Users Guide

SAP Replication Server® Data Assurance Option 15.7.1 SP200

DOCUMENT ID: DC01636-01-1571200-01
LAST REVISED: March 2014

Contents

Conventions

Learn about the style and syntax conventions used in SAP® 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, SAP® 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
```

- SAP® Adaptive Server® Enterprise (SAP® ASE) database objects are case-sensitive. Use the correct case for table names when you specify SAP Adaptive Server database objects in your DA configuration.
- Oracle database objects are case-sensitive. Use the correct case for table names when you specify Oracle database objects in your DA configuration.

SAP Replication Server Data Assurance Option

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

SAP Replication Server Data Assurance Option is a scalable, high-volume, and configurable data comparison product, allowing 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. SAP 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.

SAP Replication Server Data Assurance Option allows large tables to be split into multiple partitions for comparing data in parallel. You can also compare row data between any combinations of SAP® Adaptive Server® Enterprise (SAP® ASE) and SAP® IQ, SAP HANA®, IBM DB2 Universal Database(UDB), Microsoft SQL Server, or Oracle databases in a heterogeneous comparison environment.

SAP 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

An SAP 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.

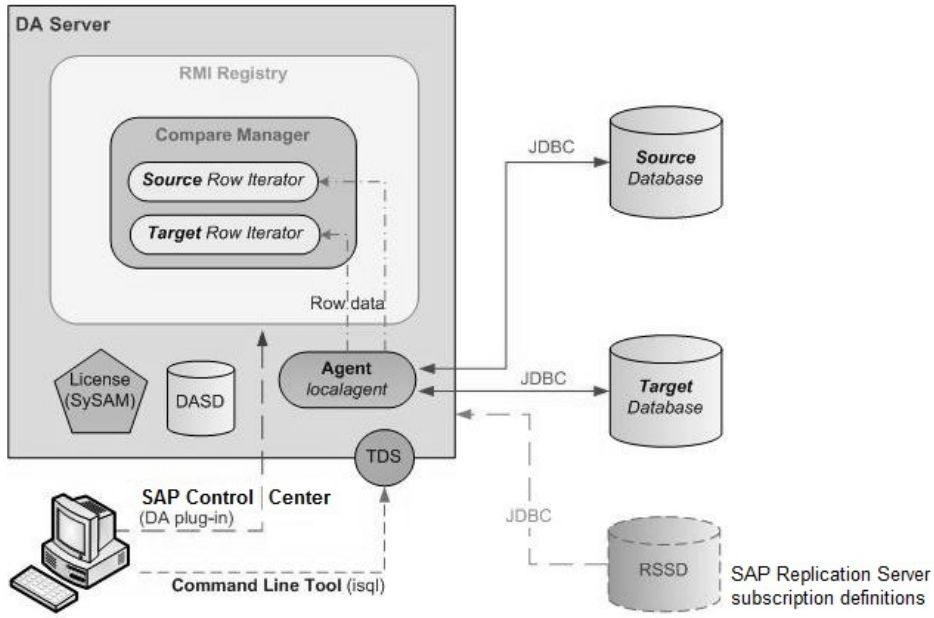
Single-Server Deployment

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

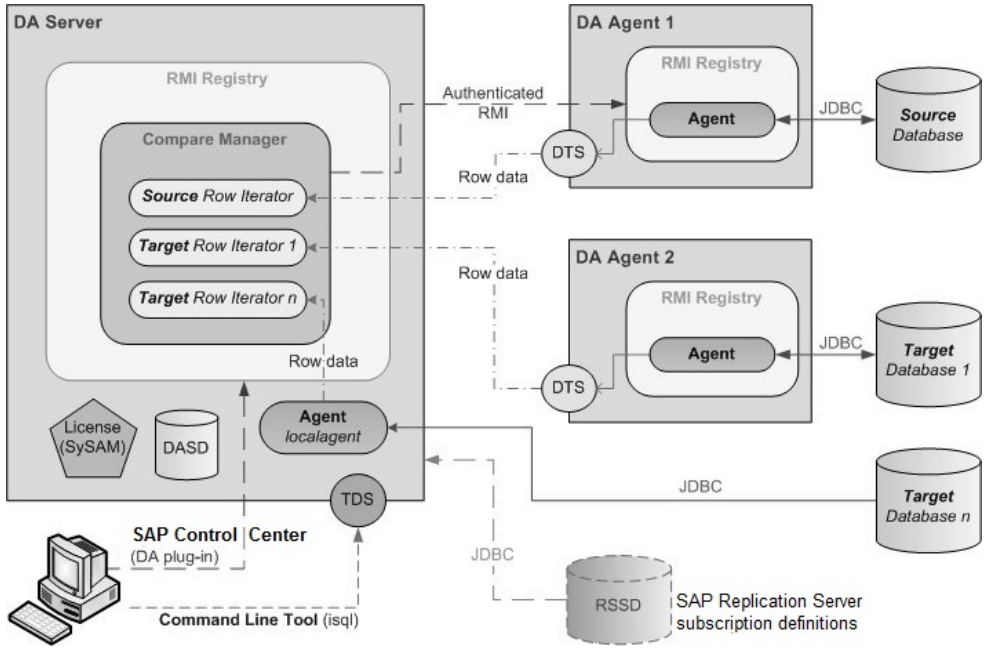


Distributed Deployment

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. We recommend you to 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 *SAP Adaptive Server Enterprise > Utility Guide > Using Interactive isql from the Command Line*.
- The SAP® Control Center (SAP SCC) Data Assurance plug-in. See *SAP Control Center for Data Assurance Documentation*.

DA server integrates with SAP Replication Server to automatically generate DA server jobs that are based on the information in the RSSD. SAP 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 14
- *Integration with SAP Replication Server* on page 16
- *import job* on page 126

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 Java Database Connectivity (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 50

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 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 58

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 73

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 an 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 92
- *create schema job* on page 109
- *Creating a Job* on page 43
- *Creating a Schema Job* on page 43

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 190

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

The `database_hash` comparison option is supported only for Adaptive Server-to-Adaptive Server comparisons.

As the Adaptive Server **hashbytes** function does not accept large objects (LOB) datatypes as a parameter, neither does the `database_hash` column comparison option support those datatypes (such as `text`, `image`, and `unitext`).

Adaptive Server supports the **hashbytes** function only 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.

Database Hash and Default Column Compare Modes

Use the **column_option_helper_visit_db** configuration option to allow DA to verify and, if necessary, amend the default column compare modes for a comparison, when the **hash_type** comparison option is set to `database_hash`.

When the **column_option_helper_visit_db** is false, the default column compare mode is set to the **default_column_compare_mode** parameter value.

When the **column_option_helper_visit_db** is true, DA connects to the database and checks whether the **default_column_compare_mode** is appropriate and legal for each column. DA overwrites the **default_column_compare_mode**, and sets the column compare to `literal`, if the Java SQL column type is:

- `LONGVARCHAR`, `LONGVARBINARY`, or `LONGNVARCHAR`
- `DOUBLE`, `FLOAT`, or `REAL`
- `NUMERIC` or `DECIMAL` and the scale is greater than zero

Large Objects Support

All LOB types use a "first N bytes" parameter, where *N* is a configurable with **lob_fetch_size** parameter. If the number of bytes in the LOB column is less than "N", the entire column value is used.

See also

- *config* on page 133

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 Options

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 92
- *Creating a Job* on page 43

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 119

Data Partitions

Data partitions allows you to split large tables into logical partitions.

For maximum performance, run each logical partition in parallel on a database with multiple engines. You can also run the data partitions in smaller groups, if DA is configured to run fewer comparisons at the same time.

Note: When you run a comparison with a new compareset for the first time, DA runs the comparison using a single partition, and collects partition information for subsequent runs.

The SQL **where** clause on the compareset key columns defines the upper and lower partition boundaries. The boundary key values are sampled for a compareset when a comparison that uses it is run for the first time. These boundary samples are stored in the DASD.

On subsequent runs, any comparison that uses the same compareset will use those boundary samples to split a table into many logical partitions.

For example, if there are 16 boundary samples stored and the **NUM_PARTITIONS** comparison option is set to 2, only the "middle" boundary sample is required to split the table into two logical partitions.

Each time a comparison is run, DA collects new boundary samples for its compareset. The new boundary samples for the compareset in use are updated in the DASD.

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
- Data partition boundary samples for each compareset

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

See also

- *create backup* on page 131
- *Backing Up and Restoring the DASD* on page 45

Heterogeneous Comparison

A heterogeneous comparison environment means two or more of the databases in use are of different vendors.

To use heterogeneous comparison:

- Configure DA to use the JDBC drivers for Oracle, Microsoft SQL Server, IBM DB2 UDB, and SAP HANA database instances, if your heterogeneous comparison include these databases.
- Create connections for each database supported.
- Set the **hash_type** comparison option to `agent_hash`.

Table 1. Databases Supported

Database Name	Version
Oracle	11g, 10g
SAP IQ	15.4, 15.3
SAP HANA	1.0
IBM DB2 UDB	9.5, 9.7, 10.1
Microsoft SQL Server	2008, 2008 R2, 2012

You can compare data between any source and target database combinations. For example:

- A homogeneous comparison IBM DB2 UDB to IBM DB2 UDB
- An SAP Adaptive Server source database and an IBM DB2 UDB target database in a heterogeneous comparison
- An Oracle source database with multiple target databases, for example, both SAP Adaptive Server and SAP IQ
- An SAP Adaptive Server source database and an SAP HANA target database

Default Column Compare Modes for Comparisons

A comparison uses a set of options to compare tables defined in a compareset.

Use the **default_column_compare_mode** configuration option to set the default column comparison option for a comparison. This configuration option is not used, if you explicitly define all column compare modes when you create the comparison.

The default column compare mode for compareset key columns is `literal`, regardless of the column type.

All compareset non-key columns default to the **default_column_compare_mode** value.

Rules that apply to select the default column compare mode for compareset non-key columns:

- When the **config** option **default_column_compare_mode** is `literal`, the column compare mode is `literal`.
- When the **config** option **default_column_compare_mode** is `column_hash`, and if the **hash_type** comparison option is `agent_hash`, the column compare mode is `column_hash`.
- When the **config** option **default_column_compare_mode** is `row_hash`, and if the **hash_type** comparison option is `agent_hash`, the column compare mode is `row_hash`.

When the **default_column_compare_mode** configuration option is `row_hash` or `column_hash`, and the **hash_type** comparison option is set to `database_hash`, a different set of rules apply. See *Database Hash and Default Column Compare Modes* on page 11.

See also

- *Comparison and Reconciliation Strategies* on page 8
- *config* on page 133

Integration with SAP Replication Server

SAP Replication Server Data Assurance Option integrates with SAP Replication Server to automatically import DA server jobs that are based on the information in the 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: SAP Replication Server Data Assurance Option does not support database replication definitions.

Jobs that are imported from SAP 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 126
- *Importing a Job from SAP Replication Server* on page 44

Getting Started

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

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

Single-Server Deployment

SAP 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 DA server with the local embedded agent. No remote DA agents are used.

Table 2. Deployment Summary

Component Name	Machine Name	Port Numbers
DA server	mars	• 4500 – RMI • 4501 – TDS • 4503 – DASD
SAP Adaptive Server Enterprise	venus	5000 – server
SAP 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 SAP Adaptive Server databases installed on *venus* and *pluto*. The user sa and null password are defaults for SAP 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 num_partitions to 2
      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
 NUM_PARTITIONS         2
 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)
```

18. To view report of an individual job history, specify the History_ID number for that job:

```
show report authors_job1 3
go
 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
```

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

SAP 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 DA server and two remote DA agents. The local DA agent is not used.

Table 3. 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
SAP Adaptive Server Enterprise	venus	5000 – server
SAP 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 SAP Adaptive Server database installed on *venus*, and *agent_pluto* connects to the SAP 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 4. Deployment Summary

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


COMPARISON TARGET RECONCILIATION SCRIPT
---------- ------
------------------------------------------------------------------------
--------------------------------
```

```
cmp          0    C:\Sybase\DA-15_5\server\instance\data
\authors_job3\2012-10-05\09.11.28.585\cmp_authors3_T_recon_ins.sql
                  C:\Sybase\DA-15_5\server\instance\data
\authors_job3\2012-10-05\09.11.28.585\cmp_authors3_T_recon_upd.sql
                  C:\Sybase\DA-15_5\server\instance\data
\authors_job3\2012-10-05\09.11.28.585\cmp_authors3_T_recon_del.sql


(0 rows affected)
```

8. Execute the **show report** with the job history ID.

```
show report authors_job3 1
go
```

The return result is:

```
 FILE         SERVER PATH
------------
-------------------------------------------------------------------------
---------------------
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)
```

9. Execute the `cmp_authors3_T_recon_ins` **insert** 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_ins.sql"

Password:
(1 row affected)
```

An example of the reconciliation script:

```
--
-- Replication Server Data Assurance Option/15.7.2/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 1 difference(s).
-- Missing: 1 (insert)
--
-- 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
```

Heterogeneous Comparison Configuration

Before performing a heterogeneous comparison, configure DA to use the JDBC driver for IBM DB2 UDB, Microsoft SQL Server, Oracle and SAP HANA databases.

Use the SAP® jConnect™ for JDBC driver that ships with DA server and DA agent to connect to SAP IQ, which does not require any further configuration.

Note: DA does not include the JDBC driver JAR files for IBM DB2 UDB, Microsoft SQL Server, Oracle or SAP HANA databases.

Configuring DA to Use the IBM DB2 UDB JDBC Driver

To use an IBM DB2 UDB database in a comparison, configure the DA server and the DA agent to use the `db2jcc4.jar` file.

1. Download the `db2jcc4.jar` JDBC driver for your database version from the IBM Web site.
2. Copy the `db2jcc4.jar` file into the DA library folder:
   - On Windows – `%SYBASE%\DA-15_5\server\lib\`
   - On UNIX – `$SYBASE/DA-15_5/server/lib/`

   where `%SYBASE%` (Windows) or `$SYBASE` (UNIX) is the directory in which you installed the Data Assurance Option, and `lib` is the library folder of your DA server instance.
3. Restart the DA server for the settings to take effect.

   Follow the same steps to configure a DA agent to use the IBM DB2 UDB JDBC driver JAR file.

Configuring DA to Use the Microsoft SQL Server JDBC Driver

To use a Microsoft SQL Server database in a comparison, configure the DA server and the DA agent to use the `sqljdbc4.jar` file.

1. Download the `sqljdbc4.jar` JDBC driver for your database version from the Microsoft Web site.
2. Copy the `sqljdbc4.jar` file into the DA library folder:
   - On Windows – `%SYBASE%\DA-15_5\server\lib\`
   - On UNIX – `$SYBASE/DA-15_5/server/lib/`

where `%SYBASE%` (Windows) or `$SYBASE` (UNIX) is the directory in which you installed the Data Assurance Option, and `lib` is the library folder of your DA server instance.

3. Restart the DA server for the settings to take effect.

   Follow the same steps to configure a DA agent to use the Microsoft SQL Server JDBC driver JAR file.

Configuring DA Server to Use the Oracle JDBC Driver

Configure the DA server to use the `ojdbc.jar` Oracle JDBC driver JAR file, before performing a heterogeneous comparison.

1. Obtain the JDBC driver JAR file from your Oracle installation directory.

   For example, in Oracle 10.2.0, the JAR file is in:
   - On Windows – `%ORACLE%\product\10.2.0\server\jdbc\lib\ojdbc14.jar`
   - On UNIX – `$ORACLE/product/10.2.0/server/jdbc/lib/ojdbc14.jar`
2. Copy the `ojdbc14.jar` file into the DA library folder:
   - On Windows – `%SYBASE%\DA-15_5\server\lib\`
   - On UNIX – `$SYBASE/DA-15_5/server/lib/`

   where `%SYBASE%` (Windows) or `$SYBASE` (UNIX) is the directory in which you installed the Data Assurance Option, and `lib` is the library folder of your DA server instance.

   Note: You need not copy the driver file, if the JAR file is available to the DA server on the same file system.
3. Restart the DA server for the settings to take effect.

   Follow the same steps to configure a DA agent to use the Oracle JDBC driver JAR file.

See also

- *create connection* on page 58
- *Data Comparison Scenario 1: SAP Adaptive Server to SAP IQ and Oracle Databases* on page 33

Configuring DA Server to Use the SAP HANA JDBC Driver

Configure the DA server to use the `ngdbc.jar` JDBC driver JAR file, before performing a heterogeneous comparison.

1. Download the type4 JDBCdriver JAR file from the SAP® Web site for your HANA DB version.
2. Copy the `ngdbc.jar` file into the DA library folder:
   - On Windows – `%SYBASE%\DA-15_5\server\lib\`
   - On UNIX – `$SYBASE/DA-15_5/server/lib/`

   where `%SYBASE%` (Windows) or `$SYBASE` (UNIX) is the directory in which you installed the Data Assurance Option, and `lib` is the library folder of your DA server instance.
3. Restart the DA server for the settings to take effect.

   Follow the same steps to configure a DA agent to use the SAP HANA JDBC driver JAR file.

Heterogeneous Data Comparison Scenarios

Learn about heterogeneous comparison using databases from different vendors.

Data Comparison Scenario 1: SAP Adaptive Server to SAP IQ and Oracle Databases

Perform a heterogeneous comparison using SAP Adaptive Server, Oracle, and SAP IQ databases.

This example uses a DA server with a local agent and two remote DA agents installed on different machines with an SAP Adaptive Server, a SAP IQ, and an Oracle database connection.

Table 5. Deployment Summary

Component Name	Machine Name	Port Numbers
DA server	mars	• 6500 – RMI • 6501 – TDS • 6503 – DASD
DA agent	jupiter	• 6500 – RMI • 6501 – TDS • 6502 – DTS

Component Name	Machine Name	Port Numbers
DA agent	saturn	• 6500 – RMI • 6501 – TDS • 6502 – DTS
SAP Adaptive Server database	mars	5000 – server
Oracle database	jupiter	1521 – server
SAP IQ database	saturn	2638 – server

1. Add the JDBC driver JAR file for an Oracle database to the DA classpath, before starting the DA server or the DA agent.
2. Follow step 1 to step 6 of the distributed deployment example to start the DA server instance and connect to the DA agents.
3. Create database connections for the new DA agents.

   In this example, `pubs2` database is replicated on Oracle and SAP IQ databases. The DA server's local agent on `mars` points to the SAP Adaptive Server database on `mars`, while the agents on `jupiter` and `saturn` point to their respective Oracle and SAP IQ databases:

```
create connection conn_mars
    set type=ase
    and set agent=localagent
    and set host=mars
    and set port=5000
    and set user=sa
    and set database=pubs2
    and set password=password
    go
```

```
create connection conn_jupiter
    set type=oracle
    set agent=agent_jupiter
    and set host=jupiter
    and set port=1521
    and set user=system
    and set database=pubs2
    and set password=password
    go
```

```
create connection conn_saturn
    set type=iq
    set agent=agent_saturn
    and set host=saturn
    and set port=2638
    and set user=DBA
    and set database=pubs2
```

```
    and set password=password
    go
```

4. View the newly created database connections:

```
show connection
go
```

```
show connection conn_mars
go
```

```
show connection conn_jupiter
go
```

```
show connection conn_saturn
go
```

5. Test the database connections:

```
test connection conn_mars
go
```

```
test connection conn_jupiter
go
```

```
test connection conn_saturn
go
```

6. Create a compareset to map the entire source table.
   This example compares the SAP Adaptive Server and the Oracle data:

```
create compareset authors_demo4
  with
    source conn_mars dbo authors s
    target conn_jupiter SCOTT 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
```

7. View the compareset:

```
show compareset authors_demo4
go
```

8. Create a job.
   This job creates a single comparison that uses the compareset defined in step 6. It compares all rows in the `authors` table:

```
create job authors_job4
        set MAX_CONCURRENT_COMPARISONS = 100
    add comparison cmp_authors4
        set COMPARESET=authors_demo4
        and set NUM_PARTITIONS to 1
        and set ENABLE_ROW_COUNT to false
```

```
        and set COMPARE_MODE to row_compare
        and set HASH_TYPE to AGENT_HASH
    go
```

Note: Set the **hash_type** comparison option to `agent_hash` for heterogeneous comparison. The `database_hash` comparison option is used only for SAP Adaptive Server-to- SAP Adaptive Server comparisons.

9. Execute the job to compare the data:

```
run job authors_job4
go
```

DA compares equivalent values stored in distinct datatypes accurately. For example, the value 1 stored in a SAP IQ `NUMERIC` column is equivalent to the value 1 stored in an Oracle `NUMBER` column, and to 1.0 stored in an `ASE FLOAT` column.

See also

- *Heterogeneous Comparison* on page 14
- *create connection* on page 58
- *Configuring DA Server to Use the Oracle JDBC Driver* on page 32
- *Configuring DA Server to Use the SAP HANA JDBC Driver* on page 32

Data Comparison Scenario 2: SAP Adaptive Server to Microsoft SQL Server

Perform a heterogeneous comparison using SAP Adaptive Server and Microsoft SQL Server databases.

Prerequisites

Before starting the DA agent, add the JDBC driver JAR file for a Microsoft SQL Server to the DA classpath.

Task

This example uses a DA server local agent connecting to an SAP Adaptive Server, and a DA agent connecting to a Microsoft SQL Server.

Table 6. Deployment Summary

Component Name	Machine Name	Port Numbers
DA server	mars	• 6500 – RMI • 6501 – TDS • 6503 – DASD

Component Name	Machine Name	Port Numbers
DA agent	pluto	• 6500 – RMI • 6501 – TDS • 6502 – DTS
SAP Adaptive Server database	mars	5000 – server
Microsoft SQL Server database	pluto	1433 – server

1. Start the DA server instance named mars:

   ```
   $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`. On UNIX or Linux platforms, the file is named `RUN_instance_64.sh`.

2. Start the DA agent instance on the machine named pluto:

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

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

```
$SYBASE/OCS-15_0/bin/isql -S mars:6501 -U da_admin -P password -w 250
```

4. Create a DA agent connection that connects to the Microsoft SQL Server:

   ```
   create agent agent_pluto
       set host=pluto
       and set port=6500
       and set user=da_admin
       and set password=password
   go
   ```

5. View the newly created DA agent:

   ```
   show agent agent_pluto
   go
   ```

6. Test connection settings for the DA agent:

   ```
   test agent agent_pluto
   go
   ```

7. Create database connections for the new DA agent.
   In this example, the `pubs2` database is replicated on the Microsoft SQL Server database. The DA server's local agent on `mars` points to the SAP Adaptive Server database on `mars`, while the agent on `pluto` points to the Microsoft SQL Server database:

   ```
   create connection conn_mars
       set type=ase
   ```

```
    and set agent=localagent
    and set host=mars
    and set port=5000
    and set user=sa
    and set database=pubs2
    and set password=password
    go
```

```
create connection conn_pluto
   set type=MSSQL
   and set agent=agent_pluto
   and set host=pluto
   and set port=1433
   and set database=pubs2
   and set user=steve
   and set password=ibmste11
   go
```

8. View the newly created database connections:

```
show connection
go
```

```
show connection conn_mars
go
```

```
show connection conn_pluto
go
```

9. Test the database connections:

```
test connection conn_mars
go
```

```
test connection conn_pluto
go
```

10. Create a compareset to map the entire source table.

    This example compares the SAP Adaptive Server and the Microsoft SQL Server data:

```
create compareset authors_demo4
  with
    source conn_mars 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
```

11. View the compareset:

```
show compareset authors_demo4
go
```

12. Create a job.

This job creates a single comparison that uses the compareset defined in step 10. It compares all rows in the `authors` table:

```
create job authors_job4

        add comparison cmp_authors4
        set COMPARESET=authors_demo4
        and set NUM_PARTITIONS to 1
        and set ENABLE_ROW_COUNT to false
        and set COMPARE_MODE to row_compare
        and set HASH_TYPE to AGENT_HASH
    go
```

Note: Set the **HASH_TYPE** comparison option to `agent_hash` for heterogeneous comparison. The `database_hash` comparison option is used only for comparisons between SAP Adaptive Server.

13. Execute the job to compare the data:

```
run job authors_job4
go
```

DA compares equivalent values stored in distinct datatypes accurately.

Data Comparison Scenario 3: SAP Adaptive Server to IBM DB2 UDB

Perform a heterogeneous comparison using SAP Adaptive Server and IBM DB2 UDB databases.

Prerequisites

Before starting the DA agent, add the JDBC driver JAR file for an IBM DB2 UDB to the DA classpath.

Task

This example uses a DA server local agent connecting to an SAP Adaptive Server and a DA agent connecting to an IBM DB2 Universal database.

Table 7. Deployment Summary

Component Name	Machine Name	Port Numbers
DA server	mars	• 6500 – RMI • 6501 – TDS • 6503 – DASD

Component Name	Machine Name	Port Numbers
DA agent	neptune	• 6500 – RMI • 6501 – TDS • 6502 – DTS
SAP Adaptive Server database	mars	5000 – server
IBM DB2 Universal Database	neptune	5001– server

1. Start the DA server instance named mars:

   ```
   $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`. On UNIX or Linux platforms, the file is named `RUN_instance_64.sh`.

2. Start the DA agent instance on the machine named neptune:

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

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

```
$SYBASE/OCS-15_0/bin/isql -S mars:6501 -U da_admin -P password -w 250
```

4. Create a DA agent connection that connects to the IBM DB2 Universal Database:

   ```
   create agent agent_neptune
       set host=neptune
       and set port=6500
       and set user=da_admin
       and set password=password
   go
   ```

5. View the newly created DA agent:

   ```
   show agent agent_neptune
   go
   ```

6. Test connection settings for the DA agent:

   ```
   test agent agent_neptune
   go
   ```

7. Create database connections for the new DA agent.

   In this example, the `pubs2` database is replicated on the IBM DB2 UDB. The DA server's local agent on `mars` points to the SAP Adaptive Server database on `mars`, while the agent on `neptune` points to the IBM DB2 UDB:

   ```
   create connection conn_mars
       set type=ase
   ```

```
    and set agent=localagent
    and set host=mars
    and set port=5000
    and set user=sa
    and set database=pubs2
    and set password=password
    go
```

```
create connection conn_neptune
   set type=UDB
   and set agent=agent_neptune
   and set host=neptune
   and set port=5001
   and set database=PUBS2
   and set user=JOHN
   and set password=mssqt12
   go
```

8. View the newly created database connections:

```
show connection
go
```

```
show connection conn_mars
go
```

```
show connection conn_neptune
go
```

9. Test the database connections:

```
test connection conn_mars
go
```

```
test connection conn_neptune
go
```

10. Create a compareset to map the entire source table.

This example compares the SAP Adaptive Server and the IBM DB2 UDB data:

```
create compareset authors_demo4
  with
    source conn_mars dbo authors s
    target conn_netpune DB2INST1 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
```

11. View the compareset:

```
show compareset authors_demo4
go
```

12. Create a job.

This job creates a single comparison that uses the compareset defined in step 10. It compares all rows in the `authors` table:

```
create job authors_job4
        add comparison cmp_authors4
        set COMPARESET=authors_demo4
        and set NUM_PARTITIONS to 1
        and set ENABLE_ROW_COUNT to false
        and set COMPARE_MODE to row_compare
        and set HASH_TYPE to AGENT_HASH
    go
```

Note: Set the **HASH_TYPE** comparison option to `agent_hash` for heterogeneous comparison. The `database_hash` comparison option is used only for comparisons between SAP Adaptive Server.

13. Execute the job to compare the data:

```
run job authors_job4
go
```

DA compares equivalent values stored in distinct datatypes accurately.

Administrative Tasks

Create comparison jobs, import jobs from SAP 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 50
- *create connection* on page 58
- *create compareset* on page 73
- *create job* on page 92
- *run job* on page 117
- *monitor job* on page 115
- *show history* on page 119

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 50
- *create connection* on page 58
- *create schema job* on page 109
- *run job* on page 117
- *show history* on page 119

Importing a Job from SAP Replication Server

Import a job based on predefined table replication definitions and subscriptions from Replication Server System Database(RSSD) 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 SAP 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 50
- *create connection* on page 58
- *import job* on page 126
- *alter job* on page 85
- *run job* on page 117
- *monitor job* on page 115

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 133

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 131
- *show backup* on page 132
- *restore backup* on page 132

Deleting Data and Log Files

Delete job history and DASD backup.

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

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

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

See also

- *drop history* on page 114
- *truncate history* on page 125
- *drop backup* on page 131
- *truncate backup* on page 133

Data Assurance Command Line Tool

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

Wildcard Characters

Wildcard characters allow command line tool (CLT) commands to return a subset of results.

Filter results using:

- An asterisk (*) – for zero or more characters.
- A question mark (?) – for exactly one character.

You can use both wildcards multiple times in a single command.

Table 8. Wildcard Examples

Command	Description
`show job *`	Shows all jobs. Same as **show job** command.
`show job a*`	Shows all jobs with names beginning with "a". The shortest match is the name “a”.
`show job *z`	Shows all jobs with names ending with "z". The shortest match is the name “z”.
`show job a*z`	Shows all jobs with names beginning with "a" and ending with "z". The shortest match is the name “az”.
`show job ***`	Shows all jobs. The additional wildcards are ignored.
`show job a?`	Shows all jobs with names beginning with "a" that are two characters in length. For example, “ab”, “a1”, and “a2”, but not “a” or “a12”.
`show job ???`	Shows jobs with names that are three characters in length.
`show job a?m*z`	Shows all jobs with names beginning with "a", followed by any character, followed by "m", followed by zero or more characters and ending with "z". For example, “almaraz”.

See also

- *Data Assurance Server Command Reference* on page 49
- *Remote Data Assurance Agent Command Reference* on page 163

Data Assurance Server Command Reference

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 58
- *alter connection* on page 57
- *test agent* on page 55

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 65

drop agent

Deletes an existing agent.

Syntax

drop agent *agent_name*

Parameters

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

Examples

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

```
drop agent myagent
go
```

- **Example 2 –** Removes agent names that begin with "a":

```
drop agent a*
go
```

show agent

Shows details of one, or all, agent.

Syntax

show agent *agent_name*

Parameters

- **agent_name –** (optional) the name of the agent with wildcards 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
```

- **Example 2 –** shows agent names that begin with "a":

```
show agent a*
go
```

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:

```
TASK ID    ESTIMATE COUNT FETCHING QUEUE TAKEN ESTIMATE SECONDS LEFT
---------- -------------- -------- ----- ----- ----------------------
4          511            true     0     511   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
------------------
-------------------------------------------------------------------------------
```

```
----------- ---------- ----------------------------------
SAP Java Server VM Jan 14 2014 23:45:56 - 71_REL - optU - windows amd64 -
6 - bas2:200241 (mixed mode) SAP AG    Java 1.7.0_25, VM 7.1.011 23.5-b11

JVM TOTAL MEM JVM FREE MEM JVM MAX MEM
------------- ------------ -----------
33.3 MB       19.4 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:

```
SERVER TASK ID CONNECTION OBJECT  STAGE              OBJ PROCESSED
PREDICATE
------ ------- ---------- ------- -----------------  -------------
---------
       3       venus      authors processing result  308
       5       pluto      authors processing result  219



SQL PROCESSED  TASK ID   ESTIMATE COUNT QUEUE TAKEN ESTIMATE SECONDS LEFT
--- ---------  -------   -------------- ----- ----- ----------------------
               3         511            0     308   0
               5         378            52    167   1
```

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
```

test agent config

Validates DA agent configuration details without creating an agent configuration object.

Syntax

test agent config

```
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

- **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 –** tests the agent configuration:

```
test agent config
set host=jupiter
and set port=4510
and set user=da_user
and set password=op3nSesame
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
```

Note: You cannot use the **alter connection** command to change the connection type.

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 JDBC driver connection property name.

  See *Connection Properties* in the *SAP jConnect for JDBC 16.0 Programmers Reference*.
- **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
```

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 | MSSQL | ORACLE | IQ |
HANADB | UDB}
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 –** the connection type supported by Replication Server Data Assurance Option.

  Each DA connection configuration object has a **type** parameter:

  - ASE – for an SAP Adaptive Server database connection.
  - ASE_CLUSTER– for an SAP Adaptive Server Cluster database connection.
  - RSSD – for an SAP Replication Server System Database (RSSD) connection.
  - IQ – for an SAP IQ database connection.
  - ORACLE– for an Oracle database connection.

- HANADB – for an SAP HANA database connection.
- MSSQL – for a Microsoft SQL Server database connection.
- UDB – for an IBM DB2 Universal Database (UDB) connection.

Note: Configure the JDBC drivers before creating IBM DB2 UDB, Microsoft SQL Server, Oracle or SAP HANA database connections. **type** is case-insensitive.

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 the 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 JDBC driver connection property name.

  See *Connection Properties* in the *SAP jConnect for JDBC 16.0 Programmers Reference*.
- **property_value –** the value of the property to be set.

Examples

- **Example 1 –** creates an SAP 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
```

- **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
```

- **Example 3 –** creates an SAP 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
```

- **Example 4 –** creates an SAP 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
go
```

  See *Connection Properties* in the *SAP jConnect for JDBC 16.0 Programmers Reference*.

- **Example 5 –** creates an Oracle database connection:

```
create connection ora1
set type=oracle
and set agent=localagent
and set host="10.0.14.129"
and set port=1521
and set database=XE
and set user=Aladdin
and set password=Op3nSesame
with properties
   set sessionTimeZone 'Europe/London'
go
```

  The Oracle database attribute must be set to the Oracle `SERVICE_NAME` value, which is stored in the `tnsnames.ora` file, in :`%ORACLE%\product\10.2.0\server\NETWORK\ADMIN\`. The exact path varies, based on the Oracle database version installed on your machine.

The attribute might contain:

```
SERVICE_NAME = XE
```

The `sessionTimeZone` connection property is used by the Oracle JDBC driver, when comparing `TIMESTAMP WITH LOCAL TIME ZONE` columns.

- **Example 6 –** creates an SAP IQ database connection:

```
create connection iq1
set type=IQ
and set agent=myagent
and set host= “10.0.14.119”
and set port=2638
and set database=iqdemo
and set user=Aladdin
and set password = Op3nSesame
go
```

- **Example 7 –** creates an SAP HANA database connection:

```
create connection myHDBconn1
set type=HANADB
and set agent=myagent
and set host="10.0.14.119"
and set port=30315
and set user=HAUSER
and set password=Pa55word
go
```

- **Example 8 –** creates an IBM DB2 UDB connection:

```
create connection myIBMdb1
   set type=UDB
   and set agent=myagent
   and set host="10.0.14.121"
   and set port=5001
   and set database=PUBS2
   and set user=JOHN
   and set password=ibmsq12
   go
```

- **Example 9 –** creates a Microsoft SQL Server connection:

```
create connection mysqldb2
   set type=MSSQL
   and set agent=myagent
   and set host="10.0.14.133"
   and set port=1433
   and set database=pubs2
   and set user=steve
   and set password=mste11
   go
```

See also

- *Heterogeneous Comparison Configuration* on page 31
- *create compareset* on page 73

- *create schema job* on page 109
- *show agent connection* on page 52
- *test connection* on page 65
- *Configuring DA Server to Use the Oracle JDBC Driver* on page 32
- *Configuring DA Server to Use the SAP HANA JDBC Driver* on page 32

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 with optional wildcards.

Examples

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

```
drop connection MyConnPDB
go
```

  The returned result is:

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

- **Example 2 –** drops all connection names that begin with "d":

```
drop connection d*
go
```

replace connection

replaces a current database connection definition, which includes the agent and the JDBC parameters with a new connection definition.

Syntax

```
replace connection connection_name
set type [{to|=}] {ASE | ASE_CLUSTER | RSSD | IQ | MSSQL | Oracle |
HANADB | UDB}
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 replaced.
- **type –** the connection type supported by Replication Server Data Assurance Option.

  Each DA connection configuration object has a **type** parameter:

  - ASE – for an SAP Adaptive Server database connection.
  - ASE_CLUSTER– for an SAP Adaptive Server Cluster database connection.
  - RSSD – for an SAP Replication Server System Database (RSSD) connection.

- IQ – for an SAP IQ database connection.
- ORACLE– for an Oracle database connection.
- HANADB – for an SAP HANA database connection.
- MSSQL – for a Microsoft SQL Server database connection.
- UDB – for an IBM DB2 Universal Database (UDB) connection.

Note: Configure the JDBC drivers before creating IBM DB2 UDB, Microsoft SQL Server, Oracle or SAP HANA database connections. **type** is case-insensitive.

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 the 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 JDBC driver connection property name.

  See *Connection Properties* in the *SAP jConnect for JDBC 16.0 Programmers Reference*.
- **property_value –** the value of the property to be set.

Examples

- **Example 1 –** replaces the current `prod1` connection with a new connection definition:

```
replace connection prod1
 set type = ASE
 set agent = localagent
 and set host = pluto
 and set port = 5050
 and set database = prod1
 and set user = prod_admin
 and set password = openSes4me
 with properties
   set ENCRYPT_PASSWORD = true
```

```
   and set PACKETSIZE = 1024
go
```

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 with wildcards. 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)
```

- **Example 2 –** shows connection names that end with "`t`":

  ```
  show connection *t
  go
  ```

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)
```

test connection config

Validates database connection details, which includes the agent and the JDBC parameters, without creating a connection configuration object.

Syntax

test connection config

```
set type [{to|=}] {ASE | ASE_CLUSTER | RSSD | MSSQL | ORACLE | IQ |
HANADB | UDB}
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

- **type –** the connection type supported by Replication Server Data Assurance Option.

  Each DA connection configuration object has a **type** parameter:

  - ASE – for an SAP Adaptive Server database connection.
  - ASE_CLUSTER– for an SAP Adaptive Server Cluster database connection.
  - RSSD – for an SAP Replication Server System Database (RSSD) connection.
  - IQ – for an SAP IQ database connection.
  - ORACLE– for an Oracle database connection.
  - HANADB – for an SAP HANA database connection.
  - MSSQL – for a Microsoft SQL Server database connection.
  - UDB – for an IBM DB2 Universal Database (UDB) connection.

Note: Configure the JDBC drivers before creating IBM DB2 UDB, Microsoft SQL Server, Oracle or SAP HANA database connections. **type** is case-insensitive.

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 the 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 JDBC driver connection property name.

  See *Connection Properties* in the *SAP jConnect for JDBC 16.0 Programmers Reference*.
- **property_value –** the value of the property to be set.

Examples

- **Example 1 –** tests the connection details to an Adaptive Server:

```
test connection config
set type=ASE
and set agent=localagent
and set host=pluto
and set port=5000
and set database=custdb
and set user=sa
go
```

The returned result is:

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

Count Commands

Commands for counting DA objects by type with optional wildcards.

count agent

Counts agents.

Syntax

count agent *agent_name*

Parameters

- **agent_name –** specifies an agent name with optional wildcards to count subsets of agents.

Examples

- **Example 1 –** counts all agents:

```
count agent
go

COUNT
-----
4

(0 rows affected)
```

count connection

Counts connections.

Syntax

count connection *connection_name*

Parameters

- **connection_name –** specifies the connection name with optional wildcards to count subsets of connections.

count compareset

Counts comparesets.

Syntax

count compareset *compareset_name*

Parameters

- **compareset_name –** specifies the compareset name with optional wildcards to count subsets of comparesets.

Examples

- **Example 1 –** counts all comparesets with names that begin with `foo`:

```
count compareset foo*
go

COUNT
-----
5
(0 rows affected)
```

count job

Counts jobs and comparisons.

Syntax

count job [*job_name* [*comparison_name*]]

Parameters

- **job_name –** specifies the job name with optional wildcards to count subsets of jobs.
- **comparison_name –** specifies the comparison name with optional wildcards to count subsets of comparisons.

Note: You cannot use wildcard characters in *job_name* when you specify a *comparison_name*.

Examples

- **Example 1 –** counts all jobs with names that end with `bar`:

```
count job *bar
go

COUNT
-----
3

(0 rows affected)
```

- **Example 2 –** counts all comparisons in the job `job1` that have names which contain the letter `a`:

```
count job job1 *a*
go
```

```
COUNT
-----
15

(0 rows affected)
```

count schema job

Counts schema jobs and comparisons.

Syntax

count schema job [*job_name* [*comparison_name*]]

Parameters

- **job_name –** specifies the schema job name with optional wildcards to count subsets of schema jobs.
- **comparison_name –** specifies the comparison name with optional wildcards to count subsets of comparisons.

Note: You cannot use wildcard characters in *job_name* when you specify a *comparison_name*.

Examples

- **Example 1 –** counts all schema jobs with names that begin with `test`:

```
count schema job test*
go

COUNT
-----
6

(0 rows affected)
```

- **Example 2 –** counts all comparisons in the schema job `schjob4` that have names that end with `x`:

```
count schema job schjob4 *x
go

COUNT
-----
1

(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 …]
```

This command also deletes all partition boundary values stored for this compareset.

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
  ```

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

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

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 [{to|=}] {true|false}]
[and set strict_key [{to|=}] {true|false}]
[and set strict_type [{to|=}] {true|false}]
[and set keep_computed [{to|=}] {true|false}]
[and set keep_encrypted [{to|=}] {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: When you use the **map all** statement, the compareset key columns are set by examining the primary key, identity and uniquely indexed columns in the source database table.
- **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_key –** if set to true, this parameter checks the source database table for at least one primary key, identity or uniquely indexed column. If no primary key or a uniquely indexed column is found, the command fails and the compareset is not created. If set to false, this parameter uses all non-LOB columns as compareset key columns. By default, **strict_key** is false.

  Note: When **strict_key** is false and all non-LOB columns are used as the compareset key columns, the keys might not be unique. Non-unique keys are detected during the data comparison and this might terminate the data comparsion prematurely.
- **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**, **strict_key**, **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
```

- **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
```

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 58
- *create job* on page 92

Considerations and Limitations

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

- When using the **map all** statement in **create compareset**, the source and target table columns must have identical names.
- When the source table contains one or more identity columns, these columns are mapped as DA compareset key columns, and DA does not look for primary keys or unique indexes.
- When the source table does not have any identity columns, but has one or more primary-key columns, these columns are mapped as DA compareset key columns. DA does not look for unique indexes.
- When the source table does not have any identity or primary key columns, but has one or more unique indexes, the indexed columns are mapped as compareset key columns from the first unique clustered or first unique nonclustered index DA finds.
- When no identity, primary key, or unique indexed columns are found:
  - If the **strict_key** is set to true, the **create compareset** command fails.
  - Otherwise, all non-LOB columns are used as the compareset key column.
- The **map all** statement never maps non-key SAP Adaptive Server timestamp columns.

Limitations for SAP Adaptive Server:

- 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 as:

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

```
  sp_primarykey 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 lacks a primary-key integrity constraint. You can create an index to enforce a primary-key integrity constraint. For example:

```
create unique clustered index ordernumidx on orders (order_num)
go
```

You can verify that a table has a primary-key constraint by using the **sp_helpconstraint** system procedure. See *Specifying Unique and Primary Key Constraints* in the *SAP Adaptive Server Enterprise Transact-SQL Users Guide*.

create compareset foreach

Creates a compareset for each table after searching for all matching table names in the source and target databases.

Syntax

```
create compareset [compareset_name_prefix] foreach table
  with
    source source_connection_name owner_name
    target target_connection_name owner_name
    [and target target_connection owner_name]…
  [{include | exclude} table_name_pattern [and table_name_pattern]…]
  [set abort_on_collision  [{to|=}] {true|false}
    [and set commit_batch_size [{to|=}] number_of_comparesets
    [and set strict_all_columns [{to|=}] {true|false}    [and set
strict_column_keys [{to|=}] {true|false}
    [and set strict_column_types [{to|=}] {true|false}
    [and set keep_computed_columns [{to|=}] {true|false}
    [and set keep_encrypted_columns [{to|=}] {true|false}
  ]]]]]]
go
```

Parameters

- **compareset_name_prefix –** a string to prefix all compareset names.
- **source_connection_name –** the name of the source database connection.
- **target_connection_name –** the name of the target database connection.
- **owner_name –** the name of the owner of the source and target tables.
- **table_name_pattern –** a pattern that table names must match to be included or excluded.
- **create_compareset_parameter –** See *Table 9. Create Compareset Parameters* table.

Note: The source and target *connection_name* and *owner_name* pairings must be unique within each command.

Table 9. Create Compareset Parameters

Parameter	Value
abort_on_collision	A compareset name generated during the execution of this command might match an existing compareset name (either created manually or through an earlier invocation of this command). When this parameter is set to true and a generated compareset name matches an existing compareset name, the command aborts immediately and all uncommitted comparesets are discarded. Valid values: true or false. Default: true.
commit_batch_size	The number of comparesets DA server stores in memory before committing them to the internal database. If an error is generated, all uncommitted comparesets are discarded. Valid values: 1 to 2147483647. Default: 1000.
strict_all_columns	When this parameter is set to true and one of the target tables does not contain all of the source columns, no compareset is generated. Valid values: true or false. Default value: false.
strict_column_keys	When this parameter is set to true and one of the source tables has no primary key, identity or uniquely indexed column, no compareset is generated. Valid values: true or false. Default: false.
strict_column_types	When this parameter is set to true and any target column type does not exactly match the source column type, no compareset is generated. Valid values: true or false. Default: false.

Parameter	Value
keep_computed_columns	Whether to include computed columns in the compareset column mappings. Valid values: true or false. Default: false.
keep_encrypted_columns	Whether to include encrypted columns in the compareset column mappings. Valid values: true or false. Default: false

Examples

- **Example 1 –** creates a compareset for each table that exists in both the source and target databases for DA connections "`conn_venus`" and "`conn_pluto`" respectively. Each compareset name uses the default prefix:

```
create compareset foreach table
  with source conn_venus dbo
       target conn_pluto dbo
go
```

- **Example 2 –** creates a compareset for each table that exists in both the source and target databases using the prefix "`cmpset_`", excluding the SAP Replication Server tables, which have names that are prefixed with "`rs_`":

```
create compareset cmpset_ foreach table
  with source conn_venus dbo
       target conn_pluto dbo
  exclude rs_*
go
```

- **Example 3 –** creates a compareset for each table that exists in both the source and target databases using the prefix "`cmpset_`" excluding SAP Replication Server tables. The command does not abort if a compareset is generated with the same name as one that already exists and buffered comparesets are committed to the DASD in batches of 100:

```
create compareset cmpset_ foreach table
  with source conn_venus dbo
       target conn_pluto dbo
  exclude rs_*
  set abort_on_collision false
    and set commit_batch_size 100
go
```

- **Example 4 –** creates a compareset for each table that exists in the source and both target databases using the default prefix, including tables only with names that begin with the prefix "`customer_`" or "`cust_`". Each target table must contain all the source table columns with the same name, type, precision, and scale:

```
create compareset foreach table
  with source conn_venus dbo
       target conn_pluto dbo
   and target conn_earth dbo
  include customer_*
      and cust_*
  set strict_all_columns true
    and set strict_column_types true
go
```

Usage

- When overriding the default compareset prefix:
  - The compareset name prefix cannot begin with a digit.
  - Do not create unnecessary duplicate comparesets.

    Note: DA creates duplicate comparesets, if you run **create compareset foreach** multiple times with different prefixes.
- You can use wildcards in the *table_name_pattern*. For example:
  - include a* and b* – includes all table names beginning with `a` or `b` only.
  - exclude *_???– excludes all table names that end with an underscore followed by any three characters, such as `table_001` and `table_002`.

See also

- *Wildcard Characters* on page 47

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 106

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 with optional wildcards.

Examples

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

  ```
  drop compareset cust_orders
  go
  ```

  This command also deletes all partition boundary values stored for this compareset.

- **Example 2 –** drops compareset names that end with "s":

  ```
  drop compareset *s
  go
  ```

replace compareset

Replaces the current compareset definition, which includes the source and target tables to be compared, with a new compareset definition.

Syntax

```
replace 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 [{to|=}] {true|false}]
[and set strict_key [{to|=}] {true|false}]
[and set strict_type [{to|=}] {true|false}]
[and set keep_computed [{to|=}] {true|false}]
```

```
[and set keep_encrypted [{to|=}] {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 replaced.
- **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: When you use the **map all** statement, the compareset key columns are set by examining the primary key, identity and uniquely indexed columns in the source database table.
- **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_key –** if set to true, this parameter checks the source database table for at least one primary key, identity or uniquely indexed column. If no primary key or a uniquely indexed column is found, the command fails and the compareset is not created. If set to false, this parameter uses all non-LOB columns as compareset key columns. By default, **strict_key** is false.

  Note: When **strict_key** is false and all non-LOB columns are used as the compareset key columns, the keys might not be unique. Non-unique keys are detected during the data comparison and this might terminate the data comparsion prematurely.
- **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**, **strict_key**, **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 –** replaces the current "`cust1`" compareset with a new compareset definition:

```
replace compareset cust1
with source prod1 dbo_cust1 s
     target prod1 dbo_cust1 t
map all
go
```

Usage

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

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 with wildcards.

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
```

- **Example 4 –** shows compareset names that begin with "s":

```
show compareset s*
go
```

Data Partition Commands

Commands for viewing and managing the data partition boundaries.

show boundary

Shows the data partition boundaries stored in the DASD, or a count of boundaries for each compareset.

Syntax

show boundary [*compareset_name*]

Parameters

- **compareset_name –** the name of the compareset for which data partition boundaries are shown.

  Use the optional wildcards to filter compareset names.

Examples

- **Example 1 –** shows the data partition boundaries stored in the DASD:

```
show boundary
go
```

  The returned result is:

```
COMPARESET BOUNDARIES
---------- ----------
cmpset_orders       8
cmpset_orders1      8
cmpset_orders2      8
```

- **Example 2 –** shows the data partition boundaries that match the compareset name filter *orders*:

```
show boundary *orders*
go
```

  The returned result is:

```
COMPARESET   BOUNDARIES
-----------   ----------
cmpset_orders      8
```

- **Example 3 –** shows the data partition boundaries for a compareset named "cmpset_orders":

```
show boundary cmpset_orders
go
```

  The returned result is:

```
KEY NAME        TYPE
--- ----------  -------
0   order_id  INTEGER


BOUNDARY POSITION KEY VALUE
-------- -------- --- -------
0        1000     0   1001000
1        2000     0   1002000
2        3000     0   1003000
3        4000     0   1004000
4        5000     0   1005000
5        6000     0   1006000
6        7000     0   1007000
7        8000     0   1008000
```

drop boundary

Removes the data partition boundaries stored in the DASD for a *compareset_name*.

Syntax

drop boundary [*compareset_name*]

Parameters

- **compareset_name –** the name of the compareset for which data partition boundaries are deleted.

Examples

- **Example 1 –** deletes the data partition boundaries for a compareset named "cmpset_orders":

```
drop boundary cmpset_orders
go
(8 rows affected)
```

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_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 NUM_PARTITIONS [{to|=}] number
[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_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 NUM_PARTITIONS [{to|=}] number
[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 –** (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.
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.

Parameter	Value
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.
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.

Parameter	Value
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 SAP Replication Server Data Assurance Option. **Note:** Set the hash type comparison option to `agent_hash` for heterogeneous comparison. The `database_hash` comparison option is used only for SAP Adaptive Server-to- SAP Adaptive Server comparisons. Default value: database_hash.
NUM_PARTITIONS	Specifies the number of partitions for a table.
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 –** alters “myjob_1” to remove its comparison:

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

- **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
```

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 [{to|=}]compareset_name
[and set ABORT_DIFF_MAX [{to|=}] number_of_differences]
[and set ABORT_DIFF_ROW_COUNT [{to|=}] {true|false}]
[and set COMPARE_MODE [{to|=}] {row_compare | key_compare |
row_count}]
[and set COMPRESS_DATA_TRANSFER [{to|=}] {true|false}]
[and set CREATE_COL_LOG [{to|=}] {true|false}
    [and set AUTO_RECONCILE [{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 NUM_PARTITIONS [{to|=}] number]
[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}
  [and set column_name [{to|=}] {literal | column_hash | row_hash}]
 […]]
[and comparison comparison_name2…]
```

```
[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 are active.

To create a job with a comparison for each compareset:

```
create job job_name
   [set description [{to|=}] description
   [and set max_concurrent_comparisons [{to|=}]
      max_concurrent_comparisons
  ]]
  add comparison foreach compareset
       [{include | exclude} compareset_name_pattern
       [and compareset_name_pattern]…]
       [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_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 NUM_PARTITIONS [{to|=}] number
       [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
]]]]]]]]]
```

This command creates multiple comparisons using all existing comparesets that match with the compareset name pattern.

Note: You can use both **add comparison foreach compareset** and **add comparison** *name* in the same **create job** command.

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 –** (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.
- **compareset_name_pattern –** (optional) wildcard pattern of a compareset name must match to be included or excluded. If no comparesets match the *compareset_name_pattern*, no job is created and an error message is generated.
- **exist_job_name –** the name of an existing job to be cloned.

Table 13. 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.

Parameter	Value
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.
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.

Parameter	Value
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 SAP Replication Server Data Assurance Option. **Note:** Set the hash type comparison option to `agent_hash` for heterogeneous comparison. The `database_hash` comparison option is used only for SAP Adaptive Server-to- SAP Adaptive Server comparisons. Default value: database_hash.
NUM_PARTITIONS	Specifies the number of partitions for a table.
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 14. 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 15. 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
```

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

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

- **Example 3 –** creates a job named “job1” and sets the number of data partitions to 2:

```
create job job1
set MAX_CONCURRENT_COMPARISONS = 2
ADD comparison cmp1
  set COMPARESET=cs
  and set NUM_PARTITIONS to 2
  and set COMPARE_MODE to row_compare
  and set ENABLE_ROW_COUNT to false
go
```

- **Example 4 –** creates a job with a comparison for each existing compareset using all the default options:

```
create job myJob
    add comparison foreach compareset
  go
```

- **Example 5 –** creates a job with a *key compare* comparison for each compareset, except for those with a name that begins with "a":

```
create job myJob
    add comparison foreach compareset exclude a*
      set compare_mode key_compare
  go
```

- **Example 6 –** creates a job with a *row count* comparison for each compareset that have a name beginning with "a" or "b":

```
create job myJob
    add comparison foreach compareset include a* and b*
    set compare_mode to row_count
  go
```

- **Example 7 –** creates a job with a *row compare* comparison including reconciliation for each compareset with a name that begins with "a" or "b", and a *key compare* comparison for each compareset with a name that begins with "c":

```
create job myJob
    add comparison foreach compareset include a* and b*
      set compare_mode = row_compare
      and set create_col_log = true
      and set create_recon_script = true
    and comparison foreach compareset include c*
```

```
      set compare_mode = key_compare
  go
```

- **Example 8 –** creates a job with a *key compare* comparison for each compareset that begins with "a" or "b", and a *row compare* comparison for the rest:

```
create job myJob
    add comparison foreach compareset include a* and b*
      set compare_mode key_compare
    and comparison foreach compareset exclude a* and b*
      set compare_mode row_compare
  go
```

- **Example 9 –** shows how conflicts occur when a **create job** statement contains two or more *foreach* constructs:

```
create job myJob
    add comparison foreach compareset include a*
      set compare_mode key_compare
    and comparison foreach compareset include *z
      set compare_mode row_count
  go
```

  Any compareset with a name beginning with "a" and ending with "z" creates a conflict:

  - When comparisons use the name of the compareset they operate on, this leads to add two comparisons with the same name.
  - Data Assurance cannot drop one of the comparison definitions in favor of the other. As a result, an error occurs and job is not created.

drop job

Deletes an existing job.

Syntax

drop job *job_name*

Parameters

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

Examples

- **Example 1 –** drops “myjob_1”:

```
drop job myjob_1
go
```

  The returned result is:

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

- **Example 2 –** drops all jobs with names that end with "c":

```
drop job *c
go
```

replace job

Replaces a current job definition, including all comparisons and schedules, with a new job definition.

Syntax

```
replace 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_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 NUM_PARTITIONS [{to|=}] number
[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 replace a job definition with an existing job definition:

```
replace job job_name with exist_job_name
```

Note: When you replace a job with schedules, the new job includes the replaced schedules but they are inactive.

To replace a current job with a comparison for each compareset in a new job definition:

```
replace job job_name
   [set description [{to|=}] description
   [and set max_concurrent_comparisons [{to|=}]
      max_concurrent_comparisons
  ]]
  add comparison foreach compareset
       [{include | exclude} compareset_name_pattern
       [and compareset_name_pattern]…]
       [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_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 NUM_PARTITIONS [{to|=}] number
       [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

- **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 –** (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.
- **exist_job_name –** the name of an existing job to be cloned.

Table 16. 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.

Parameter	Value
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.
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.

Parameter	Value
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 SAP Replication Server Data Assurance Option. **Note:** Set the hash type comparison option to `agent_hash` for heterogeneous comparison. The `database_hash` comparison option is used only for SAP Adaptive Server-to- SAP Adaptive Server comparisons. Default value: database_hash.
NUM_PARTITIONS	Specifies the number of partitions for a table.
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 17. 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 18. 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 –** replaces the current job named "`cust_chk`" with a new job definition:

```
replace job cust_chk
add comparison cust1
    set compareset=cust1
  set create_col_log = true
  and set create_recon_script = true
go
```

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 with optional wildcards.
- **comparison_name –** the name of the comparison with optional wildcards.

Note: You cannot use wildcard characters in **job_name** when you specify a **comparison_name**.

Examples

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

```
show job
go
```

The returned result is:

```
NAME       ACTIVE  DESCRIPTION
---------  ------- ------------------------------------
job_cust   true    Compares the data in the customer db
job_bo     true    Compares the back office datamy_job2
```

- **Example 2 –** shows job names that end with "h":

```
show job *h
go
```

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
```

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

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

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 job comparison `sc_job_test1` and includes all tables with some exceptions:

```
create schema job sc_job_test1
  set max_concurrent_comparisons = 10
  add comparison cmp1
    with source pdb0_conn s
         target rdb1_conn t1
     and target rdb2_conn t2
  include all tables
  exclude
     s.tab_x and s.tab_y
```

- **Example 2 –** creates a schema job comparison `sc_job_test2` and maps the source and target tables with similar names:

```
create schema job sc_job_test2
  set max_concurrent_comparisons = 4
  add comparison cmp1
    with source pdb0_conn s
         target rdb1_conn t1
```

```
    and target rdb2_conn t2
map tables
      s.dbo.tab_a = t1.dbo.tab_a = t2.dbo.tab_a
  and s.dbo.tab_b = t1.dbo.tab_b = t2.dbo.tab_b
  and s.dbo.tab_c = t1.dbo.tab_c
  and s.dbo.tab_d = t2.dbo.tab_d
```

Usage

- The **include all tables** clause specifies that all tables in the source database should be included for comparison, and use automatic name mapping between source database and target database tables. 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
```

replace schema job

Replaces the current schema job for comparing database object schemas with a new schema job definition.

Syntax

```
replace 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 –** replaces the current schema job comparison `cust_sch_chk` with the new definition:

```
replace schema job cust_sch_chk
  set max_concurrent_comparisons 3
  add comparison chk1
    with source prod1 s
         target back1 t
    map tables
       s.dbo.cust1 = t.dbo.cust1
go
```

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 with optional wildcards.
- **schema_job_comparison_name –** the name of the schema comparison with optional wildcards.

Note: You cannot use wildcard characters in **schema_job_name** when you specify a **schema_job_comparison_name**.

Examples

- **Example 1 –** shows schema job names that begin with "sc":

```
show schema job sc*
go
```

The returned result is:

```
NAME       ACTIVE   DESCRIPTION
---------- -------- -------------------------------------
scjob_cust true     Compares the tables in the customer db
scjob_bo   true     Compares the back office tables
```

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
```

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
```

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.

  Use asterisk (*) to drop all job history sequence IDs.

Examples

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

```
drop history myjob_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
```

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 with optional wildcards.
- **comparison_name –** the name of the comparison with optional wildcards.

Note: You cannot use wildcard characters in **job_name** when you specify a **comparison_name**.

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 the runtime information for the job named "`j1`":

```
monitor job j1
go
```

The returned result is:

```
COMPARISON PART STATUS   SUBMIT TIME        END TIME          RUN
PROGRESS
---------- ---- -------- ------------------- ------------------
--- --------
cmp1       0    FINISHED 2011-10-10 17:16:13 2011-10-10 17:16:33 6
100%
cmp2       0    FINISHED 2011-10-10 17:16:13 2011-10-10 17:16:32 6
100%
```

- **Example 3 –** shows the comparison information for the job named "`j1`":

```
monitor job j1 cmp1
go
```

The returned result is:

```
COMPARISON  PART   SUBMIT TIME          END TIME
----------  ----   ------------------- -------------------
cmp1        0      2011-10-10 17:16:13 2011-10-10 17:16:33
            0      2011-10-10 17:16:13 2011-10-10 17:16:33


RUN  PHASE                PART  TYPE SUMMARY              START TIME
---  -------------------- ----  ---- -------------------- -----------
1    COMPARE_ALL             0     S    conn1/dbo.da1_10m     2011-10-10
17:16:14
                                   T    conn2/dbo.da1_10m     2011-10-10
17:16:14
                             0     S    conn1/dbo.da1_10m     2011-10-10
17:16:14
                                   T    conn2/dbo.da1_10m     2011-10-10
17:16:14
2    RECHECK_DIFFERENCES     0     T    conn2/dbo.da1_10m     2011-10-10
17:16:14
                             0     T    conn2/dbo.da1_10m     2011-10-10
17:16:14
3    RECHECK_DIFFERENCES     0     T    conn2/dbo.da1_10m     2011-10-10
17:16:14
                             0     T    conn2/dbo.da1_10m     2011-10-10
17:16:14
4    RECHECK_DIFFERENCES     0     T    conn2/dbo.da1_10m     2011-10-10
17:16:14
                             0     T    conn2/dbo.da1_10m     2011-10-10
17:16:14
5    VERIFY_DIFFERENCES      0     S    conn1/dbo.da1_10m     2011-10-10
17:16:14
                                   T    conn2/dbo.da1_10m     2011-10-10
17:16:14
                             0     S    conn1/dbo.da1_10m     2011-10-10
17:16:14
                                   T    conn2/dbo.da1_10m     2011-10-10
17:16:14
```

```
6  CREATE_RECONCILIATION_SCRIPT ALL   T    conn2/dbo.da1_10m
2011-10-10 17:16:33
```

- **Example 4 –** shows the summary of comparison named "cmp_dp" with the combined progress:

  ```
  monitor job job1 cmp_dp summary
  go
  ```

  The returned result set is:

```
COMPARISON PART SUBMIT TIME         END TIME
---------- ---- ------------------ ------------------

cmp_dp     0    2012-10-02 11:18:44

RUN PHASE       PART TYPE SUMMARY              START TIME          END
TIME            COUNT   READ   M  O I     R PROGRESS ESTIMATE END ERROR
--- ----------- ---- ---- -------------------- -------------------
------------------- ------- ------ -- - ----- - -------- ------------
-----
1   COMPARE_ALL ALL  S    soka2ase/dbo.da1_10m 2012-10-02
11:18:50               1000000 586533              62.9%
                    T    soka3ase/dbo.da1_10m 2012-10-02
11:18:50               999990  586521 10 0 10015   62.9%
```

run job

Starts a specified job.

Syntax

run job *job_name* [*wait* [*timeout*]]

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

Parameters

- **job_name –** the name of the job to be started.
- **wait –** (optional) the **isql** prompt does not return until DA server completes the job.
- **timeout –** (optional) specifies a value, in seconds, after which you can regain control of the **isql** prompt if, for example, the job is taking too long to complete. Valid values are 1 to 2147483647; if you do not specify a **timeout** parameter value, the **isql** prompt waits indefinitely.

Examples

- **Example 1 –** executes “myjob_1”:

  ```
  run job myjob_1
  go
  ```

- **Example 2 –** waits indefinitely until the job is completed. You cannot regain the **isql** prompt control until DA server completes the job. The job is considered complete when all

comparisons run successfully, or if there are errors that abort the job, or if you abort the job manually using a different **isql** prompt:

```
run job myjob wait
go
```

The returned result is:

```
SUBMIT TIME         FINISH TIME         COMPARISONS READ (S) DIFFS M O I R
ERRORS
------------------- ------------------- ----------- -------- ----- - - -
- ------
2013-06-27 10:20:51 2013-06-27 10:26:59 1          10000    0     0 0 0 0 0

(0 rows affected)
```

In this example, you regain control of the **isql** prompt after 6 seconds approximately (the difference between the Submit Time and the Finish Time).

- **Example 3 –** waits for 10 seconds before transferring control of the **isql** prompt to you. The job is considered complete when all comparisons run successfully, or if there are errors that abort the job, or if you abort the job manually using a different **isql** prompt:

  ```
  run job myjob wait 10
  go
  ```

  The returned result is:

```
SUBMIT TIME         FINISH TIME         COMPARISONS READ (S) DIFFS M O I R
ERRORS
------------------- ------------------- ----------- -------- ----- - - -
- ------
2013-06-27 10:31:52 2013-06-27 10:38:02 1          10000    0     0 0 0 0 0

(0 rows affected)
```

  In this example, you regain control of the **isql** prompt after 6 seconds approximately (the difference between the Submit Time and the Finish Time).

- **Example 4 –** waits for 5 seconds before transferring control of the **isql** prompt to you. The job is considered complete when comparisons run successfully, or if there are errors that abort the job, or if you abort it manually using a different **isql** prompt:

  ```
  run job myjob wait 5
  go
  ```

  The returned result is:

```
COMPARISON PART STATUS  SUBMIT TIME         END TIME RUN PROGRESS NEXT
RETRY ERROR
---------- ---- ------- ------------------- -------- --- --------
---------- -----
mycmp      0    RUNNING 2013-06-27 10:49:54          1   89%
[#103] Waited 5 seconds, the job is still running.

(0 rows affected)
```

In this example, you regain control of the **isql** prompt after 5 seconds and the output shows the current 'running' state of the job.

show history

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

Syntax

```
show history
[job_name
    [ {history_id | latest
      [ {comparison_name | summary }]
        [where where_argument]
] ]
```

Parameters

- **job_name –** the name of the job for which to show history. If a job name contains optional wildcards, it acts as a filter.
- **history_id –** the job history sequence ID.
- **latest –** indicates the job history with the highest sequence ID.
- **comparison_name –** the comparison name filter to show job history. If the comparison name contains optional wildcards, all matching comparisons are shown.
- **summary –** shows the history in a summary format, in which the history for all comparisons is summarized into a single row.
- **where –** the **where** argument to filter the job history.

Table 19. where Agruments

where Argument	Description
differences	Shows only the comparisons that found one or more missing, inconsistent, or orphaned rows
missing	Shows only the comparisons that found one or more missing rows
inconsistent	Shows only the comparisons that found one or more inconsistent rows
orphaned	Shows only the comparisons that found one or more orphaned rows
reconciliation	Shows only the comparisons that reconciled one or more rows
errors	Shows only the comparisons that encountered one or more errors

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 set:

```
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)
COMPARISON TARGET RECONCILIATION SCRIPT
---------- ------
-------------------------------------------------------------------------
---------------
c          0      C:\Sybase\DA-15_5\server\instance\data
\job2\2012-10-05\09.11.28.585\c1_T_recon_ins.sql
                  C:\Sybase\DA-15_5\server\instance\data
```

```
\job2\2012-10-05\09.11.28.585\c1_T_recon_upd.sql
                  C:\Sybase\DA-15_5\server\instance\data
\job2\j1\2012-10-05\09.11.28.585\c1_T_recon_del.sql
```

- **Example 3 –** shows the number of histories for each job:

```
show history
go

JOB NAME     HISTORIES
-----------  ---------
custTables   3
stockTables  12
test1        2

(0 rows affected)
```

- **Example 4 –** shows the individual history items for the job named `custTables`:

```
show history custTables
go

HISTORY ID   SUBMIT TIME           FINISH TIME
----------   -------------------   -------------------
42           2012-07-13 11:24:43   2012-07-13 11:25:51
39           2012-07-12 10:18:01   2012-07-12 10:19:11
37           2012-07-11 10:33:12   2012-07-11 10:34:20

(0 rows affected)
```

- **Example 5 –** shows a summary of history item 42 for the job named `custTables`:

```
show history custTables 42 summary
go


SUBMIT TIME        FINISH TIME        COMPARISONS READ (S) DIFFS M O I  R
ERRORS
------------------- ------------------- ----------- -------- ----- - - --
-- ------
2012-07-13 11:31:56 2012-07-13 11:32:44 3          30045    45    0 0 45 45
0

(0 rows affected)
```

Using the **latest** keyword is equivalent to using the highest history ID. This command is equivalent to the command above:

```
show history custTables latest summary
go
```

- **Example 6 –** shows the latest history for the job named `test1`:

```
show history test1 latest
go

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

```
---------- --- ----------- ---- ----------------------------
------------------- ------------------- ----- ---- - - - - -----
c          1   COMPARE_ALL S    zeus:5000/myasedb.dbo.test1  2012-07-17
16:35:38 2012-07-17 16:35:39 1     1
                           T    hera:1521/qsora11g.QA1.TEST1 2012-07-17
16:35:38 2012-07-17 16:35:39 1     1    0 0 0

(0 rows affected)
```

- **Example 7 –** shows the latest history for the job named `stockTables`, showing only the comparisons that encountered some differences:

```
show history stockTables latest where differences
go

COMPARISON RUN PHASE                       TYPE SUMMARY
START TIME          END TIME            COUNT READ  M O I  R  ERROR
---------- --- ---------------------------- ----
-------------------------- ------------------- ------------------- -----
----- - - -- -- -----
sku        1   COMPARE_ALL                 S    prod1:5000/stockdb.dbo.sku
2012-07-13 11:32:18 2012-07-13 11:32:20 N/A   10000
                                           T    back1:5000/stockdb.dbo.sku
2012-07-13 11:32:18 2012-07-13 11:32:19 N/A   10000 0 0 15
                                           T2   back2:5000/stockdb.dbo.sku
2012-07-13 11:32:18 2012-07-13 11:32:20 N/A   10000 0 0 0
           2   RECHECK_DIFFERENCES          T
2012-07-13 11:32:29 2012-07-13 11:32:30       15    0 0 15

T2                                                                    N/A
           3   VERIFY_DIFFERENCES           S
2012-07-13 11:32:30 2012-07-13 11:32:30       15
                                           T
2012-07-13 11:32:30 2012-07-13 11:32:30       15    0 0 15

T2                                                                    N/A
           4   CREATE_RECONCILIATION_SCRIPT T
2012-07-13 11:32:30 2012-07-13 11:32:30                    15
                                             T2

(0 rows affected)


COMPARISON TARGET RECONCILIATION SCRIPT
---------- ------
------------------------------------------------------------------------
--------------
sku        0      C:\Sybase\DA-15_5\myserver\data\stockTables
\2012-07-13\11.31.56.007\sku_T_recon_upd.sql

(0 rows affected)
```

show reconcile

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

Syntax

```
show reconcile job_name history_id comparison_name target_index
script_type
```

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.
- **history_id –** the job history sequence ID of the reconcile script to be shown.
- **comparison_name –** the name of the job comparison.
- **target_index –** the index of the target. Use zero (0) for the first target.
- **script_type –** the type of script to return: **insert**, **update**, or **delete**.

Examples

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

```
show reconcile job6 29
go
```

The returned result is:

```
COMPARISON TARGET              START TIME          FINISH TIME         R
TYPE   SCRIPT
---------- ------------------- ------------------- ------------------- -
------
--------------------------------------------------------------------------
--------------
cmp6       pluto:5000/dbo.cust 2012-11-06 11:53:02 2012-11-06 11:53:02 9
INSERT /Sybase/DA-15_5/server/myserver/data/
job6/2012-11-06/11.52.56.458/cmp6_T_recon_ins.sql
```

The `Type` column shows the type of SQL statement (**insert**, **update**, or **delete**) and the script file name contains an `ins`, `upd`, or `del` indicator.

If the reconcilation type is automatic, the output is shown as:

```
COMPARISON TARGET START TIME          FINISH TIME         R TYPE SCRIPT
---------- ------ ------------------- ------------------- - ---- -------
cmp6       0      2012-08-31 12:39:31 2012-08-31 12:39:34 8 AUTO N/A
```

- **Example 2 –** shows the **insert** reconciliation script for the `cmp1` comparison in the `job6` with history ID 29:

```
show reconcile job6 29 cmp1 0 insert
go
```

This command returns multiple rows of content columns, such as:

```
CONTENT
-------------------------------------------------------------------------
Replication Server Data Assurance Option - DA Server/15.7.1 /SP100/B/
generic...
Reconciliation Script (Auto-generated); reconciles 9 missing row(s).
```

show report

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

Syntax

show report *job_name history_id latest report type*

Parameters

- **job_name –** the name of the job for which to show the report.
- **history_id –** the job history sequence ID of the report to be shown.
- **latest –** the job history with the highest sequence ID to be shown in the report.
- **report_type –** the type of report to fetch (Text or XML).

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        /Sybase/DA-15_5/server/myserver/data/
job6/2012-11-06/11.52.56.458/report.txt
XML         /Sybase/DA-15_5/server/myserver/data/
job6/2012-11-06/11.52.56.458/report.xml
```

- **Example 2 –** shows the text report for the `job6` with history ID 29:

  ```
  show report job6 29 TEXT
  go
  ```

  The returned result is:

  ```
  CONTENT
  -------------------------------------
  Report generated: 2012-11-07 12:34:01
  File encoding: UTF-8
  ```

```
Job submitted: 2012-11-06 11:52:56
Job completed: 2012-11-06 11:53:02
Differences detected: 9
…
```

- **Example 3 –** Shows the text report using the **latest** keyword for the `job6`:

```
show report job6 latest TEXT
```

The result displayed here shows only the first row of the report:

```
CONTENT
 ---------------------------------------
 Report generated: 2014-01-10 12:32:01
 File encoding: utf8
 Job submitted: 2013-10-09 15:06:05
 Job completed: 2013-10-09 15:06:43
 Differences detected: 3

job job6
   options
     MAX_CONCURRENT_COMPARISONS = 5
   comparison c
     COMPARESET = compareset5
 ...
```

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|`*`history_id`*`]]`

Parameters

- **all –** truncates all job history records.
- **job_name –** the name of the job for which to truncate history.
- **history_id –** a 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_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 NUM_PARTITIONS [{to|=}] number
  [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 20. 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.

Parameter	Value
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.
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.

Parameter	Value
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 SAP Replication Server Data Assurance Option. **Note:** Set the hash type comparison option to `agent_hash` for heterogeneous comparison. The `database_hash` comparison option is used only for SAP Adaptive Server-to- SAP Adaptive Server comparisons. Default value: database_hash.
NUM_PARTITIONS	Specifies the number of partitions for a table.
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 21. 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 22. 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
```

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.

  Use asterisk (*) to drop all backup indexes.

Examples

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

```
drop backup 3
go
```

- **Example 2 –** deletes all backup indexes:

```
drop backup *
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.

  Use asterisk (*) to truncate all backup indexes.
- **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 23. 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.
boundary_sample_size	Determines the number of samples stored in memory during a comparison run. A **boundary_sample_size** value that is too low leads to uneven spacing of boundary samples stored because of fewer boundaries to choose from. Default: 64 Min: 8 Max: Integer.Max_Value This parameter does not require a restart of DA server to take effect.
boundary_sample_step	Determines how often rows are sampled during a comparison. A **boundary_sample_step** value that is too low leads to DA taking more time to search for the boundary samples. Default: 1000 Min: 1 Max: Integer.Max_Value This parameter does not require a restart of DA server to take effect.
boundaries_stored	Determines how many boundary samples per compareset are stored in the DASD once a comparison is completed. The **boundaries_stored** value cannot be greater than the **boundary_sample_size**, because DA stores only a subset of the boundary samples obtained during a comparison run. Default: 8 Min: 8 Max: 1024 This parameter does not require a restart of DA server to take effect.

parameter_name	parameter_value
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.
column_option_helper_visit_db	Determines whether the internal column option helper visits the database to fetch column metadata to select the most appropriate column compare modes when choosing defaults. Default: true. Valid values: true or false. When set to false, the default column compare modes are not verified and may be illegal. When job comparisons are created with explicit column compare modes, this parameter is redundant.
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
comparer_scale_rounding	Specifies the maximum number of scale digits (digits to the right of the decimal point) in a number that is compared. If a number's scale is greater than the configured *scale rounding* value, its scale is rounded up to the nearest *scale rounding* value number of digits. For example: • For 1.002 to equal 1.0017, set the *scale rounding* to 3 or lower. • For -467.84921 to equal -467.849207, set the *scale rounding* to 5 or lower. A value of zero (0) disables this parameter. Default: 10 Min: 0 Max: 2147483647 This parameter does not require a restart of DA server to take effect.
db_connection_pool_size	Specifies the maximum number of database connections DA agent allows to be open concurrently per connection profile. Default: 5 Min: 1 Max: 2147483647 **Note:** Access to the database is blocked, if the limit is reached. This parameter does not require a restart of DA server to take effect.
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.

parameter_name	parameter_value
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`, or `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.
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.

parameter_name	parameter_value
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.
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.

parameter_name	parameter_value
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: 2147483647 This parameter does not require a restart of DA server to take effect.
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.

parameter_name	parameter_value
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.
text_report_diff_page_size	Specifies the maximum number of rows to show in each "page" of differences. Where a page consists of a list of rows displayed in the order they were encountered. Default: 10 Min: 1 Max: 2147483647 Changing this parameter does not effect the reports that are generated earlier. 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)
  ```

dbfetch

Fetches schema, table, and column metadata from a database connection.

Syntax

```
dbfetch {
      SCHEMA connection_name [schema_pattern]
    | TABLE  connection_name schema_name [table_pattern]
    | COLUMN connection_name schema_name table_name [column_pattern]
    | IMPORT { SOURCE rssd_connection_name
             | TARGET rssd_connection_name rssd_source_database }
}
```

Parameters

- **connection_name –** the name of an existing DA connection object.
- **schema_pattern –** (optional) a wildcard pattern to filter the schema results.
- **schema_name –** the name of a schema.
- **table_pattern –** (optional) a wildcard pattern to filter the table results.
- **table_name –** the name of a table.
- **column_pattern –** (optional) a wildcard pattern to filter the column results.
- **rssd_connection_name –** the name of an existing RSSD connection configured in DA.
- **rssd_source_database –** the name of source database which exists in the RSSD.

Examples

- **Example 1 –** fetches all the schemas found under the connection `prod1`:

  ```
  dbfetch schema prod1
  go
  ```

  The returned result is:

  ```
  SCHEMA
  ------------
  dauser
  dbo
  ```

- **Example 2 –** fetches all the table names in the `dbo` schema with table names that begin with the prefix "rs":

  ```
  dbfetch table prod1 dbo rs_*
  go
  ```

  The returned result is:

  ```
  SCHEMA  TABLE
  ------  -----------------
  dbo     rs_lastcommit
  ```

```
dbo     rs_threads
dbo     rs_ticket_history
```

- **Example 3 –** fetches column metadata for the `cust1` table, showing only the columns with names beginning with the letter " i ":

  ```
  dbfetch column prod1 dbo cust1 i*
  go
  ```

  The returned result is:

  ```
  COLUMN  TYPE     PRECISION  SCALE  KEY
  ------  -------  ---------  -----  -----------
  id      int      0          0      PRIMARY KEY
  ```

- **Example 4 –** fetches source (primary) database names from the RSSD connection:

  ```
  dbfetch import source my_rssd
  go
  ```

  The returned result is:

  ```
  SOURCE DB
  -----------
  prod1.cust1
  ```

- **Example 5 –** fetches target (replicate) database names from the RSSD connection, for the `prod1.cust1` source database:

  ```
  dbfetch import target my_rssd 'prod1.cust1'
  go
  ```

  The returned result is:

  ```
  SOURCE DB     TARGET DB
  -----------   -----------
  prod1.cust1   back1.cust1
  ```

license

Displays the DA server license information and modifies the email reporting settings for the license.

Syntax

```
license [set parameter_name [{to|=}] parameter_value]
    [and set parameter_name [{to|=}] parameter_value][...]]
```

Parameters

- **smtp_host –** the mail server host name.
- **smtp_port –** the mail server port number.
- **email_sender –** the email ID of the sender (the 'From' field).
- **email_recipients –** a comma-separated list of email addresses of recipients.

- **email_severity –** the severity level for email reporting. Valid values are NONE, ERROR, WARNING, INFORMATIONAL, LOG or DEBUG.

Examples

- **Example 1 –** shows the license information of the DA server:

```
license
go
```

  The returned result is:

```
 SMTP_HOST          SMTP_PORT EMAIL_SENDER  EMAIL_RECIPIENTS
EMAIL_SEVERITY
 ----------------   --------- ------------- ----------------
--------------
 mail.myhost.com    1025      Administrator dba@myhost.com   NONE

 FEATURE           ISSUER                          VENDOR EDITION MAX CPU
QUANTITY EXPIRE              VERSION
 ------------------------------------------------  ------ -------
------- -------- ------------------- ----------
 REP_DATA_ASSURANCE CO=Sybase, Inc.;V=15.0;MP=365;CP=0 LT=AC  NULL
0       2        2014-04-01 00:00:00 2014.03315
```

- **Example 2 –** changes the SMTP host name and port number:

```
license set smtp_host = 'mail4.myhost.com' and set smtp_port 4025
go
```

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 185

restore config

Restores the DA server configuration to the default settings.

The command requires no arguments.

Syntax

restore config

Examples

- **Example 1 –** restores all DA server configuration settings to their default values:

```
restore config
go
```

role

Maps LDAP users to the DA administrator role.

Syntax

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

Parameters

- **rolename –** case-sensitive role name with optional wildcards.
- **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
```

session

View and modify session parameters.

Syntax

session [**parameter_name** [*parameter_value*]]

Parameters

- **parameter_name –** the name of a session parameter with optional wildcards.
- **parameter_value –** the parameter value to set.

Table 24. Parameters

Name	Description
check_column_options	When set to true, DA server that checks each comparison's selected column options are legal for the columns types found in the database. If any illegal column options are found, a warning is issued. Applicable only to **create job**. Key columns must always use the **literal** option to compare column literal values. DA enforces this option for key columns. Valid values: true or false. Default value: true.
max_rows	The maximum number of rows to show in the result. Valid values: 1 to 2147483647. Default: 100.
show_duration	When set to true, the **monitor job**, **show history**, and **show reconcile** commands show a duration column in place of a finish time or an end time column. Valid values: true or false. Default: false.
wildcard_ignore_case	Determines whether wildcard matching should be case-insensitive. Valid values: true or false. Default: false.

Examples

- **Example 1 –** shows the first three jobs:

```
session max_rows 3
go

(1 row affected)

show job
go

NAME    ACTIVE DESCRIPTION
------  ------ -----------
a_job1  true
```

```
a_job2  true
a_job3  true

[#101] Warning: showing the first 3 row(s) only.

(0 rows affected)
```

- **Example 2 –** shows the first three jobs with names beginning with "m":

```
session max_rows 3
go

(1 row affected)


show job m*
go

NAME    ACTIVE DESCRIPTION
------  ------ -----------
myjob1  true
myjob2  true
myjob3  true

[#101] Warning: showing the first 3 row(s) only.

(0 rows affected)
```

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
------------------
------------------------------------------------------------------------
----------- ---------- ---------------------------------
SAP Java Server VM Jan 14 2014 23:45:56 - 71_REL - optU - windows amd64 -
6 - bas2:200241 (mixed mode) SAP AG    Java 1.7.0_25, VM 7.1.011 23.5-b11
```

```
JVM TOTAL MEM JVM FREE MEM JVM MAX MEM
------------- ------------ -----------
33.3 MB       19.4 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 SAP Replication Server Data Assurance Option.

Syntax

version

Examples

- **Example 1 –** shows the version:

```
version
go
```

The returned result is:

```
VERSION
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-------
SAP Replication Server Data Assurance Option - DA Server/15.7.1/SP200/P/
generic/generic/damain/610/VM: SAP AG 1.7.0_25/OPT/Fri 31 Jan 2014
05:29:42 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 25. DA Server Reserved Words

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

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

Data Assurance Server Configuration Properties

Use the *instance*.cfg configuration file to set the DA server system properties.

The *instance*.cfg file for DA server is in the $SYBASE/DA-15_5/server/*instance* directory.

where *instance* is the name of your DA server.

Note: Restart DA server for any changes to take effect.

Table 26. DA Server Instance Properties

Property Name	Description and Values
da.codec.readBufferSize	Specifies the internal buffer size, measured in bytes, used to read row data from a data input (stream). Valid values are: • Minimum: 16 • Maximum: 16777216 (16 MB) Default: 8192 This property applies to DA server and DA agents.

Property Name	Description and Values
`da.codec.writeBufferSize`	Specifies the internal buffer size, measured in bytes, used to write row data to the data output (stream). Valid values are: • Minimum: 16 • Maximum: 16777216 (16 MB) Default: 8192 This property applies to DA server and DA agents.
`da.dasd.conn.poolSize`	Specifies the maximum number of concurrent database connections DA server allows to the DASD internal database. Access to the database is blocked (wait and retry) if this limit is reached. Valid values are: • Minimum: 1 • Maximum: 2147483647 Default: 5
`da.dasd.dbname`	Specifies the name for the DASD internal database. Default: none (value required)
`da.dasd.port`	Specifies the port number for the DASD database server. Valid values are: • Minimum: 1025 • Maximum: 65535 Default: none (value required).

Property Name	Description and Values
`da.sec.dasd.enableRemoteAccess`	Specifies whether the DASD database server should accept network connections. When set to false, the DASD accepts connections only from clients running within the same Java Virtual Machine (JVM) as itself. Valid values are true or false. Default: none (value required).
`da.dasd.user`	Defines the login name of the DASD user. Default: none (value required).
`da.jdbc.derby.embedded`	Specifies the JDBC driver class name for connecting to the DASD internal database. Default: `org.apache.derby.jdbc.EmbeddedDriver`.
`da.rmi.port`	Specifies the port number at which DA server and DA agent accept remote method invocation (RMI) connections. Valid values are: • Minimum: 1025 • Maximum: 65535 Default: none (value required). This property applies to DA server and DA agents.
`da.sec.enableAudit`	Determines whether audit messages are printed to the *instance* log file. Valid values are true or false. Default: false. This property applies to DA server and DA agents.

Property Name	Description and Values
da.sec.passwdMaxLength	Specifies the maximum length of the da_admin user's password. Valid values are: • Minimum: 0 • Maximum: 64 Default: 30 This property applies to DA server and DA agents.
da.sec.passwdMinLength	Specifies the minimum length of the da_admin user's password. Valid values are: • Minimum: 0 • Maximum: 64 Default: 6 This property applies to DA server and DA agents.
da.sec.rmi.enableRemoteAccess	Determines whether the RMI service can accept connections from remote clients. Valid values are true or false. Default: false. This property applies to DA server and DA agents.
da.sec.roleMaxUsers	Specifies the maximum number of users per DA role. Default: none (value required). This property applies to DA server and DA agents.
da.sec.sessionTimeoutMinutes	Specifies the length of time an RMI session remains idle before it expires. Default: none (value required). This property applies to DA server and DA agents.

Property Name	Description and Values
`da.sec.tds.enableRemoteAccess`	Determine whether the TDS listener can accept connections from remote **isql** clients. Valid values are true or false. Default: false. This property applies to DA server and DA agents.
`da.sysam.license.dir`	Specifies the path to the SySAM license directory. Default: `SYBASE/SYSAM-2_0/licenses`.
`da.sysam.license.heartbeat.delay`	Specifies the number of seconds to wait before starting the heartbeat thread that ensures the license is still valid. Valid values are: • Minimum: 1 second • Maximum: 5 minutes Default: 60 seconds.
`da.sysam.license.heartbeat.interval`	Specifies the number of seconds the heartbeat thread waits before rechecking the license. Valid values are: • Minimum: 1 second • Maximum: 2 hours Default: 60 seconds.
`da.sysam.license.type`	Specifies the SySAM license type. Default: null.

Property Name	Description and Values
`da.tds.port`	Specifies the port number TDS connections. The **isql** command line tool uses this port number for TDS communication. Valid values are: • Minimum: 1025 • Maximum: 65535 Default: none (value required). This property applies to DA server and DA agents.
`da.tds.tracing`	Determines whether the TDS connection handler prints tracing (debug) information. Valid values are true or false. Default: false This property applies to DA server and DA agents.

See also

- *Data Assurance Agent Configuration Properties* on page 172

Remote Data Assurance Agent Command Reference

You can execute remote DA agent commands with **isql** or the SAP 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.

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 _.

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 49
- *Password Policy* on page 185

role

Maps LDAP users to the DA administrator role.

Syntax

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

Parameters

- **rolename –** case-sensitive role name with optional wildcards.
- **username –** case-sensitive user name.

session

View and modify session parameters.

Syntax

```
session [parameter_name [parameter_value]]
```

Parameters

- **parameter_name –** the name of a session parameter.
- **parameter_value –** the parameter value to set.

Table 27. Parameters

Name	Description
max_rows	The maximum number of rows to show in the result. Valid values: 1 to 2147483647. Default: 100.
wildcard_ignore_case	Determines whether wildcard matching should be case-insensitive. Valid values are true or false. Default: false.

show connection

Shows the database connections for a DA agent.

Syntax

```
show connection
```

Examples

- **Example 1 –** shows the 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 DA agent.

Syntax

```
show dts
```

Examples

- **Example 1 –** shows all the DTS information for a DA 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
------------------
-------------------------------------------------------------------------
```

```
----------- ---------- ----------------------------------
SAP Java Server VM Jan 14 2014 23:45:56 - 71_REL - optU - windows amd64 -
6 - bas2:200241 (mixed mode) SAP AG    Java 1.7.0_25, VM 7.1.011 23.5-b11

JVM TOTAL MEM JVM FREE MEM JVM MAX MEM
------------- ------------ -----------
33.3 MB       19.4 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 DA agent.

Syntax

```
show task
```

Examples

- **Example 1 –** shows all the tasks for the 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.

parameter_name	parameter_value
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 SAP Replication Server Data Assurance Option.

Syntax

```
version
```

Examples

- **Example 1 –** shows the version:

  ```
  version
  go
  ```

  The returned result is:

```
VERSION
-------------------------------------------------------------------------
-------------------------------------------------------------------------
------
SAP Replication Server Data Assurance Option - DA Agent/15.7.1/SP200/P/
generic/generic/damain/610/VM: SAP AG 1.7.0_25/OPT/Fri 31 Jan 2014
05:29:42 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 28. DA Agent Reserved Words

	Words
A	add
C	config, connection
D	dts, drop
J	jvm
P	password
R	role
S	session, show, shutdown, sslconfig, system

	Words
T	task, trace
U	user
V	version

Data Assurance Agent Configuration Properties

Use the *instance*.cfg configuration file to set the DA agent system properties.

The *instance*.cfg file for DA agent is found in the $SYBASE/DA-15_5/agent/*instance* directory.

where *instance* is the name of your DA agent.

Note: Restart DA agent for any changes to take effect.

Table 29. DA Agent Instance Properties

Property Name	Description and Value
da.agent.colHashPrefix	Specifies the prefix for each column_hash column alias in the SQL **select** statements. The prefix is followed by the name of the column. For example: `select … hashbytes('md5', mycol, using msb) as sy_col_hash_mycol` The prefix must be 1 – 100 characters in length and contain only letters, digits, and underscores. Default: sy_col_hash_ This property applies to the DA server's local agent and remote DA agents.

Property Name	Description and Value
`da.agent.maxDbHashArgs`	Specifies the maximum number of arguments DA supplies to the database hash function (SAP Adaptive Server **hashbytes**). If the number of comparison columns with the `row_hash` option exceeds this number, the comparison columns are divided into groups, and multiple row hashes are generated and compared. Valid values are: • Minimum: 2 • Maximum: 450 Default: 100 This property applies to the DA server's local agent and remote DA agents.
`da.agent.rowHashPrefix`	Specifies the prefix DA uses for each `row_hash` column alias in SQL **select** statements. The prefix is followed by the `row_hash` number, starting with 1. For example: `select … hashbytes(…)as sy_row_hash_1` The prefix must be 1 – 100 characters in length and contain only letters, digits, and underscores. Default: sy_row_hash_ This property applies to the DA server's local agent and remote DA agents.
`da.dts.port`	Specifies the port number for the remote DA agents to accept data transfer stream (DTS) connections from DA server instances. Valid values are: • Minimum: 1025 • Maximum: 65535 Default: none (value required)

Property Name	Description and Value
`da.jdbc.asa`	Specifies the JDBC driver class name for SAP Adaptive Server Anywhere (ASA). Default: `com.sybase.jdbc4.jdbc.SybDriver` This property applies to the DA server's local agent and remote DA agents.
`da.jdbc.ase`	Specifies the JDBC driver class name for an SAP Adaptive Server database. Default: `com.sybase.jdbc4.jdbc.SybDriver` This property applies to the DA server's local agent and remote DA agents.
`da.jdbc.hanadb`	Specifies the JDBC driver class name for the SAP HANA database. Default: `com.sap.db.jdbc.Driver` This property applies to the DA server's local agent and remote DA agents.
`da.jdbc.iq`	Specifies the JDBC driver class name for an SAP IQ database. Configure this to `ianywhere.ml.jdbcodbc.jdbc3.IDriver` to use the native SQL Anywhere® ODBC driver. Default: `com.sybase.jdbc4.jdbc.SybDriver` This property applies to the DA server's local agent and remote DA agents.
`da.jdbc.oracle`	Specifies the JDBC driver class name for an Oracle database. Default: `oracle.jdbc.driver.OracleDriver` This property applies to the DA server's local agent and remote DA agents.

Property Name	Description and Value
`da.sec.dts.enableRemoteAccess`	Determine whether the data transfer stream (DTS) listener can accept connections from remote clients. Valid values are true or false. Default: false

See also

- *Data Assurance Server Configuration Properties* on page 157

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 164
- *role* on page 148

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 30. 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 RMI or 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 (TLS) to remote method invocation (RMI) and Data Transfer Socket (DTS) communication.

Use SSL for encrypting:

- RMI communication between the SAP Control Center (SAP SCC) 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 SAP SCC Data Assurance Plug-In to DA Server RMI Communication

Configure the DA server and the SAP Control Center (SAP 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 SAP 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 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 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
```

Default: none (Value required)

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

c) Restart DA server.

SAP Adaptive Server and DA JDBC Communication Using SSL

Configure the DA server to use SSL to encrypt SAP Adaptive Server communication.

- Use **create connection** to create a DA connection for SAP Adaptive Server with the SAP jConnect for JDBC `SYBSOCKET_FACTORY` connection property for SSL communication.
- Use **sslconfig** to configure DA server or DA agent to use the SAP Adaptive Server certificate.

See *Connection Properties* in the *SAP jConnect for JDBC 16.0 Programmers Reference*.

Configuring DA Server to Use SSL for JDBC Communication

Configure the DA server to use SSL to encrypt JDBC communication with an SAP Adaptive Server.

1. Configure a truststore.

   If you have already enabled SSL for DA server to DA agent JDBC communication, you can skip this step.

   This example shows how to configure a truststore:

   ```
   sslconfig truststore "C:/ssl/truststore.ks"
   go
   sslconfig truststore_passwd openSesame
   go
   ```

   Note: Enter Windows directory paths using forward slashes instead of backslashes.

2. Add an SAP Adaptive Server certificate to a DA truststore.
   For example:

   ```
   %JAVA_HOME%\bin\keytool -importcert -alias myAlias
   -file %SYBASE%\ASE-15_0\certificates\myase.crt
   -keystore "C:\ssl\truststore.ks"
   -storepass openSesame
   ```

where:

- `%JAVA_HOME%\bin\keytool.exe` – is the Java **keytool** location on Windows. Java **keytool** is available in all Java Development Kits (JDKs).

  Note: Use the Java **Keytool** to add the SAP Adaptive Server certificates into the DA truststore. If the truststore does not already exist, it is created.

- `%SYBASE%\ASE-15_0\certificates\myase.crt` – is an SAP Adaptive Server certificate location.
- `C:\ssl\truststore.ks` – is the keystore flat file absolute path.

Select **Yes** when prompted to trust the certificate.

3. Restart DA server.
4. To create a DA connection for an SAP Adaptive Server database using the SAP jConnect for JDBC `SYBSOCKET_FACTORY` connection property, enter:

```
create connection instance_ssl
    set agent agent_name
and set type ASE
and set host host_name
and set port port_number
and set database database_name
and set user user_name
and set password password
with properties
 set SYBSOCKET_FACTORY 'com.sybase.da.jdbc.AseSslSocketFactory'
go
```

See also

- *create connection* on page 58
- *sslconfig* on page 152

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.

   SAP 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 133
- *config* on page 163

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 133

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 190

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 31. Row Comparison Considerations

Factor	Explanation
Hash types	We recommend 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.

SAP Adaptive Server Connection Fails

Problem: SAP Adaptive Server connection details are configured correctly, but DA fails to establish a connection.

Possible cause: SAP Adaptive Server is configured to require all clients to use password encryption. If the **net password encryption reqd** parameter is configured to a nonzero value, login fails.

Solution: Ensure that the SAP Adaptive Server connection has the `ENCRYPT_PASSWORD` property set to true. Add this property to existing connections by issuing:

```
alter connection myAseConn
and set ENCRYPT_PASSWORD = true
go
```

See *Connection Properties* in the *SAP jConnect for JDBC 16.0 Programmers Reference*.

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.
- If the **comparer_scale_rounding** value is too low, two unequal decimal numbers may be considered equal and the difference is not detected.

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

Note: Reconciliation cannot fix false differences.

You may also be able to avoid false differences by lowering the **comparer_scale_rounding** configuration parameter to allow DA to consider two unequal, yet sufficiently similar decimal numbers to be considered equal.

For example, for 3.141592654 to equal 3.1415926535897, lower the scale rounding value from the default value of 10 to 9.

See also

- *config* on page 133

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 Detect Differences In LOB Column

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 85
- *config* on page 133

Comparison Uses A Single Partition

Problem: Comparison used only a single partition.

Possible cause: When you run the comparison for the first time and no other comparison uses the same compareset, there are no boundary samples for the compareset.

Solution: Re-run the comparison.

See also

- *show boundary* on page 83

Comparison Considers Two Distinct Values Equal

Problem: Comparison uses trailing zeroes when comparing `binary` and `varbinary` datatypes.

Possible cause: When two binary columns are being compared, one column value has one or more trailing zero bytes (0x00) the other column value does not. DA always ignores trailing zero bytes to enable padded `binary` values to match `varbinary` values.

Solution: There is no workaround for this problem.

Glossary

Definitions of terms related to SAP Replication Server Data Assurance Option.

SAP Adaptive Server – The version 11.5 and later relational database server. If you choose the Replication Server System Database (RSSD) option when configuring SAP Replication Server, SAP Adaptive Server maintains Replication Server system tables in the RSSD database.

DA agent – Data Assurance (DA) agent

A component that 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 profile – 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.

DASD – Data Assurance System Database

The DA server database that stores system and configuration settings.

DTS – Data Transfer Stream

An application protocol used by DA agent during a comparison to stream data.

JDBC – Java Database Connectivity

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 SAP 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.

LDAP –

Lightweight Directory Access Protocol

Is an application protocol for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network.

LAN – local area network

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.

RMI – Remote Method Invocation

Is a remote procedure call used for communication between DA server and DA agents.

SAP Replication Server – The SAP server program that maintains replicated data, typically on a LAN, and processes data transactions received from other SAP Replication Servers on the same LAN or on a WAN.

SAP Replication Server Data Assurance Option – The SAP server program that compares row data and schema between two or more databases, and reports discrepancies.

RSSD – Replication Server System Database

The SAP Adaptive Server Enterprise (SAP ASE) database containing an SAP Replication Server system tables. The user can choose whether to store Replication Server system tables on SAP ASE or embedded in an SAP SQL Anywhere database hosted by SAP Replication Server.

row comparison job – A job used for row comparison.

schema – The structure of the database.

schema comparison job – A job used for comparing database object schemas.

SSL – Secure Sockets Layer

An industry standard, cryptographic protocol for transmitting data securely over the Internet. See also *TLS*.

TDS – Tabular Data Stream™

An application protocol by which Open Client™ and Open Server™ applications exchange information.

TLS – Transport Layer Security

An industry standard protocol, secures client/server communications using digital certificates and public-key cryptography. Transport layer security enables encryption, tamper detection, and certificate-based authentication. See also *SSL*.

WAN – wide-area network

a system of local area networks connected together with data communication lines.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

O

P

R

S

T

U

V

W