row count to estimate the comparison progress and end time. **Note:** DA server counts rows if **COMPARE_MODE** is row_count. Use **ENABLE_ROW_COUNT** only if **COMPARE_MODE** is a value other than row_count. Valid values: true or false. Default value: true.

Parameter	Value
EXTERNAL_SORT	Sorts rows on the agent, thereby reducing the impact of processing the **ORDER BY** clause in the databases. Valid values: true or false. Default value: false.
HASH_TYPE	Specifies the hash type for the comparison. • `database_hash` – use the hash function provided by the database. • `agent_hash` – use the hash function provided by Replication Server Data Assurance Option. Default value: database_hash.
PRIORITY	Specifies the job comparison order in the comparison queue. Valid values are: • `highest` • `high` • `normal` • `low` Default value: normal.
RETRY_DIFF	Specifies the retry option. • `never` – no recompare. • `wait_and_retry` – run recompare based on **RETRY_MAX** and **RETRY_DELAY_SEC** settings. Default value: never.
RETRY_DELAY_SEC	Specifies the number of seconds delay for each recomparison. Valid values: 0 to 86400. Default value: 10.

Parameter	Value
RETRY_MAX	Specifies the total number of recomparison for rows that have differences resulting from a previous comparison. Valid values: 0 to 100. Default value: 3.

Table 11. Column Comparison Option

Column Option	Value
COMPARE_MODE	Specifies for how each column is compared. • column_hash – compares using column hash value. • row_hash – compares all columns with this option together with a whole hash value. • literal – compares using column literal value.

Table 12. Scheduling Options

Parameter	Value
date_value	Specifies a date in the scheduler.
time_value	Specifies a time in the scheduler.
keep_value	Specifies the number of keep units for which this schedule remains active.
cron_value	Specifies the cron option value in the scheduler.

Examples

- **Example 1 –** creates a new job named “myrsjob_1”:

```
import job myrsjob_1
with rssd connection MyRSSDConn
with map MyConnPDB1 repdef_ds repdef_db
with map MyConnRDB1 repdef_ds2 repdef_db2
set max_concurrent_comparisons = 3
with comparison options
set COMPARE_MODE= row_compare
and set ABORT_DIFF_MAX = 20
and set ABORT_DIFF_ROW_COUNT = true
and set RETRY_DIFF = wait_and_retry
and set RETRY_MAX= 2
and set RETRY_DELAY_SEC = 10
and set HASH_TYPE = database_hash
```

```
with schedule myschedule_1
set type=every_day
and set every=2
and set time=10:00
and set keep=1
and set keep_unit=month
go
```

The returned result is:

```
Job “myrsjob_1” was created successfully.
```

Data Assurance System Database (DASD) Commands

Commands for managing the DASD.

create backup

Creates a backup of the current Data Assurance System Database (DASD) database. Backup files are stored in `da\server\`*`instance`*`\dasd\backup\`*`unique_backup_id`* .

Syntax

create backup

Examples

- **Example 1 –** creates DASD backup:

```
create backup
go
```

drop backup

Deletes a specific backup specified by the `backup_index`.

Syntax

drop backup *backup_index*

Parameters

- **backup_index –** specifies the backup index entry.

Examples

- **Example 1 –** deletes backup with index entry 3:

```
drop backup 3
go
```

restore backup

Restores the Data Assurance System Database (DASD) database from a backup copy.

Syntax

restore backup *backup_index*

Examples

- **Example 1 –** restores the DASD:

```
restore backup 3
go
```

Usage

- If **restore backup** succeeds, the server automatically shuts down; you must manually restart it.

show backup

Shows where the Data Assurance System Database (DASD) is backed up.

Syntax

show backup

Examples

- **Example 1 –** shows the DASD backup path:

```
show backup
go
```

The returned result is:

```
INDEX    DATE                    PATH
-----  ------------------  ---------------------------------------------
1      2011-1-12 13:29:58  C:\Sybase\DA-15_5\server\myserver\dasd\backup
                                                                    \1297407
(0 row affected)
```

truncate backup

Deletes all existing backups or a specific backup.

Syntax

truncate backup `all | backup_index`

Parameters

- **all –** truncates all backups.
- **backup_index –** the backup index entry.

Examples

- **Example 1 –** deletes all backups:

```
truncate backup all
go
```

- **Example 2 –** deletes backup index entry 3:

```
truncate backup 3
go
```

Note: In this example, **truncate backup** deletes all previous backups (1 and 2), including the one indicated by the *backup_index*.

Other Commands

Commands for configuring and troubleshooting DA server.

config

Configures and shows DA server configuration parameters.

Syntax

config [**parameter_name** [*parameter_value*]]

Parameters

- **parameter_name –** the DA server parameter to be set.
- **parameter_value –** the value of the DA server parameter.

The current values of all the global configuration parameters are stored in the Data Assurance System Database (DASD).

Table 13. Global Configuration Parameters

parameter_name	parameter_value
agent_client_ctx_timeout_secs	Specifies the connection timeout, in seconds, between the DA server and the DA agent. Default: 5 Min: 1 Max: 2147483647 This parameter does not require a restart of DA server to take effect.
agent_access_timeout_mins	Specifies the length of time the connection between the DA server and the DA agent remains open, even when there is no activity between them. Default: 60 Min: 1 Max: 2147483647 This parameter does not require a restart of DA server to take effect.
agent_max_queue	Specifies the maximum number of rows the agent buffers in its output queue. The DA server reads the rows from the queue. The agent temporarily stops reading rows from the database table when the queue is full. Default: 1000 Min: 1 Max: 2147483647 This parameter does not require a restart of DA server to take effect.
agent_max_request_queue	Specifies the maximum queue size for server requests for retry comparison and reconciliation. Default: 100 Min: 1 Max: 2147483647 This parameter requires a restart of DA server to take effect.

parameter_name	parameter_value
auto_recon_stmt_batch_size	Specifies the maximum number of SQL statements the DA server sends to the DA agent in a single invocation. Default: 100 Min: 1 Max: 10000 This parameter does not require a restart of DA server to take effect.
clt_password_encryption_reqd	Determines the level of password encryption the server requires. Default: 0 Valid values: • 0 – allows the client to choose the encryption algorithm used for login passwords on the network, including no password encryption. • 1 – restricts clients to use only the RSA encryption algorithms to encrypt login passwords on the network. This provides strong RSA encryption of passwords. Clients that attempt to connect without using the RSA encryption fail. This parameter does not require a restart of DA server to take effect.
comparer_max_concurrent_threads	Specifies the maximum number of comparison threads used for concurrent comparisons. Default: 5 Min: 1 Max: 2147483647 This parameter does not require a restart of DA server to take effect.

parameter_name	parameter_value
comparer_retry_delay_threshold_secs	Specifies the threshold value, in seconds, a DA server comparison can hold on to a comparison thread before attempting a retry. If the value is higher than **retry_delay_sec**, the DA server comparison holds on to the current comparison thread while waiting to retry. This may delay another comparison thread that is in the queue from starting. If the value is less than or equal to **retry_delay_sec**, the DA server comparison releases the current comparison thread and starts processing the next comparison in the queue. Default: 20 Min: 1 Max: 2147483647 This parameter does not require a restart of DA server to take effect.
comparer_recently_finished_ttl_secs	Specifies the maximum time, in seconds, for job information to remain in the monitor job view. Default: 300 Min: 1 Max: 2147483647 This parameter does not require a restart of DA server to take effect.
comparer_retry_max_keys_per_clause	The maximum number of single keys in a **WHERE** clause. Default: 10 Min: 1 Max: 100 This parameter does not require a restart of DA server to take effect.
comparer_retry_min_keys_in_range	Specifies the minimum number of keys used when calculating the selection criteria for keys as a range rather than as individuals. Default: 5 Min: 2 Max: 100 This parameter does not require a restart of DA server to take effect.

parameter_name	parameter_value

comparer_retry_min_fill_percent	Specifies the minimum fill percentage required when combining “single” keys into a range. When selecting a result set of adjacent or near-adjacent row keys, it is usually faster to select keys in a range rather than specifying each key separately in your statement.
	For example, to select every alternate rows between 1 to 100, use:
	`"select ... where id in(1,3,5,7..97,99)"`
	Alternatively, you can fetch all rows in a range:
	`"select ... where id >=1 and id <= 100"`
	Fetching all rows in a range is typically faster than running one or more `in(...)` statements. The above example has a fill percentage of 50, because only half of the selected rows are required. DA server skips all the extra rows.
	Default: 10
	Min: 1
	Max: 100
	This parameter does not require a restart of DA server to take effect.

comparer_retry_min_fill_percent_literal	Specifies the minimum fill percentage required when combining “single” keys into a range for literal comparison. Default: 90 Min: 1 Max: 100 **Note:** Typically, you should set **comparer_retry_min_fill_percent_literal** to a higher value than **comparer_retry_min_fill_percent** because the cost of transmitting extra literal row data soon outweighs the performance benefit of range selection. This parameter does not require a restart of DA server to take effect.

parameter_name	parameter_value
db_connection_retry_times	Specifies the maximum number of retries by the connection manager to connect to a database if the initial attempt fails. Default: 2 Min: 1 Max: 2147483647 This parameter does not require a restart of DA server to take effect.
db_connection_retry_interval	Specifies the maximum wait time for the connection manager between successive database connection attempts. Default: 3 Min: 1 Max: 2147483647 This parameter does not require a restart of DA server to take effect.
default_column_compare_mode	Specifies the default compare mode for columns. Default: column_hash Valid values: literal, column_hash, row_hash This parameter does not require a restart of DA server to take effect.
db_hash_ase_algorithm	Specifies the hash algorithm for the Adaptive Server database. Default: MD5 Valid Values: MD5 or SHA This parameter does not require a restart of DA server to take effect.
db_hash_ase_ignore_null	Specifies whether to ignore the issue of the Adaptive Server hashbytes limitation when calculating multihash values for the Adaptive Server database. Default: true Valid values: true or false This parameter does not require a restart of DA server to take effect.

parameter_name	parameter_value
db_hash_ase_using_option	Specifies the byte order option in the hash algorithm for the Adaptive Server database. Default: UNICODE_LSB Valid values: LSB, MSB, UNICODE, UNICODE_LSB, UNICODE_MSB This parameter does not require a restart of DA server to take effect.
enable_report_generator	Specifies whether or not to generate job reports. When you set **enable_report_generator** to false, it prevents the report generator from creating XML and text reports when a job history item is viewed and the reports have not yet been generated. This may be useful if the column log is very large, and the reports may take a long time to generate. Default: true Valid values: true or false This parameter does not require a restart of DA server to take effect.
external_sort_max_thread	Specifies the maximum number of threads used for external sort. Default: 5 Min: 3 Max: 10 This parameter does not require a restart of DA server to take effect.
external_sort_max_size	Specifies the maximum number of rows that can be sorted in memory. Default: 1000000 Min: 2 Max: 2147483647 This parameter does not require a restart of DA server to take effect.

parameter_name	parameter_value
external_sort_max_file	Specifies the maximum number of files used for external sort. Default: 60 Min: 10 Max: 2147483647 This parameter does not require a restart of DA server to take effect.
external_sort_compress_file	Specifies whether or not to compress the data files. Default: false Valid values: true or false This parameter does not require a restart of DA server to take effect.
external_sort_activate_size	Specifies the minimum number of rows required in a database table for activating external sort. Default: 1000000 Min: 2 Max: 2147483647 This parameter does not require a restart of DA server to take effect.
file_output_encoding	Specifies the file output encoding for all reconciliation scripts and report files. Valid values: Any character set encoding supported by the DA server Java Runtime Environment (JRE). This parameter does not require a restart of DA server to take effect.
lob_fetch_size	Specifies the maximum number of large object (LOB) bytes to read and compare. Default: 1024 Min: 1 Max: 1048576 This parameter does not require a restart of DA server to take effect.

parameter_name	parameter_value
recon_tran_max_stmts	Specifies the maximum number of statements in a reconciliation transaction. If the number of statements needed is greater than the specified number, you need multiple transactions. A value of zero means unlimited number of statements in a transaction. Default: 0 Min: 0 Max: 2147483647 This parameter does not require a restart of DA server to take effect.
text_report_max_column_width	Specifies the maximum column width in a text report. Default: 30 Min: 10 Max: 80 This parameter does not require a restart of DA server to take effect.
text_report_max_line_length	Specifies the maximum line length in a text report. Default: 200 Min: 100 Max: 1000 This parameter does not require a restart of DA server to take effect.

Examples

- **Example 1 –** shows all the configuration parameters:

```
config
go
```

The returned result is:

```
NAME                                VALUE   PENDING   REQUIRE RESTART
----------------------------------  -----   -------   ---------------
agent_access_timeout_mins             60                  false
agent_client_ctx_timeout              5                   false
agent_max_queue                       1000                false
agent_max_request_queue               100                 true
auto_recon_stmt_batch_size            100                 false
clt_password_encryption_reqd          1                   false
comparer_max_concurrent_threads       5                   false
comparer_recently_finished_ttl_secs   300                 false
```

```
comparer_retry_delay_threshold_secs      20                  false
comparer_retry_max_keys_per_clause       10                  false
comparer_retry_min_fill_percent          10                  false
comparer_retry_min_fill_percent_literal  90                  false
comparer_retry_min_keys_in_range         5                   false
db_connection_retry_interval             3                   false
db_connection_retry_times                2                   false
db_hash_ase_algorithm                    MD5                 false
db_hash_ase_ignore_null                  false               false
db_hash_ase_using_option                 UNICODE_LSB         false
default_column_compare_mode              ROW_HASH            false
enable_report_generator                  true                false
external_sort_activate_size              1000000             false
external_sort_compress_file              false               false
external_sort_max_file                   64                  false
external_sort_max_size                   100000              false
external_sort_max_thread                 5                   false
file_output_encoding                     cp936               false
lob_fetch_size                           1024                false
recon_tran_max_stmts                     0                   false
text_report_max_column_width             30                  false
text_report_max_line_length              200                 false
```

- **Example 2 –** changes the default value (MD5) for **db_hash_ase_algorithm** to a SHA:

  ```
  config db_hash_ase_algorithm SHA
  go
  ```

  The returned result is:

  ```
   NAME                                    VALUE       PENDING
   --------------------------------------- ----------- -------
   db_hash_ase_algorithm                   SHA
  (1 rows affected)
   DEFAULT VALID       EXPLANATION
   ------- ---------  ----------------------------------------
   MD5     MD5,SHA      The database hash algorithm for ASE
  ```

- **Example 3 –** changes the required encryption level to “encryption level 1”. If you set this configuration parameter to a nonzero value, you see a warning message.

  ```
  config clt_password_encryption_reqd 1
  go
  ```

  The returned result is:

  ```
  [#90] Warning: you have set the password encryption level to 1;
  please ensure your client tool supports this level of encryption,
  otherwise you will not be able to login again without upgrading
  your client tool.

  (1 row affected)
  ```

password

Changes the DA server login password.

password does not return a result set. If the current password is incorrect, or the new password is invalid, you see an error message.

Syntax

```
password current_password new_password
```

Parameters

- **current_password –** the existing password for the administration user login name.
- **new_password –** the new password for the administration user login name. The default minimum password length is 6 and the maximum password length is 30. You can configure the password length in the *instance*.cfg. Valid characters for input values are a-z, A-Z, 0-9, -, and _.

Examples

- **Example 1 –** changes the da_admin password from “sybase” to “onesybase”:

```
password sybase onesybase
go
```

See also

- *Password Policy* on page 120

role

Maps LDAP users to the DA administrator role.

Syntax

```
role [rolename [add|drop user username]]
```

Parameters

- **rolename –** case-sensitive role name.
- **username –** case-sensitive user name.

Examples

- **Example 1 –** shows all roles and users:

```
role
go
```

The returned result is:

```
ROLE       USER          LOCAL USER
--------   ---------     ----------
DA_Admin   da_admin      true
DA_Admin   srjones       false
```

- **Example 2 –** shows all users with the DA server administrator role:

```
role DA_Admin
go
```

The returned result is:

```
ROLE       USER          LOCAL USER
--------   ---------     ----------
DA_Admin   da_admin      true
DA_Admin   srjones       false
```

- **Example 3 –** adds "tabraham" to the DA server administrator role:

```
role DA_Admin add user tabraham
go
```

- **Example 4 –** drops "tabraham" from the DA server administrator role:

```
role DA_Admin drop user tabraham
go
```

show jvm

Shows some of the important Java Virtual Machine (JVM) details.

The command requires no arguments.

Syntax

```
show jvm
```

Examples

- **Example 1 –** shows JVM details:

```
show jvm
go
```

The returned result is:

```
JVM NAME                    JVM INFO   JVM VENDOR
         JVM VERSION
--------------------------- ---------- -----------
         -------------------------
Java HotSpot(TM) Server VM mixed mode Sun Microsystems Inc.
         Java 1.6.0_24, VM 19.1-b02
(0 rows affected)
```

```
JVM TOTAL MEM JVM FREE MEM JVM MAX MEM
------------- ------------ -----------
31.8 MB       27.1  MB     455.1 MB
```

show system

Shows some of the important system properties.

The command requires no arguments.

Syntax

```
show system
```

Examples

- **Example 1 –** shows system details:

```
show system
go
```

The returned result is:

```
NAME      HOST         LOCALE  TIME ZONE            DATE       TIME
--------  -----------  ------  -------------------  ---------- --------
myserver  10.65.0.111  en_GB   Greenwich Mean Time  2011-06-10 16:05:44

OS NAME     OS VERSION   OS ARCH   OS LOAD AVG
----------  ----------   -------   -----------
Windows XP  5.1          x86       14.897%
```

sslconfig

Configures and shows all SSL (Secure Sockets Layer) configuration parameters.

Syntax

```
sslconfig [parameter_name [parameter_value]]
```

Parameters

- **parameter_name –** the SSL parameter to be set.
- **parameter_value –** the value of the SSL parameter.

parameter_name	parameter_value
dts_client_ssl_required	A comma- or semicolon-delimited list of DA agent host names. When DA server connects to a DA agent data transfer socket, it checks the DA agent host name against this list. If the DA agent host name is found, DA server connects using an SSL socket. You must restart DA server for this parameter to take effect.
keypair_passwd	The password to grant access to the public/private key pair within the keystore. This value is encrypted on disk. When this value is displayed in the command line tool (CLT), a ******** placeholder indicates it is set to a nonblank value. You must restart DA server for this parameter to take effect.
keystore	The absolute path to the keystore (flat file) that contains the public/private key pair to use. You must restart DA server for this parameter to take effect.
keystore_passwd	The password to grant access to the keystore. This value is encrypted on disk. When this value is displayed in the CLT, a ******** placeholder indicates it is set to a nonblank value. You must restart DA server for this parameter to take effect.
rmi_client_ssl_required	A comma- or semicolon-delimited list of DA agent host names. When DA server creates an RMI connection to a DA agent, it checks the DA agent host name against this list. If the DA agent host name is found, DA server connects using an SSL socket. You must restart DA server for this parameter to take effect.
rmi_server_ssl_enabled	Whether to connect all RMI clients using SSL. If this value is set to true, DA server requires all RMI clients to connect using SSL. Default: false. This parameter requires a restart of DA server to take effect.
truststore	The absolute path to the truststore (flat file) that contains the trusted certificate. You must restart DA server for this parameter to take effect.

parameter_name	parameter_value
truststore_passwd	The password to grant access to the truststore. This value is encrypted on disk. When this value is displayed in the CLT, a ******** placeholder indicates it is set to a nonblank value. You must restart DA server for this parameter to take effect.

Examples

- **Example 1 –** shows all SSL configuration parameters:

```
sslconfig
go
```

The returned result is:

```
NAME                     VALUE               PENDING  REQUIRE RESTART
-----------------------  ------------------- -------  ---------------
dts_client_ssl_required  venus,pluto                          true
keypair_passwd           ********                             true
keystore                 C:/SSL/mars_trust.ks                 true
keystore_passwd          ********                             true
rmi_client_ssl_required  venus,pluto                          true
rmi_server_ssl_enabled   true                                 true
truststore               C:/SSL/mars_trust.ks                 true
truststore_passwd        ********                             true

(0 rows affected)
```

- **Example 2 –** shows the current value and its description of a SSL parameter:

```
sslconfig keystore
go
```

The returned result is:

```
NAME                 VALUE                PENDING  REQUIRE RESTART
------------------   -------------------- -------  ---------------
keystore             C:/SSL/mars_trust.ks               true

(0 rows affected)

DEFAULT MINIMUM MAXIMUM EXPLANATION
------- ------- ------- ------------------------------
                        The absolute path to a keystore
                        (server-side configuration).

(0 rows affected)
```

- **Example 3 –** changes the default value of an SSL parameter:

```
sslconfig rmi_server_ssl_enabled false
go
```

trace

Configures the level of system trace and returns the trace flag settings.

Syntax

trace [*flag*| all [*level*]]

Parameters

- **flag –** the name of the trace flag. Available flag names in the server container are: agent, audit, clt, compare, container, dasd, license, server, sql, and std.
- **all –** specifies all trace flags in the system.
- **level –** specifies the trace level. Available levels are: off, severe, warning, info, config, fine, finer, finest, and all.

Examples

- **Example 1 –** shows the trace level:

  ```
  trace
  go
  ```

  The returned result is:

  ```
  TRACE                LEVEL
  -------------------- ----------
  agent                INFO
  audit                ALL
  clt                  INFO
  compare              INFO
  container            INFO
  dasd                 SEVERE
  license              INFO
  server               INFO
  sql                  INFO
  std                  ALL
  ```

version

Shows the current version of the Replication Server Data Assurance Option.

Syntax

version

Examples

- **Example 1 –** shows the version:

```
version
go
```

The returned result is:

```
VERSION
-----------------------------------------------------------------------
Replication Server Data Assurance Option/15.7.1/DA Server/P/generic/
generic/da157x/121/VM: Sun Microsystems Inc. 1.6.0_24/OPT/Tue 24 Apr
2012 09:24:31 GMT
```

Reserved Words for Data Assurance Server

Reserved words have special meaning in DA server when used as part of a command. DA server does not allow words that are part of command syntax, unless you set the word in double quotes.

Table 14. DA Server Reserved Words

	Words
A	abort, add, agent, all, alter, and
B	backup
C	compareset, comparison, config, connection, create
D	dasd, depend, disable, drop, dts
E	enable, exclude
F	force
H	history
I	immediately, import, include
J	job, jvm
L	license
M	map, monitor
N	node
O	option

	Words
P	password, properties
R	reconcile, report, restore, role, rsjob, run
S	schedule, schema, set, show, shutdown, source, sslconfig, system
T	tables, target, task, test, to, trace, truncate
U	user
V	version
W	where, with

Remote Data Assurance Agent Command Reference

You can execute remote DA agent commands with **isql** or the Sybase Control Center Data Assurance plug-in.

Note: You must have “da_admin” permission to execute all DA agent commands.

config

Configures and shows DA agent configuration parameters.

Syntax

config [**parameter_name** [*parameter_value*]]

Parameters

- **parameter_name –** the DA agent parameter to be set.
- **parameter_value –** the value of the DA agent parameter.

  The current configuration parameter value is stored in the configuration file.

parameter_name	parameter_value
clt_password_encryption_reqd	Determines the level of password encryption the agent requires. Default: 0 Valid values: • 0 – allows the client to choose the encryption algorithm used for login passwords on the network, including no password encryption. • 1 – restricts clients to use only the RSA encryption algorithms to encrypt login passwords on the network. This provides strong RSA encryption of passwords. Clients that attempt to connect without using the RSA encryption fail. This parameter does not require a restart of DA agent to take effect.

Examples

- **Example 1 –** changes the required encryption level to "encryption level 1". If you set this configuration parameter to a nonzero value, you see a warning message.

```
config clt_password_encryption_reqd 1
go
```

The returned result is:

```
[#90] Warning: you have set the password encryption level to 1;
please ensure your client tool supports this level of encryption,
otherwise you will not be able to login again without upgrading
your client tool.

(1 row affected)
```

password

Changes the DA agent login password.

password does not return a result set. If the current password is incorrect, or the new password is invalid, you see an error message.

Syntax

password *current_password new_password*

Parameters

- **current_password –** the existing password for the administration user login name.
- **new_password –** the new password for the administration user login name. The default minimum password length is 6 and the maximum password length is 30. You can configure the password length in the *instance*.cfg. Valid characters for input values are a-z, A-Z, 0-9, -, and _.

Examples

- **Example 1 –** changes the da_admin password from "sybase" to "onesybase":

```
password sybase onesybase
go
```

Usage

When you change the password for a DA agent, you must also change the agent password configured in the DA servers that connect to that DA agent. Failure to do so results in the DA server not being able to authenticate with the DA agent.

See also

- *alter agent* on page 33
- *Password Policy* on page 120

role

Maps LDAP users to the DA administrator role.

Syntax

```
role [rolename [add|drop user username]]
```

Parameters

- **rolename –** case-sensitive role name.
- **username –** case-sensitive user name.

Examples

- **Example 1 –** shows all roles and users:

  ```
  role
  go
  ```

  The returned result is:

  ```
  ROLE       USER          LOCAL USER
  --------   ---------     ----------
  DA_Admin   da_admin      true
  DA_Admin   srjones       false
  ```

- **Example 2 –** shows all users with the DA administrator role:

  ```
  role DA_Admin
  go
  ```

  The returned result is:

  ```
  ROLE       USER          LOCAL USER
  --------   ---------     ----------
  DA_Admin   da_admin      true
  DA_Admin   srjones       false
  ```

- **Example 3 –** adds "tabraham" to the DA administrator role:

  ```
  role DA_Admin add user tabraham
  go
  ```

- **Example 4 –** drops "tabraham" from the DA administrator role.

  ```
  role DA_Admin drop user tabraham
  go
  ```

show connection

Shows the database connections for a remote agent.

Syntax

```
show connection
```

Examples

- **Example 1 –** shows the remote DA agent connections:

```
show connection
go
```

The returned result is:

```
SERVER                              NAME       TYPE CONNECTED
----------------------------------- ---------- ---- ---------
myserver:4500@soka.sybase.com       conn1_23mw ASE  3
myserver:4500@soka.sybase.com       soka2_ra   ASE  2
myserver:4500@etlaix61.sybase.com   conn1_h33  ASE  2
```

show dts

Shows the data transfer stream (DTS) information that is running on a remote agent.

Syntax

```
show dts
```

Examples

- **Example 1 –** shows all the DTS information for a remote agent:

```
show dts
go
```

The returned result is:

```
TASK ID ESTIMATE COUNT FETCHING  QUEUE TAKEN ESTIMATE SECONDS LEFT
------- -------------- --------  ----- ----- ---------------------
3       1000                      0     0
```

show jvm

Shows some of the important Java Virtual Machine (JVM) details.

The command requires no arguments.

Syntax

```
show jvm
```

Examples

- **Example 1 –** shows JVM details:

```
show jvm
go
```

The returned result is:

```
JVM NAME                    JVM INFO   JVM VENDOR
         JVM VERSION
--------------------------- ---------- -----------
         -------------------------
Java HotSpot(TM) Server VM mixed mode Sun Microsystems Inc.
         Java 1.6.0_24, VM 19.1-b02
(0 rows affected)

JVM TOTAL MEM JVM FREE MEM JVM MAX MEM
------------- ------------ -----------
31.8 MB       27.1  MB     455.1 MB
```

show system

Shows some of the important system properties.

The command requires no arguments.

Syntax

```
show system
```

Examples

- **Example 1 –** shows system details:

```
show system
go
```

The returned result is:

```
NAME      HOST          LOCALE  TIME ZONE             DATE       TIME
-------   -----------   ------  -------------------   ---------- --------
myagent   10.65.0.111   en_GB   Greenwich Mean Time   2011-06-10 16:05:44

OS NAME     OS VERSION   OS ARCH   OS LOAD AVG
----------  ----------   -------   -----------
Windows XP  5.1          x86       14.897%
```

show task

Shows the task information for a remote agent.

Syntax

```
show task
```

Examples

- **Example 1 –** shows all the tasks for the remote DA agent:

  ```
  show task
  go
  ```

  The returned result is:

```
SERVER                         TASK ID CONNECTION  OBJECT      STAGE OBJ
PROCESSED PREDICATE SQL PROCESSED
------------------------------ ------- ----------  ----------- ----- ---
--------- ----------------------
myserver:4500@soka.sybase.com  35     conn1_1t8p  dbo.da1_10m     0
myserver:4500@soka.sybase.com  37     conn1_23mw  dbo.da1_10m     0

 TASK ID ESTIMATE COUNT QUEUE TAKEN ESTIMATE SECONDS LEFT
 ------- -------------- ----- ----- ---------------------
 35      10000          0
 37      10000          0
```

sslconfig

Configures and shows all SSL (Secure Sockets Layer) configuration parameters.

Syntax

```
sslconfig [parameter_name [parameter_value]]
```

Parameters

- **parameter_name –** the SSL parameter to be set.

- **parameter_value –** the value of the SSL parameter.

parameter_name	parameter_value
dts_server_ssl_enabled	Whether DA agent creates its data transfer socket using SSL. Default: false. You must restart DA agent for this parameter to take effect.
keypair_passwd	The password to grant access to the public/private key pair within the keystore. This value is encrypted on disk. When this value is displayed in the command line tool (CLT), a ******** placeholder indicates it is set to a nonblank value. You must restart DA agent for this parameter to take effect.
keystore	The absolute path to the keystore (flat file) that contains the public/private key pair to use. You must restart DA agent for this parameter to take effect.
keystore_passwd	The password to grant access to the keystore. This value is encrypted on disk. When this value is displayed in the CLT, a ******** placeholder indicates it is set to a nonblank value. You must restart DA agent for this parameter to take effect.
rmi_server_ssl_enabled	Whether DA server requires all RMI clients to connect using SSL. Default: false. You must restart DA agent for this parameter to take effect.
truststore	The absolute path to the truststore (flat file) that contains the trusted certificate. You must restart DA agent for this parameter to take effect.
truststore_passwd	The password to grant access to the truststore. This value is encrypted on disk. When this value is displayed in the CLT, a ******** placeholder indicates it is set to a nonblank value. You must restart DA agent for this parameter to take effect.

Examples

- **Example 1 –** shows all SSL configuration parameters:

```
sslconfig
go
```

The returned result is:

```
 NAME                     VALUE      PENDING  REQUIRE RESTART
 -----------------------  -------    -------  ---------------
 dts_server_ssl_enabled   false                      true
 keypair_passwd                                     true
 keystore                                           true
 keystore_passwd                                    true
 rmi_server_ssl_enabled   false                      true
 truststore                                         true
 truststore_passwd                                  true

(0 rows affected)
```

- **Example 2 –** shows the current value and its description of a SSL parameter:

  ```
  sslconfig  dts_server_ssl_enabled
  go
  ```

  The returned result is:

```
 NAME                    VALUE   PENDING  REQUIRE RESTART
 ----------------------  ------  -------  ---------------
 dts_server_ssl_enabled  false                 true

(0 rows affected)

DEFAULT MINIMUM MAXIMUM EXPLANATION
------- ------- ------- ------------------------------
                        Whether this DA agent should use
                        SSL when streaming data via its DTS.

 (0 rows affected)
```

- **Example 3 –** changes the default value of an SSL parameter:

  ```
  sslconfig rmi_server_ssl_enabled true
  go
  ```

trace

Configures the level of system trace and returns the trace flag settings for the remote DA agent.

Syntax

trace [*flag*| all [*level*]]

Parameters

- **flag –** the name of the trace flag. Available flag names in the agent container are: agent, audit, clt, container, sql, and std.

- **all –** specifies all trace flags in the system.
- **level –** specifies the trace level. Available levels are: off, severe, warning, info, config, fine, finer, finest, and all.

Examples

- **Example 1 –** shows the trace level:

```
trace
go
```

The returned result is:

```
TRACE     LEVEL
--------- -----
agent     INFO
audit     ALL
clt       INFO
container INFO
sql       INFO
std       ALL
```

version

Shows the current version of the Replication Server Data Assurance Option.

Syntax

```
version
```

Examples

- **Example 1 –** shows the version:

```
version
go
```

The returned result is:

```
VERSION
-----------------------------------------------------------------
Replication Server Data Assurance Option/15.7.1/DA Agent/P/generic/
generic/da157x/121/VM: Sun Microsystems Inc. 1.6.0_24/OPT/Tue 24 Apr
 2012 09:24:31 GMT
```

Reserved Words for Data Assurance Agent

Reserved words have special meaning in DA agent when used as part of a command. DA agent does not allow words that are part of command syntax, unless you set the word in double quotes.

Table 15. DA Agent Reserved Words

	Words
C	config, connection
D	dts
J	jvm
P	password
R	role
S	show, shutdown, sslconfig, system
T	task
U	user
V	version

Security and Access Control

Administer security and access control for Data Assurance.

Kerberos Security

Kerberos is a network-based authentication protocol for client-server communication.

Kerberos provides a centralized and secure authentication mechanism in enterprise environments that employ the Kerberos infrastructure. Authentication occurs with a trusted, third-party server called a key distribution Center (KDC) that verifies both the client and the server

Configuring DA Agent for Kerberos

Configure your DA agent to accept Kerberos settings in a distributed deployment when connecting to a database using Java Database Connectivity (JDBC).

In this example, the remote DA agent is installed on the server called “omnivore.”

Note: In a standalone DA server deployment, use the same steps described in this procedure to configure the local agent (embedded with DA server) to work with Kerberos.

1. Go to `$SYBASE/DA-15_5/agent/`*`instance`*`/`*`instance`*`.cfg`.
2. Edit the *`instance`*`.cfg` file to set the **sun.security.krb5.debug** to true, if you want to troubleshoot any problems.

```
 #
 # Kerberos
 #
 javax.security.auth.useSubjectCredsOnly=false
 java.security.auth.login.config=${da.instance.dir}/security/
kerberos.conf
 sun.security.krb5.debug=true
```

3. Go to `$SYBASE/DA-15_5/agent/`*`instance`*`/security/`.
4. Edit the `kerberos.conf` file to include the `principal` name and the `keytab` file location:

```
com.sun.security.jgss.initiate {
         com.sun.security.auth.module.Krb5LoginModule required
doNotPrompt=true
         debug=true principal=USERNAME
         useKeyTab=true keyTab="C:\\ASE1503_krb\\SERVERNAME_key"
storeKey=true;
```

5. Restart the DA agent.
6. Verify that DA agent is installed on the server “omnivore”:

```
show agent a1
go
```

```
 NAME         HOST      PORT USER      DESCRIPTION
 ------------ --------- ---- --------- --------------
 a1           omnivore  4510 da_admin
(0 rows affected)
```

7. Create a database connection for the DA agent “a1” with the dummy user name “my_user”:

```
create connection c2
set agent a1
and set host omnivore
and set port 5000
and set database dadb
and set user my_user
with properties
 set request_kerberos_session true
    and set service_principal_name “OMNIVORE@ASE”
go
```

8. Test the database connection settings:

```
test connection c2
go
```

```
RESULT
----------
Succeeded
(0 rows affected)
```

LDAP Authentication

LDAP (Lightweight Directory Access Protocol) is an industry standard client/server protocol for accessing a directory service. An LDAP server is often used as a user repository and central authentication service.

DA supports the ability to bind LDAP users as DA administrators and the ability to delegate LDAP user authentication to an external LDAP authentication server.

DA Administrator Role

A role consists of a predefined set of functions and a set of users authorized to invoke the functions.

DA server and DA agent define a single DA Administrator role named DA_Admin.

Members of the DA_Admin role include:

- da_admin – the administrator account built-in to DA server and DA agent.
- LDAP users – you can bind LDAP users to the DA Administrator role using the **role** command.

See also

- *role* on page 105
- *role* on page 95

Configuring DA for LDAP Authentication

To configure DA server and DA agent for LDAP authentication, modify the `csi.xml` file.

1. Use any text editor to edit the `csi.xml` file.
   - DA server:
     `$SYBASE/DA_15-5/server/`*`instance`*`/security/csi.xml`
   - DA agent:
     `$SYBASE/DA_15-5/agent/`*`instance`*`/security/csi.xml`
2. Configure the **authenticationProvider** parameters to use your LDAP server.

```
<?xml version="1.0" encoding="UTF-8"?>
  <configuration xmlns:config="http://www.sybase.com/csi/2.5/
config"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <authenticationProvider
           name="com.sybase.security.ldap.LDAPLoginModule">
        <options name="ServerType" value="sunone5" />
        <options name="ProviderURL" value="ldap://
ldap.myserver.com:389" />
        <options name="DefaultSearchBase"
value="dc=sybase,dc=com" />
        <options name="AuthenticationScope" value="subtree"
     </authenticationProvider>
</configuration>
```

   where:

Table 16. LDAP Configuration Options

Option	Description
ServerType	(Optional) Specify the type of LDAP server you are connecting to. This value establishes default values for some common configuration properties. Valid values: • sunone5 – SunOne 5.x OR iPlanet 5.*x* • msad2k – Microsoft ActiveDirectory, Windows 2000 • nsds4 – Netscape Directory Server 4.*x* • openldap – OpenLDAP Directory Server 2.*x*
ProviderURL	Specify the URL used to connect to the LDAP server. Default is `ldap://localhost:389`. This default value works if the LDAP server is located on the same machine as your CSI-enabled product and the LDAP server is installed on the default port (389).
DefaultSearchBase	Specify the LDAP search base that is used if no other search base is specified for authentication, role, attribution, and self registration: • dc=<domainname>,dc=<tld> – for example, a machine in sybase.com domain has a search base of dc=sybase,dc=com. • o=<company name>,c=<country code> – for example, this might be o=Sybase,c=us for a machine within the Sybase organization.
AuthenticationScope	Define the credentials for different authentication scopes. Default: onelevel Valid values: • onelevel • subtree

3. Restart DA server and the DA agent.

SSL Security

Replication Server Data Assurance Option includes secure sockets layer (SSL) support. SSL is the standard for securing the transmission of sensitive information, such as credit card numbers and stock trades, over the Internet and other TCP/IP networks.

SSL Overview

The SSL protocol runs above TCP/IP and below application protocols such as remote method invocation (RMI) or Tabular Data Stream™ (TDS).

Before the SSL connection is established, the server and client exchange a series of I/O round trips to negotiate and agree upon a secure encrypted session.

SSL uses certificates issued by certificate authorities (CAs) to establish and verify identities. A certificate is like an electronic passport; it contains all the information necessary to identify an entity, including the public key of the certified entity and the signature of the issuing CA.

See documentation from your third-party SSL security mechanism for instructions for using that software. See also the Internet Engineering Task Force (IETF) Web site for additional information.

An SSL installation requires these items:

- Keystore – a server-side Java KeyStore (JKS). This keystore contains the DA server or DA agent private key.
- Truststore – a client-side Java KeyStore. This contains the certificates of the DA server or DA agent that the client trusts.

Note: Creating a keystore and truststore is not included in this document.

Enabling SSL

Use the **sslconfig** command to add Transport Layer Security to remote method invocation (RMI) and Data Transfer Socket (DTS) communication.

Use SSL for encrypting:

- RMI communication between the Sybase Control Center Data Assurance plug-in and the DA server.
- RMI communication between the DA server and a remote DA agent.
- DTS communication between the DA server and a remote DA agent.

Enabling SSL for SCC Data Assurance Plug-In to DA Server RMI Communication

Configure the DA server and the Sybase Control Center (SCC) Data Assurance plug-in to use SSL to encrypt all RMI network communication.

1. Configure DA server.
   a) Set up RMI client connection to the DA server and issue these commands, for example:

```
sslconfig rmi_server_ssl_enabled true
go
sslconfig keystore location_of_keystore_file
go
```

```
sslconfig keystore_passwd password
go
sslconfig keypair_passwd password
go
```

b) Restart DA server.

2. Configure the SCC Data Assurance plug-in.
   a) Open the `SCC agent-plugin.xml` script for editing:

```
%SYBASE%\SCC-3_2\plugins\DAMAP\agent-plugin.xml
```

   b) Set the DA RMI and Java truststore SSL properties:

      You can add your DA server certificate to your own truststore, or you can add it to the truststore that already exists within SCC, which is located in `$SCC_HOME/services/EmbeddedWebContainer/cacerts`. The default password is `changeit`.

      For example:

```
<properties>
  <set-property property="da.rmi.client.ssl.required" value="myserver" />
   <set-property property="da.rmi.client.debug" value="false" />
   <set-property property="javax.net.ssl.trustStore" value="C:\Sybase
\SCC-3_2\services\EmbeddedWebContainer\cacerts"/>
  <set-property property="javax.net.ssl.trustStorePassword"
value="changeit" />
</properties>
```

      where:

      - **da.rmi.client.ssl.required** – is the host name of the DA server that requires all RMI communication to be encrypted with SSL.

        Note: You can add a comma-delimited list of host names for multiple DA servers.
      - **da.rmi.client.debug** – enables the debug mode. The default is false.
      - **javax.net.ssl.trustStore** – is the location for the truststore file.
      - **javax.net.ssl.trustStorePassword** – is the truststore password.

   c) Restart Sybase Control Center Data Assurance plug-in.

Enabling SSL for DA Server to DA Agent RMI Communication

Configure the DA server and the remote DA agent to use SSL to encrypt all RMI network communication.

1. Configure DA agent.
   a) Establish a command line tool (CLT) session to the DA agent.
   b) Ensure that you have a keystore configured. If you have already enabled SSL for DA server to DA agent DTS communication, you can skip this step.

      To configure a keystore, issue these commands:

```
sslconfig keystore location_of_keystore_file
go
```

```
sslconfig keystore_passwd password
go
sslconfig keypair_passwd password
go
```

c) Set the **rmi_server_ssl_enabled** option to true:

```
sslconfig rmi_server_ssl_enabled true
go
```

d) Restart DA agent.

2. Configure DA server.

a) Ensure that you have a truststore configured. If you have already enabled SSL for DA server to DA agent DTS communication, you can skip this step.

To configure a truststore, issue these commands:

```
sslconfig truststore truststore_file_location
go
sslconfig truststore_passwd password
go
```

b) Set the DA agent host name in the **rmi_client_ssl_required** host list:

```
sslconfig rmi_client_ssl_required host_list
go
```

The **host list** parameter is a comma-delimited list of DA agent hosts that require SSL-enabled DTS.

c) Restart DA server.

Enabling SSL for DA Server to DA Agent DTS Communication

Configure the DA server and the remote DA agent to use SSL to encrypt all DTS network communication.

1. Configure DA agent.

a) Establish a command line tool (CLT) session to the DA agent.

b) Ensure that you have a keystore configured. If you have already enabled SSL for DA server to DA agent RMI communication, you can skip this step.

To configure a keystore, issue these commands:

```
sslconfig keystore location_of_keystore_file
go
sslconfig keystore_passwd password
go
sslconfig keypair_passwd password
go
```

c) Set the **dts_client_ssl_required** option to true:

```
sslconfig dts_client_ssl_required true
go
```

d) Restart DA agent.

2. Configure DA server.

a) Ensure that you have a truststore configured. If you have already enabled SSL for DA server to DA agent RMI communication, you can skip this step.

To configure a truststore, issue these commands:

```
sslconfig truststore truststore_file_location
go
sslconfig truststore_passwd password
go
```

b) Set the DA agent host name in the **dts_client_ssl_required** host list:

```
sslconfig dts_client_ssl_required host_list
go
```

The **host list** parameter is a comma-delimited list of DA agent hosts that require SSL-enabled DTS.

c) Restart DA server.

Password Administration

Configure password policy, enable password encryption, and reset a lost or forgotten password.

Password Policy

The password policy ensures that the DA administrator password is sufficiently secure.

Rules apply to the password policy:

- The default minimum password length is 6.
- The default maximum password length is 30.
- The legal password characters are:
  - 0 – 9
  - A – Z, a – z
  - Hyphen (-) and underscore (_)

You can override the values of password length by adding `da.sec.passwdMinLength` and `da.sec.passwdMaxLength` properties to the *`instance`*`.cfg`.

- DA server – `$SYBASE/DA-15_5/server/`*`instance`*`/`*`instance`*`.cfg`
- DA agent – `$SYBASE/DA-15_5/agent/`*`instance`*`/`*`instance`*`.cfg`

For example, to change the minimum and maximum password lengths to 8 and 20, add:

```
da.sec.passwdMinLength=8
da.sec.passwdMaxLength=20
```

Resetting a Lost or Forgotten Password

Reset a lost or forgotten da_admin password.

Use the **-P** password recovery start-up parameter to reset the password for the da_admin user. You cannot use the parameter to reset passwords of any other account.

1. Stop DA server or DA agent if it is running:
   - If another DA administrator is authenticated using LDAP login that DA administrator can shut down the server, otherwise,
   - Terminate the DA server or DA agent process. This process is platform-dependent.
2. Execute the start-up script:
   - DA server:
     - On Windows 32-bit:
       `%SYBASE%\DA-15_5\server\`*`instance`*`\RUN_`*`instance`*`_32.bat -P`
     - On Windows 64-bit:
       `%SYBASE%\DA-15_5\server\`*`instance`*`\RUN_`*`instance`*`_64.bat -P`
     - On Unix 64-bit:
       `$SYBASE/DA-15_5/server/`*`instance`*`/RUN_`*`instance`*`_64.sh -P`
   - On DA agent:
     - On Windows 32-bit:
       `%SYBASE%\DA-15_5\agent\`*`instance`*`\RUN_`*`instance`*`_32.bat -P`
     - On Windows 64-bit:
       `%SYBASE%\DA-15_5\agent\`*`instance`*`\RUN_`*`instance`*`_64.bat -P`
     - On UNIX 64-bit:
       `$SYBASE/DA-15_5/agent/`*`instance`*`/RUN_`*`instance`*`_64.sh -P`

     where:
     - $SYBASE (on UNIX) or %SYBASE% (on Windows) is the directory in which you installed the Data Assurance Option.
     - *instance* is the name of your DA server instance or DA agent instance.
     - `RUN_`*`instance`*`_32.bat` or `RUN_`*`instance`*`_64.bat` is the start-up script file on Windows.
     - `RUN_`*`instance`*`_64.sh` is the start-up script file on UNIX
     - **-P** is the password recovery start-up parameter

   On start-up, the DA server or the DA agent generates a new da_admin password and writes it to the log file.
3. Obtain the new password:
   - DA server:
     `$SYBASE/DA_15-5/server/`*`instnace`*`/log/da_0.log`

- DA agent:
  ```
  $SYBASE/DA_15-5/agent/instnace/log/da_0.log
  ```

  For example:

  ```
  S 2012-04-03 11:59:27.027 CONTAINER
  FileLoginModule.changePassword@1
  #260 Generated a new password for user "da_admin": "l3Fcza7I"
  ```

  The new password in this example is l3Fcza7I.

4. Log in to DA with the new password.

   Sybase recommends that you now change the da_admin password to one of your own choosing.

Password Encryption

Use the **isql -X** option to encrypt your password when you log in to DA server and DA agent.

You can set the level of password encryption using the **clt_password_encryption_reqd** configuration parameter.

See also

- *config* on page 85
- *config* on page 103

Performance and Tuning

You can tune DA server performance by changing the default values of your server configuration parameters, using the correct comparison options, and changing your deployment.

Deployment Settings

The deployment can have a significant impact on performance. Follow these guidelines when configuring deployment settings for optimal performance:

- Use a distributed environment, with a DA agent installed on a machine that shares a fast Ethernet connection with your database, to minimize the database-to-agent JDBC network traffic.
- Run DA server on a separate machine.

Network Latency

The performance of the overall network, or network latency, is a major factor in system performance. Maximize the network performance between DA server and DA agents. For example, ensure a high network throughput and using a LAN rather than a WAN.

See also

- *config* on page 85

General Settings

Helpful guidelines for improving the overall system performance when executing jobs.

- Choose the right level of comparison for your requirements. For example, schedule row counts for quick daily checks and full-row data checks once a week.
- Whenever possible, schedule comparisons to run after replication has finished.
- Configure the databases to optimize for **select** and **order by** statements.
- Preferably make sure that each table being compared has a single column primary key.
- Run your comparison using the `database_hash` option rather than the `literal` option.
- Generate a summary report instead of a column log for a job. Choosing a column log adds an extra database lookup for column values.
- Configure jobs to abort if there are too many differences.

See also

- *Row Comparison Optimization* on page 124

Row Comparison Optimization

Optimize your row comparisons by fine-tuning various factors such as hash types, column comparison types, and row counts.

When you configure DA server for maximum performance, the bottleneck is often the database server itself; there is a limit to how fast a server can read, sort, and return your data. Use these guidelines to achieve optimal performance for row comparisons:

- Configure non key columns to be compared using the `row_hash` option.
- Configure row comparison using options in this order:
  1. `database_hash`.
  2. `agent_hash` and having a DA agent installed on a machine that shares a fast Ethernet connection with your database.

The `database_hash` option is the fastest row comparison choice, but if you have DA agent installation on the same machine as your DA server, the benefits of using the `database_hash` over `agent_hash` reduces. The key to ideal performance is minimizing the amount of data that is sent from your database to the DA server using the DA agent.

Table 17. Row Comparison Considerations

Factor	Explanation
Hash types	Sybase recommends you to select `database_hash` over `agent_hash`. Hashing each database row is effectively a form of compression; hashed data is smaller, so there is less data to transfer and less data for comparison. The `database_hash` option compresses the data at source database, and offers maximum performance when using remote DA agents and the local DA agent. When you choose `agent_hash`, the DA agent must first receive each row in full before it can hash it, which typically takes longer.

Factor	Explanation
Column compare options	Generally, choose the `row_hash` option over `column_hash` or `literal`. `row_hash` creates a single hash value for all columns in the row. This is the least amount of data DA server can send and compare, while reliably identifying differences between two or more rows. Choose `column_hash` over `literal` to see differences in individual columns. Each column configured with `column_hash` is assigned its own hash value. When using the hash option, the larger the datatype, the greater the advantage.
Row counts	When you have large tables that have no index, or your compareset defines a complex **where** constraint, initial **select count (1)** queries can take a long time to execute. In such cases, for your comparisons to complete more quickly, set **enable_row_count** to false.

Troubleshooting

Determine the cause of problems and apply the recommended solution.

When a DA server or DA agent error occurs, the error log records a message. Review log for diagnostic information about errors encountered by DA server while running comparison jobs.

Approximate Numeric Datatypes Comparison

Problem: Comparison errors are generated for columns that use approximate numeric datatypes.

Possible causes: Approximate numeric datatypes include float, double precision, and real. The exact value of an approximate numeric datatype can vary from one platform to another, and can cause comparison errors such as:

- If a key column includes an approximate numeric datatype, there is no guarantee of the DA server matching the source and target columns. Each failure to do so creates two false differences: one “missing” row in the source database and one “orphaned” row in the target database.
- If a set of columns for comparison include an approximate numeric datatype, there is no guarantee the two matching source and target rows are considered to be equal. Each failure creates a false “inconsistent” difference.

Solution: None.

You may be able to avoid false differences by skipping approximate numeric datatypes when creating column mappings—although this introduces the risk of genuine differences between two approximate numeric datatype columns that may go unnoticed.

Note: Reconciliation cannot fix false differences.

DA Server Out of Memory Errors

Problem: DA server runs out of memory space and exhibits performance issues.

Solution 1: Decrease Comparer Max Concurrent Threads

During a comparison, DA server receives row data from DA agents at different rates, so at any given time, the server may be buffering tens or hundreds of rows for each source or target. If individual rows are large (user-database-table-dependent) and the number of comparisons is sufficiently high (configured by the user), this buffering might cause DA server to run out of memory.

To solve this problem, set **comparer_max_concurrent_threads** to a lower value.

Solution 2: Decrease LOB Fetch Size
The configuration parameter **lob_fetch_size** may be set to a high value.

To solve this problem, set **lob_fetch_size** to a lower value.

Solution 3: Decrease External Sort Max Size
The external sort option uses a large amount of memory. By default, the external sort keeps thousands of rows in memory. While this is not usually a problem, it depends on the size of each row and the simultaneous activity occurring within the same Java Virtual Machine (JVM). For instance, if there are five concurrent comparisons using the DA agent, the memory requirement increases fivefold. Or if the "localagent" is being used, the source and target agents and the compare function are sharing the same JVM memory allocation.

To solve this problem, decrease the number of rows the DA agent stores in the memory by changing the **external_sort_max_size** configuration parameter value.

Solution 4: Increase the Memory Available to DA Server
Global solution: You can address all of the possible causes for out of memory issues described above by configuring DA server to start with more memory. By default, the JVM where the DA server runs uses 512 MB. Increase the value (dependent on the platform and the amount of system memory available) by editing the DA server's **RUN_<instance>.bat** file.

External Sort Option Configuration

Problem: You have configured DA agent to perform external sort, but the database is still performing the sort operation.

The enable sort option is not activated because the number of rows in the table is less than the **external_sort_activate_size** value, which by default is 10 million.

Solution: Decrease the **external_sort_activate_size** to a value less than the number of rows in the source and target tables.

Comparison Fails to Record an Inconsistency

Problem: Job comparison results are not recording inconsistencies in database hash column comparisons.

Solution 1: Increase LOB Fetch Size
The inconsistency might exist in a large object (LOB) column, such as `image`. By default, DA server compares only the first 1024 bytes of LOB columns, so some sections of LOB values are not compared.

To solve this problem, increase the **lob_fetch_size** value.

Solution 2: Increase LOB Fetch Size
The source and target column values might produce the same MD5/SHA/CRC32 hash value.

To solve this problem, use the **literal** option to recompare the rows.

Job Comparison Stops Responding

Problem: A job has successfully executed, but one or more of its comparisons (source or target) stops responding and shows a count of -1, 0 percent progress, and no error message.

Possible cause: DA server is waiting for the row count to complete. If the database table is not optimized for the **select count** query, it may take the database server a long time to execute the row count. While DA server is waiting, the command prompt shows a negative count, 0 percent progress, and no error message for the job.

Try either of these solutions:

- Optimize the database table by creating a new index on the key column so the **select count** query executes faster.
- Alter the job comparison to set **enable_row_count** to false.

Comparison Fails with Stack Space Error

Problem: Job comparison does not complete and shows: `The transaction was aborted because it used too much stack space`.

Possible cause: The compareset table contains hundreds of columns, which results in the DA agent creating a large select query string. This query string can be sufficiently large that the database server does not have enough internal stack space to process the query.

Use any one of these solutions:

- Increase the stack space in the database server using the Adaptive Server stored procedure, **sp_configure**.
- If the DA server configuration parameter **db_hash_ase_ignore_null** is set to false, set it to true; this decreases the size of the select query string.
- Create two new comparesets, each of which compares one distinct half of the database table, then create two new comparisons to replace your current comparison, so the database table is fully compared using the two comparesets in two phases.

Comparisons Against Compressed Tables Fail

Problem: Comparisons fail repeatedly when a compareset points to one or more compressed tables created in Adaptive Server 15.7.

Possible cause: A defect in the Adaptive Server 15.7 compression memory pool that causes the Adaptive Server to enter an error state from which it cannot recover without a restart. The defect occurs when Adaptive Server fails to allocate compression memory. Check the Adaptive Server log for errors.

Either:

- Use **comparer_max_concurrent_threads** and **comparer_max_concurrent_threads** to decrease the number of DA threads that concurrently select from Adaptive Server 15.7, or,
- Increase the size of Adaptive Server compression memory by editing these configuration parameters:
  - **compression memory size**
  - **compression info pool size**

  The amount by which these parameters must be increased varies.

  See the alphabetical listing of configuration parameters in the *Adaptive Server Enterprise System Administration Guide: Volume 1*.

See also

- *Row Comparison Job Commands* on page 53
- *config* on page 85

Obtaining Help and Additional Information

Use the Sybase Getting Started CD, Product Documentation site, and online help to learn more about this product release.

- The Getting Started CD (or download) – contains release bulletins and installation guides in PDF format, and may contain other documents or updated information.
- Product Documentation at *http://sybooks.sybase.com/* – is an online version of Sybase documentation that you can access using a standard Web browser. You can browse documents online, or download them as PDFs. In addition to product documentation, the Web site also has links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, Community Forums/Newsgroups, and other resources.
- Online help in the product, if available.

To read or print PDF documents, you need Adobe Acrobat Reader, which is available as a free download from the *Adobe* Web site.

Note: A more recent release bulletin, with critical product or document information added after the product release, may be available from the Product Documentation Web site.

Technical Support

Get support for Sybase products.

If your organization has purchased a support contract for this product, then one or more of your colleagues is designated as an authorized support contact. If you have any questions, or if you need assistance during the installation process, ask a designated person to contact Sybase Technical Support or the Sybase subsidiary in your area.

Downloading Sybase EBFs and Maintenance Reports

Get EBFs and maintenance reports from the Sybase Web site.

1. Point your Web browser to *http://www.sybase.com/support*.
2. From the menu bar or the slide-out menu, under **Support**, choose **EBFs/Maintenance**.
3. If prompted, enter your MySybase user name and password.
4. (Optional) Select a filter from the **Display** drop-down list, select a time frame, and click **Go**.
5. Select a product.

   Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as an authorized support contact. If

you have not registered, but have valid information provided by your Sybase representative or through your support contract, click **My Account** to add the "Technical Support Contact" role to your MySybase profile.

6. Click the **Info** icon to display the EBF/Maintenance report, or click the product description to download the software.

Sybase Product and Component Certifications

Certification reports verify Sybase product performance on a particular platform.

To find the latest information about certifications:

- For partner product certifications, go to *http://www.sybase.com/detail_list?id=9784*
- For platform certifications, go to *http://certification.sybase.com/ucr/search.do*

Creating a MySybase Profile

MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1. Go to *http://www.sybase.com/mysybase*.
2. Click **Register Now**.

Accessibility Features

Accessibility ensures access to electronic information for all users, including those with disabilities.

Documentation for Sybase products is available in an HTML version that is designed for accessibility.

Vision impaired users can navigate through the online document with an adaptive technology such as a screen reader, or view it with a screen enlarger.

Sybase HTML documentation has been tested for compliance with accessibility requirements of Section 508 of the U.S Rehabilitation Act. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

Note: You may need to configure your accessibility tool for optimal use. Some screen readers pronounce text based on its case; for example, they pronounce ALL UPPERCASE TEXT as initials, and MixedCase Text as words. You might find it helpful to configure your tool to announce syntax conventions. Consult the documentation for your tool.

For information about how Sybase supports accessibility, see the Sybase Accessibility site: *http://www.sybase.com/products/accessibility*. The site includes links to information about Section 508 and W3C standards.

You may find additional information about accessibility features in the product documentation.

Glossary

Definitions of terms related to Replication Server Data Assurance Option.

Adaptive Server – the Sybase version 11.5 and later relational database server. If you choose the RSSD option when configuring Replication Server, Adaptive Server maintains Replication Server system tables in the RSSD database.

DA agent – fetches and compresses data from databases into the DA server.

comparesets – sets of tables and columns that define what is being compared in a particular job.

connection profiles – information required to establish a database connection.

database – a set of related data tables and other objects that are organized and presented to serve a specific purpose.

Data Assurance System Database (DASD) – the DA server database that stores system and configuration settings.

Data Transfer Stream (DTS) – an application protocol used by DA agent during a comparison to stream data.

Hypertext Transfer Protocol (HTTP) – the communications protocol used to connect to Web servers on the Internet or on a local area network.

Java Database Connectivity (JDBC) – is a specification for an application program interface (API) that allows Java applications to access multiple database management systems using Structured Query Language (SQL).

jConnect – the Sybase high-performance JDBC driver.

jobs – a collection of one or more comparison tasks.

inconsistent row – a table row that is present both in primary and replicate databases, but has different values for one or more of the columns being compared.

local area network (LAN) – a system of computers and devices, such as printers and terminals, connected by cabling for the purpose of sharing data and devices.

missing row – a table row that is present in the primary, but not in the replicate database.

orphaned row – a table row that is present in the replicate, but not in the primary database.

parameter – an identifier representing a value that is provided when a procedure executes. Parameter names are prefixed with an @ character in function strings.

primary key – a set of table columns that uniquely identifies each row.

quoted identifiers – object names that contain special characters such as spaces and non-alphanumeric characters, start with a character other than alphabet, or correspond to a reserved word and need to be enclosed in quote (single or double) characters to be parsed correctly.

reconciliation – the process of updating the target database tables to match with the source database tables.

replication – a process by which the changes to the data in one database—including creation, updating, and deletion of records—are also applied to the corresponding records in another database.

Remote Method Invocation (RMI) – is a remote procedure call used for communication between DA server and DA agents.

Replication Server – the Sybase server program that maintains replicated data, typically on a LAN, and processes data transactions received from other Replication Servers on the same LAN or on a WAN.

Replication Server Data Assurance Option – the Sybase server program that compares row data and schema between two or more Adaptive Server databases, and reports discrepancies.

Replication Server System Database (RSSD) – the Adaptive Server database containing a Replication Server system tables. You can choose whether to store Replication Server system tables on the RSSD or the SQL Anywhere® ERSSD.

row comparison job – is a job used for row comparison.

schema – the structure of the database.

schema comparison job – is a job used for comparing database object schemas.

Tabular Data Stream™ (TDS) – is an application protocol by which Open Client™ and Open Server™ applications exchange information.

wide area network (WAN) – a system of local area networks connected together with data communication lines.

Index

A

abort job command 71
agent commands 33
agent_access_timeout_mins 86
agent_client_ctx_timeout_secs 86
agent_max_queue 86
agent_max_request_queue 86
alter agent command 33
alter compareset command 46
alter connection command 40
alter job command 53
alter schema job command 65
auto_recon_stmt_batch_size 87

C

clt_password_encryption_reqd 87
column comparison problem, database hash 128
command line tool 33
commands
    abort job 71
    alter agent 33
    alter compareset 46
    alter connection 40
    alter job 53
    alter schema job 65
    config 85, 103
    create agent 34
    create backup 83
    create compareset 48
    create connection 41
    create job 59
    create schema job 68
    depend agent 34
    depend compareset 51
    depend connection 43
    disable job 71
    drop agent 35
    drop backup 83
    drop compareset 51
    drop connection 44
    drop history 72
    drop job 64
    drop schema job 70
    enable job 72
    import job 78
    monitor job 73
    restore backup 84
    role 95, 105
    run job 74
    show agent 35
    show agent connection 36
    show agent dts 36
    show agent system 38
    show agent task 38
    show backup 84
    show compareset 52
    show connection 44, 106
    show dts 106
    show history 75
    show job 65
    show jvm 96
    show reconcile 76
    show report 77
    show schema job 70
    show system 97, 107
    show task 108
    sslconfig 97, 108
    test agent 39
    test connection 45
    trace 100, 110
    truncate backup 84
    truncate history 77
    user name 95, 105
    version 100, 111
comparer_max_concurrent_threads 87
comparer_recently_finished_ttl_secs 88
comparer_retry_delay_threshold_secs 88
comparer_retry_max_keys_per_clause 88
comparer_retry_min_fill_percent 89
comparer_retry_min_fill_percent_literal 89
comparer_retry_min_keys_in_range 88
comparesets 7
    commands 45
comparison options 9
comparison strategies 8
comparison task 7
config command 85, 103
configuration parameters
    agent_access_timeout_mins 85

agent_client_ctx_timeout_secs 85
agent_max_queue 85
auto_recon_stmt_batch_size 85
clt_password_encryption_reqd 85, 103
comparer_max_concurrent_threads 85
comparer_recently_finished_ttl_secs 85
comparer_retry_delay_threshold_secs 85
comparer_retry_max_keys_per_clause 85
comparer_retry_min_fill_percent 85
comparer_retry_min_fill_percent_literal 85
comparer_retry_min_keys_in_range 85
db_connection_retry_interval 85
db_connection_retry_times 85
db_hash_ase_algorithm 85
db_hash_ase_ignore_null 85
db_hash_ase_using_option 85
default_column_compare_mode 85
enable_report_generator 85
external_sort_activate_size 85
external_sort_compress_file 85
external_sort_max_file 85
external_sort_max_size 85
external_sort_max_thread 85
lob_fetch_size 85
recon_tran_max_stmts 85
text_report_max_column_width 85
text_report_max_line_length 85
connection profile commands 39
conventions
style 1
syntax 1
create agent command 34
create backup command 83
create compareset command 48
create connection command 41
create job command 59
create schema job command 68

D

DA server commands 33
DASD 12
commands 83
Data Assurance agent
local agent 6
remote agent 6
standalone agent 6
database connections 7, 39
database hash column comparison problem 128
db_connection_retry_interval 90
db_connection_retry_times 90
db_hash_ase_algorithm 90
db_hash_ase_ignore_null 90
db_hash_ase_using_option 91
default_column_compare_mode 90
deleting
backup 31
job history 31
depend agent command 34
depend compareset command 51
depend connection command 43
disable job command 71
drop agent command 35
drop backup command 83
drop compareset command 51
drop connection command 44
drop history command 72
drop job command 64
drop schema job command 70

E

enable job command 72
enable_report_generator 91
external sort problem 128
external_sort_activate_size 92
external_sort_compress_file 92
external_sort_max_file 92
external_sort_max_size 91
external_sort_max_thread 91

F

file_output_encoding 92

G

getting started 15

I

import job command 78

J

job comparison planning 8
job history 12
job option 9

job reports 12
jobs
    row comparison 7
    schema comparison 7

K

Kerberos
    agent configuration 113
    network-based security 113
    security 113
    security mechanism 113
key options 9

L

LDAP authentication 114
    users 115
LDAP authentication server
    authenticationProvider 115
    csi.xml 115
limitation 50
lob_fetch_size 92
local agent 6

M

managing jobs 71
map all limitations 50
monitor job command 73

O

other commands 85
out of memory error, DA server 127

P

password
    password length 120
    policy 120
Password
    reset 121
password encryption 122
performance and tuning 123
    general settings 123
    optimizing row comparisons 124

R

recon_tran_max_stmts 93
reconciliation 8
    automatic 24
    database table 24
    script 24
remote DA agent commands 103
    show connection 106
    show dts 106
    show jvm 107
    show task 108
    sslconfig 108
    trace 110
Replication Server Data Assurance Option
    integration 13
    overview 3
    system architecture 3
reports 12
reserved words
    DA server 101
    remote DA agent 112
restore backup command 84
role
    DA_Admin 114
    LDAP users 114
role command 95, 105
row comparison job commands 53
run job command 74

S

schema comparison job commands 65
secure sockets layer 116
security
    LDAP 114
    password administration 120
    password policy 120
    password resetting 121
show agent command 35
show agent connection command 36
show agent dts command 36
show agent system command 38
show agent task command 38
show backup command 84
show compareset command 52
show connection command 44, 106
show dts command 106
show history command 75
show job command 65
show jvm command 96
show reconcile command 76
show report command 77

show schema job command 70
show system command 97, 107
show task command 108
SSL
    DA server and remote DA agent 118, 119
    DA server and Sybase Control Center for Data Assurance 117
    Data Transfer Socket (DTS) 117
    DTS protocols 119
    emote method invocation (RMI) 117
    overview 117
    Remote Method Invocation (RMI) 117
    RMI protocols 117, 118
    Tabular Data Stream (TDS) 117
SSL (secure sockets layer) 116
sslconfig command 97, 108
standalone agent 6
Sybase Common Security Infrastructure (CSI) 115
system database 12

T

task flow
    back up 31
    deleting data files 31
    deleting log files 31
    import job 30
    job 29
    restore 31
    schema job 29
    server configuration parameters 31
terms
    Replication Server Data Assurance Option 135
test agent command 39
test connection command 45
text_report_max_column_width 93
text_report_max_line_length 93
trace command 100, 110
troubleshoot 127
    allocate compression memory 130
    approximate numeric datatype issue 127
    compressed tables 130
    database hash column comparison problem 128
    external sort problem 128
    job comparisons not responding 129
    out of memory error 127
    stack space error 129
truncate backup command 84
truncate history command 77

U

user task flow example 15, 22
using map all 50

V

version
    DA server 100
    remote DA agent 111
version command 100, 111