This is an overview of commands and information related to Sybase Replication Server. All information applies to RepServer 11.5 or later, replicating between ASE servers, and to a Unix environment, except where indicated otherwise. Functionality that is new in, or specific to, a RepServer version is indicates with a version number, for example **(12.0)**.

Last updated: 24 June 2001. This document can be found at www.sypron.nl; to be notified of any updates, subscribe to the mailing list at www.sypron.nl/notify.html.

Contents

Abbreviations

Some abbreviations which are used in this Quick Reference Guide:

ASE	Adaptive Server Enterprise
DSI	DataServer Interface - RS thread for a connection to a replicate database
DIST	Distributor - RS thread per primary database
LTM	Log Transfer Manager; synonym for RepAgent
RS, RepServer	Replication Server
RepAgent	Replication Agent (in primary dataserver)
RepDef	Replication Definition
RSI	Replication Server Interface - RS thread related to a route to another RepServer
RSM	Replication Server Manager (a DBA tool normally used in combination with Sybase Central)
RSSD	Replication Server System Database
SQM	Stable Queue Manager - RS thread managing a queue
SQT	Stable Queue Manager - RS thread assembling transactions

Replication Server Commands

The following commands can only be executed in Replication Server.

/* a comment between delimiters */
-- a comment until the end of this line
Two styles of commenting; same as in ASE.

abort switch *for* *logical_dataserver.logical_database*
Aborts the **switch active** command, unless RepServer has already gone too far in the switch process to abort it (the **switch active** command changes the active database in a warm standby application).

activate subscription *sub_name* ***for*** *{ table_rep_def | function_rep_def |* ***publication*** *pub_name* ***with primary at*** *dataserver.database } }* ***with replicate at*** *dataserver.database [* ***with suspension*** *[* ***at active replicate only*** *]]*
Starts the distribution of updates from the primary database to the replicate database for a subscription. The subscription status is set to ACTIVE. This command is part of the bulk materialization process, or part of the process of refreshing a publication subscription.

add partition *logical_name* ***on*** *'physical_name'* ***with size*** *<n> [* ***starting at*** *vstart]*
Adds a partition of size <n> Mb to RepServer. This can be a raw partition or an OS file (if a file, it must already exist). Note: use single quotes around *physical_name*.

admin disk_space
Displays utilization (in 1Mb segments) of each disk partition known by RepServer.

admin echo, *character_string*
Returns the string entered by the user (to check if local RepServer is running).

admin get_generation , *dataserver, database*
Retrieves the generation number for a primary database.

admin health
Displays the status of the RepServer.

admin log_name
Displays the path to the current log file

admin logical_status *[, logical_dataserver, logical_database]*
Displays status information for logical connections.

admin pid
Displays the process ID of the RepServer.

admin quiesce_check
Determines if the RepServer is currently quiescent (i.e. all messages have processed and there are no open transactions).

admin quiesce_force_rsi
Determines whether a RepServer is currently quiescent and forces the RepServer to deliver and obtain acknowledgements for messages in RSI queues.

admin rssd_name
Displays the names of the data server and database for the RSSD.

admin security_property *[, mechanism_name]*
Displays information about supported network-based security mechanisms/ services.

admin security_setting *[, rs_idserver* |, *rs_server* |, *dataserver.database]*
Displays network-based security parameters and values for RepServer.

admin set_log_name , *log_file*
Closes the existing RepServer log file and opens a new log file.

admin show_connections
Displays information about all connections from the RepServer to data servers and to other RepServers.

admin show_function_classes
Displays the names of existing function-string classes and their parent classes, and indicates the number of levels of inheritance.

admin show_route_versions
Displays the version number of routes that originate or terminate at the RepServer.

admin show_site_version
Displays the site version of the Replication Server.

admin sqm_readers , *q_number , q_type*
Displays the read and delete points of the threads that are reading a stable queue.

(RS 11.5)
admin statistics , *{* ***mem*** | ***md*** | ***reset*** *}*
(RS 12.0)
admin statistics , *{* ***mem*** | ***md*** | ***dsi*** | ***reset*** *}*
Displays statistics about memory use (**mem**), message delivery (**md**), and replicated transactions (**dsi**) by the current RepServer. **reset** resets the statistics counters.

(RS 12.0)
admin translate , *value, source_datatype, target_datatype*
Converts *value* from the specified *source_datatype* to *target_datatype* (used for testing/troubleshooting data conversion issues in heterogenous replication systems).

admin version
Displays software version information about the currently running RepServer.

admin who *[,* ***dist*** | ***dsi*** | ***rsi*** | ***sqm*** | ***sqt*** *]*
Displays information about the various types of threads running in the RepServer.

Queues are identified as <db-nr>:<something>, where <something> is one of:

- <positive 8-digit number> is a route
- <negative 8-digit number> is a materialization queue
- <0> is an outbound queue (going a route or a replicate dataserver)
- <1> is an inbound queue (coming from a RepAgent)

admin who_is_down
Displays information about RepServer threads that are not running.

admin who_is_up
Displays information about RepServer threads that are running.

allow connections
First, start RepServer in stand-alone mode and execute **set log recovery** for each database whose logs will be replayed, then execute **allow connections** to begin replaying log records from reloaded dumps.
After executing **allow connections**, RepServer accepts connect requests only from RepAgents started in recovery mode for the specified databases. This ensures that RepServer receives the replayed log records before current transactions.

alter connection *to* *dataserver.database*
{ ***set function string class*** *[* ***to*** *] function_class* |
set error class *[* ***to*** *] error_class* | ***set password*** *[* ***to*** *] passwd* |
set log transfer *[* ***to*** *] {* ***on*** | ***off*** *}* | ***set*** *database_param [* ***to*** *] 'value'* |
set *security_param [* ***to*** *] 'value'* | ***set security_services*** *[* ***to*** *]* ***'default'*** *}*
Modifies attributes for this database connection. For values for *database_param* and *security_param*, see **configure replication server. set security_services to 'default'** resets all network security features to "not required" for this connection.

alter function *rep_def.function*
add parameters *@param datatype [,...]*
Adds parameters to a user-defined function. Note: this is old (RepServer version 10) functionality which is still supported but has been replaced by **alter function replication definition**.

alter function replication definition *function_rep_def { **deliver as** ' proc_name'* |
add *@param datatype [,...]* | ***add searchable parameters*** *@param [,...]* | ***send standby*** *{* ***all*** | ***replication definition*** *} **parameters** }*
Modifies an existing function RepDef.

alter function string *[rep_def.]function[;function_string]*
for function_class *[* ***scan*** *'input_template']*
[***output*** *{* ***language*** *'lang_output_template'* |
rpc 'execute *procedure_name [@param =] { constant* | ***?****variable****!****modifier****?*** *} [,...] '* |
writetext *[* ***use primary log*** | ***with log*** | ***no log*** *]* | ***none*** *}]*
Replaces an existing function string. To restore the default function string: **alter function string MyRepDef.***function* **for** *function_class* (only for function class **rs_sqlserver_function_class**). For modifiers, see **create function string**.

alter function string class *function_class **set parent to** { parent_class* | ***null*** *}*
Alters a function-string class, specifying whether it should be a base or derived class.

alter logical connection *to* *logical_dataserver.logical_database*
{ ***set distribution*** *{* ***on*** | ***off*** *}* |
set *logical_database_param* ***to*** *'value'* |
set send_truncate_table to *{* ***on*** | ***off*** *} }*
Alters a logical connection. For *logical_database_param*, see **configure replication server. send_truncate_table** applies to replication of **truncate table** commands in a warm standby database.

(RS 11.5)
alter replication definition *rep_def*
{ ***with replicate table named*** *[owner_name.]'table_name'* |
add *column_name [* ***as*** *replicate_column_name] [datatype [* ***null*** | ***not null*** *]] [,...]* |
alter columns with *column_name* ***as*** *replicate_column_name [,...]* |
add searchable columns *column_name [,...]* |
send standby *[* ***off*** | *{* ***all*** | ***replication definition*** *}* ***columns*** *]* |
replicate *{* ***minimal*** | ***all*** *}* ***columns*** |
replicate_if_changed *column_name [,...]* |
always_replicate *column_name [,...] }*

(RS 12.0)
alter replication definition *rep_def*
{ ***with replicate table named*** *[owner_name.]'table_name'* |
add *column_name [* ***as*** *replicate_column_name] [datatype [* ***null*** | ***not null*** *]] [* ***map to*** *published_datatype] [,...]* |
alter columns with *column_name [* ***as*** *replicate_column_name] [datatype [* ***null*** | ***not null*** *]] [* ***map to*** *published_datatype] [,...]* |
add primary key *column_name [,...]* |
drop primary key *column_name [,...]* |
add searchable columns *column_name [,...]* |
drop searchable columns *column_name [,...]* |
send standby *[* ***off*** | *{* ***all*** | ***replication definition*** *}* ***columns*** *]* |
replicate *{* ***minimal*** | ***all*** *}* ***columns*** |
replicate_if_changed *column_name [,...]* |
always_replicate *column_name [,...] }*
Modifies an existing RepDef. Note 1: **null** and **not null** apply only to **text**, **image** or **rawobject** columns. Note 2: customised function strings are not automatically altered when columns are added to a RepDef; this must be done using **alter function string**. Note3: **replicate minimal columns** cannot be used when also using custom function strings (except when using *rs_default_fs* for **rs_update** & **rs_delete**).

alter route *to* *remote_replication_server*
{ ***set next site*** *[* ***to*** *] next_replication_server* |
set username *[* ***to*** *] 'user'* ***set password*** *[* ***to*** *] 'passwd'* | ***set password*** *[* ***to*** *] 'passwd'* | ***set*** *route_param [* ***to*** *] 'value'* | ***set*** *security_param [* ***to*** *] 'value'* |
set *security_services [* ***to*** *] 'default' }*
Modifies attributes of a route from the current RepServer to a replicate RepServer. For *route_param* and *security_param*, see **configure replication server.**

alter user *user **set password** { new_passwd* | ***null*** *} [* ***verify password*** *old_passwd]*
Modifies a user's password.

assign action *{* ***ignore*** | ***warn*** | ***retry_log*** | ***log*** | ***retry_stop*** | ***stop_replication*** *} **for** error_class **to** dataserver_error [, dataserver_error]...*
Assigns RepServer error-handling actions to data server errors received by the DSI thread.

check publication *pub_name **with primary at** dataserver.database*
Finds the status of a publication and the number of articles the publication contains.

check subscription *sub_name*
for *{ table_rep_def* | *function_rep_def* | *{ article_name* ***in*** *pub_name* | ***publication*** *pub_name } **with primary at** dataserver.database } **with replicate at** dataserver.database*
Finds the materialization status of a subscription to a table of function RepDef, all articles of a publication, or a specific article of a publication.

configure connection …
This command is identical to **alter connection.**

configure logical connection …
This command is identical to **alter logical connection.**

configure replication server
{ ***set*** *repserver_param* ***to*** *'value'* | ***set*** *route_param* ***to*** *'value'* |
set *database_param* ***to*** *'value'* | ***set*** *logical_database_param* ***to*** *'value'* |
set *security_param* ***to*** *'value'* | ***set*** *id_security_param* ***to*** *'value'* |
set security_services to 'default' |
(RS 12.0) ***set ha_failover to*** *{* ***'on'*** | ***'off'*** *} }*
Sets configuration parameters of this RepServer. These are stored in the **rs_config** table in the RSSD and can be viewed (and modified) using the ASE stored procedure **rs_configure**. **set security_services to 'default'** resets all network security features to '**not required**'.
repserver_param can be the following:

- ***cm_max_connections*** *(default=64)*
- ***current_rssd_version***
- ***fstr_cachesize*** *(default=200000 (bytes))*
- ***id_server***
- ***init_sqm_write_delay*** *(default=1000 (milliseconds))*

- ***init_sqm_write_max_delay*** *(default=10000 (milliseconds))*
- ***memory_limit*** *(default=20 (Mb))*
- ***minimum_rssd_version***
- ***num_client_connections*** *(default=30)*
- ***num_concurrent_subs*** *(default=10)*
- ***num_msgqueues*** *(default=178)*
- ***num_msgs*** *(default=45568)*
- ***num_mutexes*** *(default=128)*
- ***num_stable_queues*** *(default=32) (HP only)*
- ***num_threads*** *(default=50)*
- ***oserver***
- ***password_encryption*** *(default=0)*
- ***prev_min_rssd_version***
- ***prev_rssd_version***
- ***queue_dump_buffer_size*** *(default=1000 (bytes))*
- ***rec_daemon_sleep_time*** *(default=2 (minutes))*
- ***rssd_error_class*** *(default=rs_sqlserver_error_class)*
- ***sqm_warning_thr1*** *(default=75 (percent))*
- ***sqm_warning_thr2*** *(default=90 (percent))*
- ***sqm_warning_thr_ind*** *(default=70 (percent))*
- ***sqt_max_cache_size*** *(default=1048576 (bytes))*
- ***sre_reserve*** *(default=0 (percent))*
- ***sts_cachesize*** *(default=100 (#rows cached))*
- ***sub_daemon_sleep_time*** *(default=120 (seconds))*

route_param can be the following:
- ***rsi_batch_size*** *(default=262144 (bytes))*
- ***rsi_fadeout_time*** *(default=-1)*
- ***rsi_packet_size*** *(default=2048 (bytes))*
- ***rsi_sync_interval*** *(default=60 (seconds))*
- ***save_interval*** *(default=0 (minutes))*

database_param can be the following:
- ***batch*** *(default='on')*
- ***batch_begin*** *(default='on')*
- ***command_retry*** *(default=3 (#retries))*
- ***db_packet_size*** *(default=512 (bytes))*
- ***dsi_charset_convert*** *(default='on')*
- ***dsi_cmd_batch_size*** *(default=8192 (bytes))*
- ***dsi_cmd_separator*** *(default='\n')*
- ***dsi_exec_request_sproc*** *(default='on')*
- ***dsi_fadeout_time*** *(default=600 (seconds))*
- ***dsi_keep_triggers*** *(default='on'; except for standby databases)*
- ***dsi_large_xact_size*** *(default=100)*
- ***dsi_max_cmds_to_log*** *(default=-1)*
- ***dsi_max_text_to_log*** *(default=-1)*
- ***dsi_num_large_xact_threads*** *(default=0)*
- ***dsi_num_threads*** *(default=1)*
- ***dsi_replication*** *(default='off'; 'on' for active databases in warm standby)*
- ***dsi_serialization_method*** *(default='wait_for_commit')*
- ***dsi_sql_data_style*** *{ 'db2' | 'notes' | 'watcom' | 'sqlremote' } (default='' (for ASE)) Note: this option is obsolete in 12.0.*
- ***dsi_sqt_max_cache_size*** *(default=0)*
- ***dsi_xact_group_size*** *(default=65536 (bytes))*
- ***dsi_max_xacts_in_group*** *(default=20)*
- ***dump_load*** *(default='off')*
- ***parallel_dsi*** *(default='off')*
- ***save_interval*** *(default=0 (minutes))*

logical_database_param can be the following:
- ***materialization_save_interval*** *(default='strict' for standby databases)*
- ***save_interval*** *(default=0 (minutes))*

security_param can be the following:
- ***msg_confidentiality*** *(default='not_required')*
- ***msg_integrity*** *(default='not_required')*
- ***msg_origin_check*** *(default='not_required')*

- ***msg_replay_detection*** *(default='not_required')*
- ***msg_sequence_check*** *(default='not_required')*
- ***mutual_auth*** *(default='not_required')*
- ***security_mechanism*** *(default=first security item in libtcl.cfg)*
- ***unified_login*** *(default='not_required')*
- ***use_security_services*** *(default='off')*

id_security_param can be the following:
- ***id_msg_confidentiality*** *(default='not_required')*
- ***id_msg_integrity*** *(default='not_required')*
- ***id_msg_origin_check*** *(default='not_required')*
- ***id_msg_replay_detection*** *(default='not_required')*
- ***id_ msg_sequence_check*** *(default='not_required')*
- ***id_mutual_auth*** *(default='not_required')*
- ***id_security_mech*** *(default=first security item in libtcl.cfg)*
- ***id_unified_login*** *(default='not_required')*

configure route …
This command is identical to **alter route**.

create article *article_name* ***for*** *pub_name* ***with primary at*** *dataserver.database*
with replication definition *{ table_rep_def | function_rep_def }*
[***where*** *{ column_name | @param } { **<** | **>** | **>=** | **<=** | **=** | ***&*** } value [***and*** { column_name | @param } { **<** | **>** | **>=** | **<=** | **=** | ***&*** } value]...]*
[***or where*** *{ column_name | @param } { **<** | **>** | **>=** | **<=** | **=** | ***&*** } value [***and*** { column_name | @param } { **<** | **>** | **>=** | **<=** | **=** | ***&*** } value]...]*
Creates an article for a table or function replication definition and specifies the publication that is to contain the article. Using **create article** automatically invalidates the publication the article is for. You cannot create new subscriptions until you validate the publication (using **validate publication**). You cannot replicate data for the new article until you refresh the publication subscription (using **create** (or **define/activate/validate**) **subscription…for new articles**).
Note that the **or** keyword is not part of the WHERE-clause, but is used to join different WHERE-clauses.

create connection *to dataserver.database*
set error class *[* ***to*** *] error_class* ***set function string class*** *[* ***to*** *] function_class*
set username *[* ***to*** *] user* ***set password*** *[* ***to*** *] passwd*
[***set*** *database_param [* ***to*** *] 'value'* ***set*** *security_param [* ***to*** *] 'value']*
[***with*** *{* ***log transfer on***, ***dsi_suspended*** *}]*
[***as active for*** *logical_dataserver.logical_database |*
as standby for *logical_dataserver.logical_database [* ***use dump marker*** *]]*
Adds a database to the replication system. To create a connection for an ASE database, best use the **rs_init** program which performs some additional initialisation actions (see the RepServer Installation Guide). When creating the connection for a standby using the dump marker, first run the **create connection… use dump marker** command, and then dump the database (not the other way around).
The username specified is the maintenance user for this database. For *database_param* and *security_param*, see **configure replication server**.

create error class *error_class*
Creates an error class. The *rs_sqlserver_error_class* is the default error class provided for ASE databases. After creating the error class, use the **rs_init_erroractions** stored procedure to initialize the error class.

create function *rep_def.function* ***(*** *[@param datatype [,...]]* ***)***
Creates a user-defined function. For more information, see the RepServer Administration Guide. Note: this is old (RepServer version 10) functionality which is still supported but has been replaced by **create function replication definition**.

create function replication definition *function_rep_def*
with primary at *dataserver.database [* ***deliver as*** *'procedure_name']*
(*[@param datatype [,...]]* ***)***
[***searchable parameters (*** *@param [,...]* ***)*** *]*
[***send standby*** *{* ***all*** *|* ***replication definition*** *}* ***parameters*** *]*
Creates a function RepDef for a stored procedure that is to be replicated (Syntax note: the brackets are required even if the procedure has no parameters).

create function string *[rep_def.]function[;function_string]*

***for** function_class [**with overwrite**] [**scan** 'input_template']*
*[**output** { **language** 'lang_output_template' |*
***rpc 'execute** procedure_name [@param =] { constant | **?**variable**!**modifier**?** } [,...] '* |
***writetext** [**use primary log** | **with log** | **no log**] | **none** }]*
Adds a function string to a function string class. RepServer uses function strings to generate instructions for data servers.
Possible modifiers are:

- *new, new_raw* : the new value of a column being inserted or updated
- *old, old_raw* : the existing value of a column being inserted or updated
- *user, user_raw* : a variable defined in the input template of an *rs_select* or *rs_select_with_lock* function string
- *sys, sys_raw* : a system-defined variable
- *param, param_raw* : a function parameter
- *text_status* : status for **text**, **image** or **rawobject** data. Possible values are:
  - 0x0000 - Text field contains NULL value, and text pointer not initialized.
  - 0x0002 - Text pointer is initialized.
  - 0x0004 - Real text data will follow.
  - 0x0008 - No text data will follow because the text data is not replicated.
  - 0x0010 - The text data is not replicated but it contains NULL values.

create function string class *function_class [**set parent to** parent_class]*
Creates a function string class. Note: you cannot use *rs_sqlserver_function_class* as a parent class. You can use *rs_default_function_class* or *rs_db2_function_class* as parent classes.

create logical connection ***to** dataserver.database*
Creates a logical connection, which is required for warm standby applications. The logical connection will be mapped to a physical connection; when no physical connection exists yet, *dataserver* and *database* can be fictitious names; otherwise, the logical connection must have the same name as the existing physical connection.

create publication *pub_name **with primary at** dataserver.database*
Creates a publication for tables or stored procedures that are to be replicated as a group to one or more subscribing replicate databases.

create replication definition *rep_def*
***with primary at** dataserver.database*
*[**with all tables named** [owner_name.]'table_name' |*
*[**with primary table named** [owner_name.]'table_name']*
*[**with replicate table named** [owner_name.]'table_name']]*
(** column_name [**as** replicate_column_name] [datatype [**null** | **not null**]] [,...] **)
primary key (** column_name [,...] **)
*[**searchable columns (** column_name [,...] **)**]*
*[**send standby** [{ **all** |**replication definition** } **columns**]]*
*[**replicate** { **minimal** | **all** } **columns**]*
*[**replicate_if_changed (** column_name [,...] **)**]*
*[**always_replicate (** column_name [,...] **)**]*
Creates a RepDef for a table that is to be replicated. Note: the clauses **replicate_if_changed** and **always_replicate** apply to **text**, **image** or **rawobject** columns only. Note2: **replicate minimal columns** cannot be used when also using custom function strings (except when using *rs_default_fs* for **rs_update** & **rs_delete**).

create route ***to** remote_rep_server { **set next site** [**to**] intermediate_rep_server |*
*[**set username** [**to**] user] [**set password** [**to**] passwd]*
*[**set rsi_batch_size** [**to**] 'value'] [**set** route_param **to** 'value']*
*[**set** security_param **to** 'value'] }*
Designates the route for a connection from the current RepServer to a remote RepServer, optionally via an intermediate RepServer. For values for *route_param* and *security_param* , see **configure replication server.**

create subscription *sub_name **for** { table_rep_def | function_rep_def | **publication** pub_name **with primary at** dataserver.database }*
***with replicate at** dataserver.database*
*[**where** { column_name | @param } { **<** | **>** | **>=** | **<=** | **=** | **&** } value [**and** { column_name | @param } { **<** | **>** | **>=** | **<=** | **=** | **&** } value]...]*
*[**without holdlock** | **incrementally** | **without materialization**]*
*[**subscribe to truncate table**] [**for new articles**]*

Creates and initializes a subscription and materializes subscription data. The subscription may be for a table RepDef, function RepDef, or publication. Default materialization for table RepDefs is atomic materialization, i.e. using "select with holdlock" . For non-atomic materialization, specify the **without holdlock** option. Non-atomic materialization is always incremental (= transactions of 10 inserts).
Note: WHERE-clauses are not possible for subscriptions to publications.

create user *user **set password** { passwd | **null** }*
Adds a new user login name to a RepServer.

define subscription *sub_name **for** { table_rep_def | function_rep_def | **publication** pub_name **with primary at** dataserver.database }*
***with replicate at** dataserver.database*
*[**where** { column_name | @param } { **<** | **>** | **>=** | **<=** | **=** | **&** } value [**and** { column_name | @param } { **<** | **>** | **>=** | **<=** | **=** | **&** } value]...]*
*[**subscribe to truncate table**] [**for new articles**]*
Creates a subscription, but does not materialize or activate the subscription. The subscription may be for a table RepDef, function RepDef, or publication. This command is used for bulk subscription materialization, or refreshing a publication subscription. Note: WHERE-clauses are not possible for publication subscriptions.

drop article *article_name **for** pub_name **with primary at** dataserver.database [**drop_repdef**]*
Drops an article from a publication and optionally drops its RepDef (if it is not used elsewhere). Using **drop article** automatically invalidates the publication the article is for. You cannot create new subscriptions until you validate the publication (using **validate publication**).

drop connection ***to** dataserver.database*
Removes a primary or replicate database from the replication system. It does not affect primary or replicated data but only modifies the RSSD. Associated RepDefs and/or subscriptions should be dropped first.

drop error class *error_class*
Drops an error class and any actions associated with it.

drop function *[rep_def.]function*
Drops a function. Note: this is old (RepServer version 10) functionality which is still supported but has been replaced by "**drop function replication definition**".

drop function replication definition *function_rep_def*
Drops a function RepDef and its user-defined function.

drop function string *[rep_def.]function[;function_string* | ***;all**] **for** function_class*
Drops a function string for a function string class.

drop function string class *function_class*
Drops a function string class.

drop logical connection ***to** dataserver.database*
Drops a logical connection (used to manage warm standby applications).

drop partition *logical_name*
Removes a disk partition from the RepServer. It will be dropped as soon as it becomes unused.

drop publication *pub_name **with primary at** dataserver.database [**drop_repdef**]*
Drops a publication and all of its articles, and optionally drops the RepDefs for the articles (if they're not used elsewhere).

drop replication definition *rep_def*
Drops a RepDef and the corresponding function strings.

drop route ***to** remote_replication_server [**with nowait**]*
Closes the route to another RepServer. Before dropping a route, all subscriptions at the remote RepServer for primary data managed by the source RepServer, should be dropped. Also, any indirect routes that use the route should be dropped. The ***with nowait*** option should only be used as a last resort: the route will be closed, even if communication with the destination RepServer is not possible. In this case, you should usually execute the **sysadmin purge_route_at_replicate** command next.

drop subscription *sub_name*
for *{ table_rep_def | function_rep_def | { **article** article_name **in** pub_name | **publication** pub_name } **with primary at** dataserver.database }*
with replicate at *dataserver.database*
*[**without purge** [**with suspension** [**at active replicate only**]] | [**incrementally**] **with purge**]*
Drops a subscription to a table RepDef, function RepDef, article or publication. For a table RepDef or publication, a dematerialization method (with/without purge) must be specified.

drop user *user*
Drops a RepServer user login name.

grant *{ **sa** | **create object** | **primary subscribe** | **connect source** } **to** user*
Assigns permissions to a user. Possible permissions are:

sa	– everything permitted
create object	– can create RepDefs, subscriptions etc.
primary subscribe	– can only create subscriptions
connect source	– required for RepAgent logins & RepServer route logins

ignore loss ***from*** *dataserver.database [**to** { dataserver.database | rep_server }]*
Allows RepServer to accept messages after it has detected a loss.

move primary ***of*** *{ **error class** | **function string class** } class_name **to** rep_server*
Modifies the primary RepServer for an error class or function string class.

rebuild queues
Rebuilds RepServer stable queues. When ready, check the RepServer error log for lines indicating lost messages. When RepServer is running in single-user mode (started with the "-M" commandline option), and rebuild queues is executed, RepServer will only accept connections from RepAgents running in recovery mode. WARNING: this is a potentially dangerous operation with risk of loss of transactions; see the *RepServer Administration Guide* and/or *Troubleshooting Guide* for instructions on how to use this command.

resume connection ***to*** *dataserver.database [**skip transaction** | **execute transaction**]*
Resumes a suspended connection. The first transaction in the queue can be skipped with **skip transaction**, which is then written into the RSSD and can be viewed with **rs_helpexception.** The option **execute transaction** should only be used when a problem with a system transaction (with error message: "`There is a system transaction whose state is not known. DSI will be shut down`") causing the DSI suspension has now been fixed.

resume distributor *dataserver.database*
Resumes a suspended Distributor thread for a connection to a database.

resume log transfer ***from*** *{ dataserver.database | **all** }*
Allows RepAgents to connect to the RepServer.

resume route ***to*** *remote_replication_server*
Resumes a suspended route.

revoke *{ **sa** | **connect source** | **create object** | **primary subscribe** } **from** user*
Revokes permissions (which were previously assigned using **grant**) from a user.

route_upgrade *source_replication_server.remote_replication_server*
An RSM Server command that upgrades the specified route for use with RepServer 11.5 or later. Upgrading a route rematerializes the data in RSSD and makes information associated with new features available to a newly upgraded RepServer.

route_upgrade_recovery *source_replication_server.remote_replication_server*
An RSM Server command to recover a failed route upgrade.

route_upgrade_status
An RSM Server command to display routes that need to be upgraded or recovered.

set autocorrection *{ **on** | **off** } **for** rep_def **with replicate at** dataserver.database*

Enables (**on**) or disables (**off**, =default) autocorrection for a RepDef in a specific replicate database. With autocorrection, RepServer converts each update or insert operation at the replicate database into a delete followed by an insert.This is used to prevent errors caused by missing or duplicate rows in a copy of a replicated table. Autocorrection should only be enabled for RepDefs whose subscriptions use non-atomic materialization (**create subscription** command specified **without holdlock**). After materialization is complete and the subscription is VALID, you should disable autocorrection to improve performance.

set log recovery ***for*** *dataserver.database*
Places RepServer in log recovery mode for a database whose log is to be recovered from offline dumps. Execute *set log recovery* after restarting RepServer in stand-alone mode, next execute *allow connections* to enter recovery mode. RepServer accepts connections only from RepAgents started in recovery mode for databases named with *set log recovery*. This ensures that old log records are replayed before new log records are accepted.

set proxy *[**to**] [user_name [**verify password** passwd]]*
Switches to another RepServer user with all the permissions of the new user and none of the permissions of the original user.

shutdown
Shuts down a RepServer.

suspend connection ***to*** *dataserver.database [**with nowait**]*
Suspends a connection to a database.

suspend distributor *dataserver.database*
Suspends the Distributor thread for a connection to a database.

suspend log transfer ***from*** *{ dataserver.database | **all** }*
Disconnects a RepAgent from a RepServer and prevents an RepAgent from connecting. This is the first step in quiescing the replication system. Use **admin quiesce_check** to test whether the system is quiesced after suspending the RepAgents.

suspend route ***to*** *remote_replication_server*
Suspends a route to another RepServer.

switch active ***for*** *logical_dataserver.logical_database **to** dataserver.database [**with suspension**]*
Modifies the active database in a warm standby application. Note: the RepAgent for the new active database should also be started.

sysadmin apply_truncate_table *, replicate_dataserver, replicate_database, { owner_name | **''** }, table_name, { **'on'** | **'off'** }*
Turns the **subscribe to truncate table** option on or off for all existing subscriptions to a table, enabling or disabling replication of the **truncate table** ASE command. **sysadmin apply_truncate_table** should be executed at the replicate RepServer.

sysadmin dropdb *, dataserver, database*
Drops a database from the ID Server. Use only when **drop connection** fails.

sysadmin dropldb *, dataserver, database*
Drops a logical DB from the ID Server. Use onlywhen **drop logical connection** fails.

sysadmin drop_queue *, q_number , q_type*
Drops a stable queue. Can be used for dropping a failed materialization queue.

sysadmin droprs *, replication_server*
Drops a RepServer from the ID Server. Note that the corresponding RSSD must also be dropped with **sysadmin dropdb**.

sysadmin dump_file *[, file_name]*
Specifies an alternative log file name to use when dumping a RepServer stable queue when using **sysadmin dump_queue** next. Issuing **sysadmin dump_file** without a filename closes the log file previously opened.

sysadmin dump_queue *, q_number , q_type , seg , blk , cnt [, **RSSD** | **client**]*

Dumps the contents of a RepServer stable queue. To dump entire undeleted queue, " *-1,-1,-2"* for *"seg, blk, cnt";* specify *"-1,-2,-2"* to start at the first unread block. If **RSSD** or **client** is not specified, the output goes to RepServer log, unless changed with **sysadmin dump_file**.

sysadmin fast_route_upgrade *, remote_replication_server*
Updates the route to a remote RepServer to the version of the current RepServer. Upgrading a route rematerializes the data in RSSD and makes information associated with new features available to a newly upgraded RepServer.

sysadmin hibernate_off *[, string]*
Turns off hibernation mode for the RepServer and returns it to an active state. If *string* was specified with **sysadmin hibernate_on**, you must specify the same value. If you forgot the *string*, you can find it in the **rs_recovery** RSSD table with the following query: **select convert(char(255), text) from rs_recovery**

sysadmin hibernate_on *[, string]*
Turns on hibernation mode ("suspends") for the RepServer. This will stop other users from accessing the RepServer, so that system information (**admin**) and system administration (**sysadmin**) commands can be executed. You can use *string* (some valid string value) to ensure that no-one else accidentally turns off hibernation mode for the RepServer while you are working on it.

sysadmin log_first_tran *, dataserver, database*
Writes the first transaction in a DSI queue into the RSSD exceptions log (it can then be viewed using **rs_helpexception**). The transaction is not deleted from the queue.

sysadmin purge_all_open *, q_number , q_type*
Purges all open transactions from an inbound queue of this RepServer.

sysadmin purge_first_open *, q_number , q_type*
Purges the first open transaction from the inbound queue of this RepServer.

sysadmin purge_route_at_replicate *, remote_replication_server*
Removes all references to this RepServer from a remote RepServer.

sysadmin restore_dsi_saved_segments *, dataserver, database*
Restores backlogged transactions so they can be replayed into the database.

sysadmin set_dsi_generation *, gen_number , primary_dataserver, primary_database , replicate_dataserver, replicate_database*
Modifies a DSI generation number (stored in **RSSD..rs_exceptslast.origin_qid**) for a replicate DB to prevent the application of transactions in the DSI stable queue after a replicate database is restored. In order to activate the new DSI generation number, the DSI must be suspended and resumed.

sysadmin site_version *[, version]*
Sets the site version number for the current RepServer (for example: 1200). This allows use of software features in the corresponding release, and prevents from downgrading to an earlier release.

sysadmin sqm_purge_queue *, q_number , q_type*
Purges all messages from a stable queue. WARNING: Purging messages from a stable queue can result in data loss and should be used only with the advice of Sybase Technical Support. When you purge a queue, RepServer cannot send the purged messages to the destination database or RepServer, and this causes inconsistencies in the replication system. If a queue contains subscription marker messages or route messages, using this command can have severe consequences.

sysadmin sqm_unzap_command*, q_number , q_type , seg , blk , row*
Undeletes a message in a stable queue. These values can be found by using the admin who command, the **admin who, sqm** command, and the **admin who, sqt** command. seg - identifies the segment in the stable queue that contains the message to be undeleted. blk - identifies the 16K block in the segment. Block numbering starts at 1 and ends at 64. row - the row number in the block of the command to be undeleted.

sysadmin sqm_zap_command *, q_number , q_type , seg , blk , row*
Deletes a single message in a stable queue.

sysadmin sqt_dump_queue *q_number , q_type , reader [,* ***open*** *]*
Dumps the transaction cache for an inbound queue or a DSI queue.

sysadmin system_version *[, version]*
Executed in the ID server, displays or sets the system-wide version number for a replication system, allowing RepServers of different release levels to work together. Valid version numbers include 1100, 1102, 1103, 1150, 1200, and others (corresponding to RepServer releases). When setting this version number, all RepServers in the system must be at that version or a higher version.

trace *{* ***'on'*** *|* ***'off'*** *} ,* ***'dsi'*** *,* ***'dsi_buf_dump'***
Dumps commands sent to the DSI into RepServer errorlog (Note: this command is not documented or supported).

validate publication *pub_name* ***with primary at*** *dataserver.database*
Sets the status of a publication to VALID, allowing new subscriptions to be created for the publication.

validate subscription *sub_name* ***for*** *{ table_rep_def | function_rep_def |* ***publication*** *pub_name* ***with primary at*** *dataserver.database }*
with replicate at *dataserver.database*
For a subscription to a RepDef or a publication, sets the subscription status to VALID. This command is part of the bulk materialization process, or part of the process of refreshing a publication subscription.

wait for create standby ***for*** *logical_dataserver.logical_database*
A blocking command that allows a client session in the RepServer to wait for the standby database creation process to complete.

wait for switch *for logical_dataserver.logical_database*
A blocking command that allows a client session in the RepServer to wait for the switch to the new active database to complete.

ASE Commands and System Stored Procedures

The following commands and procedures can only be executed in ASE:

dbcc dbrepair *(database_name,* ***ltmignore*** *)*
Command that turns off replication for an offline ASE database that has a valid secondary truncation point. For cases where **dbcc settrunc** can't be used because the database is not accessible.

dbcc gettrunc
Command to retrieve RepAgent information about the current ASE database.

- *ltm_truncpage* The first page that is not truncated in the database log
- *ltm_trunc_state* One of the following values:
  - 1 - ASE does not truncate the log on or after the *ltm_truncpage*
  - 0 - ASE ignores the *ltm_truncpage*
- *db_rep_stat* A mask constructed of the following:
  - 0x01 - *ltm_truncpage* is valid
  - 0x02 - database contains replicated tables
  - 0x04 - database contains replicated stored procedures
  - 0x20 - RepAgent enabled
  - 0x40 - autostart RepAgent
- *gen_id* The database generation ID
- *dbid* The ID number of the database
- *dbname* The name of the database
- *ltl_version* RepAgent: the log transfer language (LTL) version.

dbcc settrunc *(* ***'ltm'*** *, {* ***'valid'*** *|* ***'ignore'*** *})*
dbcc settrunc *(* ***'ltm'*** *,* ***'gen_id'*** *, db_generation)*
Command to modify the RepAgent information for the current ASE database. Note: you cannot execute this command when RepAgent is running.

- ***'valid'*** - instructs ASE to respect the secundary truncation point.
- ***'ignore'*** - instructs ASE to ignore the secundary truncation point.
- ***'gen_id'*** - instructs ASE to reset the database generation number in the log (for primary databases).

- d*b_generation* - is the new database generation number. Increment the number after restoring primary DB/log dumps to prevent RepServer from rejecting new transactions as duplicates.

set replication *[**'on'** | **'force_ddl'** | **'default'** | **'off'**]*
Command that enables or disables replication of DDL and/or DML commands to the standby database for the current session.

- *'**on**'* - enables replication of DML commands for tables marked with sp_setreptable, if **sp_reptostandy** is set to *'**none**'*. If **sp_reptostandby** is set to *'**L1**'* or *'**all**'*, enables replication of DML and DDL commands to the standby database. This is the default setting.
- ***'force_ddl'*** - always enables replication of DDL commands for the current session. If **sp_reptostandby** is set to *'**L1**'* or *'**all**'* DML commands are replicated for all user tables. If **sp_reptostandby** is set to *'**none**'* DML commands are replicated for tables marked with **sp_setreptable**.
- ***'default'*** - turns off '***force_ddl*** ' and returns to the default ('***on***').
- *'**off**'* - turns off replication of marked tables and user stored procedures for the current session. No DML or DDL commands are replicated to the standby or replicate database.

sp_configure ***'enable rep agent threads'*** *[, **1** | **0**]*
Enables (**1**) or disables (**0**) RepAgent threads in ASE (option is dynamic).

sp_config_rep_agent *[database_name*
*[, {'**enable**', 'repserver_name', 'repserver_username', 'repserver_password'}* |
*'**disable** ' [, '**preserve secondary truncpt**']* |
***rs_servername,** 'repserver_name'* |
***rs_username,** 'repserver_username'* |
***rs_password,** 'repserver_password'* |
***scan_batch_size,** 'no_of_qualifying_log_records'* |
***scan_timeout,** 'scan_timeout_in_seconds'* |
***retry_timeout,** 'retry_timeout_in_seconds'* |
***fade_timeout,** 'fade_timeout_in_seconds'* |
***skip_ltl_errors,** { **true** | **false** }* |
***batch_ltl,** { **true** | **false** }* |
***send_warm_standby_xacts,** { **true** | **false** }* |
***connect_dataserver,** 'connect_dataserver_name'* |
***connect_database,** 'connect_database_name'* |
***send_maint_xacts_to_replicate,** { **true** | **false** }* |
***security_mechanism,** 'mechanism_name'* |
***unified_login,** { **true** | **false** }* |
***mutual_authentication,** { **true** | **false** }* |
***msg_confidentiality,** { **true** | **false** }* |
***msg_integrity,** { **true** | **false** }* |
***msg_replay_detection,** { **true** | **false** }* |
***msg_origin_check,** { **true** | **false** }* |
***msg_out_of_sequence_check,** { **true** | **false** }* |
***(12.0) skip_unsupported_features,** { **true** | **false** }* |
***(12.0) ha_failover,** { **true** | **false** }]]*
Modifies or displays the configuration parameters for the RepAgent thread for an ASE database. Note: use **'disable'** only when downgrading the RepServer or changing the primary database status, as this command truncates all RepAgent entries in the **sysattributes** table. Without parameters, displays all databases where RepAgent has been configured.

sp_help_rep_agent *[dbname* | ***null** [, **'recovery'** | **'config'** | **'process'** | **'scan'** | **'security'** | **'all'**]]*
Displays static and dynamic information about RepAgent threads. To show which RepAgents are running, use: **sp_help_rep_agent** ***null, recovery***. To show RepAgent status: **sp_help_rep_agent** ***null, process***

sp_start_rep_agent *dbname [, **'for_recovery'** [, **'connect_dataserver'**, **'connect_database'** [, **'repserver_name'**, **'repserver_username'**, **'repserver_passwd'**]]]*
Starts a RepAgent thread for the specified ASE database.

sp_stop_rep_agent *dbname [, **'nowait'**]*
Shuts down the RepAgent thread for the specified database.

sp_reptostandby *dbname [, **'L1'** | **'all'** | **'none'**]*

Marks or unmarks a database for schema replication to the standby database. Enables replication of supported schema changes and data changes to user tables. Note that this may take a long time when the database contains large amounts of **text**, **image** or **rawobject** data.

- ***'L1'*** - sets the schema replication support level to the level of the current RepServer/ASE. If you upgrade the database, the feature set will remain at this level. For RepServer 11.5 and 12.0, *'**L1**'* is equivalent to *'**all**'*.
- ***'all'*** - sets the schema replication support level to that of the current ASE Server. If you upgrade the database, the feature set of the later release is enabled automatically.
- ***'none'*** - unmarks all database tables for replication and turns off data and schema replication to the standby database (this may take some time; it also takes exclusive locks on all user tables in the database).

The following Transact-SQL DDL commands and ASE System Stored Procedures are replicated to the standby database when *'**L1**'* is specified:

alter table	create default	create index
create procedure	create rule	create table
create trigger	create view	drop default
drop index	drop procedure	drop rule
drop table	drop trigger	drop view
grant	revoke	sp_addalias
sp_addgroup	sp_addmessage	sp_adduser
sp_addtype	sp_bindefault	sp_bindmsg
sp_bindrule	sp_changegroup	sp_chgattribute
sp_commonkey	sp_config_rep_agent	sp_dropalias
sp_dropgroup	sp_dropkey	sp_dropmessage
sp_droptype	sp_dropuser	sp_foreignkey
sp_primarykey	sp_procxmode	sp_recompile
sp_rename	sp_setrepcol	sp_setreplicate
sp_setrepproc	sp_setreptable	sp_unbindefault
sp_unbindmsg	sp_unbindrule	

sp_setrepdefmode *table_name [, { **'owner_on'** | **'owner_off'** }]*
Modifies or displays the owner status of tables marked for replication. This is for ASE version 11.5 or later databases.

- ***owner_on*** - changes the owner status of the table so the table name and owner name are considered when the table is marked for replication. Enables replication of multiple tables of the same name with different owners
- ***owner_off*** - changes the owner status of the table so that only the table name is considered when the table is marked for replication.

sp_setrepcol *table_name [, { column_name* | ***null** }] [, { **do_not_replicate** | **always_replicate** | **replicate_if_changed** }]*
Sets or displays the replication status for **text**, **image** or **rawobject** columns.

sp_setrepproc *[procedure_name [, { **'function'** | **'table'** | **'false'** }]]*
Enables or disables replication for a stored procedure or displays the current replication status of a stored procedure. Note: the ***'table'*** option is old (RepServer version 10) functionality which is still supported but has been replaced by the *'function'* option.

sp_setreptable *[table_name [, { **'true'** | **'false'** } [, { **'owner_on'** | **'owner_off'** }]]]*
Enables or disables replication for an ASE table or displays the current replication status of a table. **sp_setreptable** sets the replication status of all columns in a table, including **text**, **image** or **rawobject** columns, to ***always_replicate***. See **sp_setrepdefmode** for the description of the ***owner_on*** and ***owner_off*** options.

sp_setreplicate *[object_name [, { **'true'** | **'false'** }]]*
Note: This RepServer version 10 procedure is still supported, but has been superseded by **sp_setreptable** and **sp_setrepproc**. It enables or disables replication for a table or stored procedure. **sp_setreplicate** sets the replication status of **text**, **image** or **rawobject** columns to **do_not_replicate**.

ASE RSSD Stored Procedures

The following stored procedures exist only in the RSSD (an ASE database):

rs_capacity *TranDuration , FailDuration , SaveInterval , MatRows*
Helps you estimate stable queue size requirements.

rs_configure *[config_variable [, value]]*
Displays or changes RepServer configuration variables. You can also change most RepServer configuration parameters using the **configure replication server** command.

rs_delexception *[transaction_id]*
Deletes a transaction from the RSSD exceptions log.

rs_fillcaptable *RepDefName , InChRateI , InChRateD , InChRateU , OutChRateI , OutChRateD , OutChRateU , InTranRate , OutTranRate , DelFlag*
Records estimated transaction rates for an existing RepDef.

rs_helpclass *[class_name]*
Displays error classes and function string classes and their primary RepServer.

rs_helpclassfstring *class_name [, function_name]*
Displays the function-string information for function strings. Non-customized, inherited function strings are not displayed for derived function-string classes.

rs_helpdb *[dataserver, database]*
Provides information about databases that RepServer knows about.

rs_helperror *server_error_number [,**'v'**]*
Displays the RepServer error actions mapped to a given dataserver error number. Specify***'v'*** to display a detailed listing.

rs_helpexception *[transaction_id [, **'v'**]]*
Displays transactions in the exceptions log. Specifying ***'v'*** displays the text of the transaction in a detailed listing.

rs_helpfstring *rep_def [, function_name]*
Displays the parameters and function string text associated with a RepDef.

rs_helpfunc *[rep_def [, function_name]]*
Displays information about functions available for a RepServer or a RepDef.

rs_helppartition *[partition_name]*
Displays information about RepServer partitions.

rs_helppub *[pub_name [, primary_dataserver [, primary_database [, article_name]]]]*
Displays information about publications or articles. The parameters act as a filter and can be specified independently. To get information about a specific article, specify:
rs_helppub null, null, null, *article_name*.

rs_helppubsub *[sub_name [, pub_name [, primary_dataserver [, primary_database [, replicate_dataserver [, replicate_database]]]]]]*
Displays information about publication subscriptions and article subscriptions. The parameters act as a filter and can be specified independently.
Use **rs_helppub** to determine all subscriptions for an article or a publication. Use **check subscription** to get the most accurate report of subscription status.

rs_helprep *[rep_def]*
Displays information about RepDefs.

rs_helprepdb *[dataserver, database]*
Displays information about databases with subscriptions for RepDefs in the current RepServer.

rs_helproute *[replication_server]*
Provides status information about routes.

rs_helpsub *[sub_name [, rep_def [, dataserver, database]]]*
Displays information about subscriptions.

rs_helpuser *[user]*
Displays information about user login names known to a RepServer.

rs_helpreptable *database, [table_owner ,] table*
Displays information about RepDefs created against a primary table.

rs_init_erroractions *new_error_class , template_class*
Initializes a newly created error class with error actions from an existing error class.

rs_zeroltm *dataserver, database*
Resets the secundary truncation point value for a database to zero (0). Use this procedure after you have used the ASE command **dbcc settrunc('ltm', 'ignore')** to disable the secundary truncation point and truncate the logs, but before you restart the RepAgent.

Replication Server System Database (RSSD) Tables

The RSSD is an ASE database. It contains the following tables:

rs_articles
Stores information about articles known to this RepServer.

rs_classes
Stores the names of function string classes and error classes.

rs_columns
Contains information about the columns of RepDefs

rs_config
Holds configuration parameter values that can be modified using SQL in the RSSD.

rs_databases
Contains database names known at a RepServer site; 'dbid' values start at 101.

(RS 12.0)
rs_datatype
Contains information about all user-defined datatypes in replication definitions.

rs_diskpartitions
Stores information about the disk partitions used for stable message queues.

rs_erroractions
Maps a data server error number to an action to be taken by a RepServer.

rs_exceptscmd
Stores information used to retrieve the text of transactions from the exceptions log.

rs_exceptshdr
Stores information about failed transactions. The source and output commands of the transactions are stored in the system tables**rs_exceptscmd** and **rs_systext**. All rows for a transaction in **rs_exceptscmd** and **rs_exceptshdr** are identified by the column sys_trans_id.

rs_exceptslast
Stores the origin ID, secondary queue ID, and associated information about the last logged transaction written into the exceptions log. When status=2, transactions are rejected due to message loss. When =1, in loss detetction mode; when =0, valid.

rs_funcstrings
Stores the function strings associated with each function.

rs_functions
Stores information about RepServer functions.

rs_idnames
Stores the names of RepServers and databases known to the ID server. This table is relevant only at the ID Server site.

rs_ids
Stores the last ID used for various types of objects.

rs_lastcommit

RepServer uses this table to find the last transaction committed from each data source. The **rs_lastcommit** table is stored in each user database, not in the RSSD.

rs_locater
Stores the last locater field received by stable queues from each of their senders, (i.e. last recorded scan point in the pdb log). For normal RepAgents, type="E".

rs_maintusers
Stores the user login names and passwords which RepServer uses to access other RepServers and data servers.

rs_msgs
Stores the localized error messages used during installation and by some RepServer stored procedures.

rs_objects
Stores RepDefs. For table RepDefs, objtype="R"; for function RepDefs, objtype="F".

rs_oqid
Stores the last queue ID received from an origin site. Values for the "valid" column are the same as for **rs_exceptslast.status** .

rs_publications
Stores information about publications known to this RepServer.

rs_queuemsg
When asked to dump its queues into the RSSD using **sysadmin dump_queue**, RepServer dumps queue entries into **rs_queuemsg**. If this table already has rows for a segment, they are deleted from the table before dumping the latest rows from that segment.

rs_queuemsgtxt
Stores the command or text portion of messages in stable queues. Each stable queue entry is represented by one or more rows in this table. Multiple rows are needed when the length of data in the stable queue entry exceeds the maximum command field length of 255 bytes.

rs_queues
Stores information used by the RepServer stable queue manager and guaranteed delivery system to allow site recovery.

rs_recovery
Logs actions to be performed by RepServer upon recovery, in case of a failure.

rs_repdbs
Contains information about all of the databases known by a primary RepServer. This information is stored when a subscription is entered for a database at a replicate site.

rs_repobjs
Stores autocorrection flags for RepDefs at replicate RepServers.

rs_routes
Stores routing information about network traffic. If the column **dest_rsid** contains the same value as **through_rsid**, that row's status indicates the status of the local RepServer.

rs_routeversions
Stores version information about the RepServers on each end of a route.

rs_rules
Stores subscription rules. There is one row for each term in a subscription WHERE clause, as well as for article WHERE-clauses.

rs_segments
RepServer uses raw disk space to store message data in the queues. Each 1Mb segment is allocated by only one queue, and contains 64*16Kb blocks. This table holds information about the allocation of each segment.

rs_sites

Stores the names of RepServers known at a site. For the IDserver, id=16777317; for other RepServers, higher than this.

rs_subscriptions
Stores information about subscriptions, triggers, and fragments.

rs_systext
Stores the text of repeating groups for various other tables such as **rs_funcstrings** .

rs_threads
RepServer uses the information in this table to detect deadlocks and to perform transaction serialization between parallel DSI threads. An entry is updated in this table each time a transaction is started and more than 1 DSI thread is defined for a connection. The **rs_threads** table is stored in each user database, not in the RSSD.

(RS 12.0)
rs_translation
Contains information about datatype translation between replication servers.

rs_users
Stores a row for each user with access to the RepServer. In order to log into RepServer, the login name must be in **rs_users**.

rs_version
Stores version number information for the replication system. At local RepServers, only the local version number and the system-wide version number are stored. At the ID Server, version information is stored for all RepServers in the replication system.

rs_whereclauses
Stores information about WHERE-clauses used in articles known to this RepServer. The actual WHERE-clause information is stored in **rs_rules**.

Replication Server System Functions

Replication Server System Functions are involved in the mapping of primary transactions to replicated transactions. Custom function strings may be defined for these functions.

rs_begin
Begins a transaction in a data server.

rs_check_repl
Checks to see if a table is marked for replication.

rs_commit
Commits a transaction in a data server.

rs_datarow_for_writetext
Provides an image of the data row associated with a **text** or **image** column updated with the Transact-SQL **writetext** command.

rs_delete
Deletes a row in a replicated table.

rs_dumpdb
Initiates a coordinated database dump.

rs_dumptran
Initiates a coordinated transaction log dump.

rs_get_charset
Returns the character set used by a data server. This function enables RepServer to print a warning message if the character set is not what is expected.

rs_get_lastcommit
Returns rows from the rs_lastcommit system table.

rs_get_sortorder
Returns the sort order used by a data server. This function enables RepServer to print a warning message if the sort order is not what is expected.

rs_get_textptr
Retrieves the description for a **text** or **image** column.

rs_get_thread_seq *@rs_id*
Returns the current sequence number for the specified entry in the **rs_threads** system table. *@rs_id* is a number of int datatype. It is the ID of the entry to be checked. It matches the value of the id column in the **rs_threads** system table.

rs_get_thread_seq_noholdlock *@rs_id*
Returns the current sequence number for the specified entry in the **rs_threads** system table, using the noholdlock option.

rs_initialize_threads *@rs_id*
Sets the sequence of each entry in the rs_threads system table to 0

rs_insert
Inserts a single row into a table in a replicate database.

rs_marker *@rs_api*
Passes its parameter to RepServer as an independent command. Warning: if used to simulate atomic materialization as described in the *Replication Server Administration Guide,* be careful not to make any syntax errors in the "@rs_api" argument.

rs_repl_off
Specifies whether transactions executed by the maintenance user in the ASE database are replicated.

rs_rollback
Rolls back a transaction. This function is reserved for future use.

rs_select
Selects rows for subscription materialization from the primary copy of a replicated table and, for subscription dematerialization, from the replicate copy of the table.

rs_select_with_lock
Selects rows for subscription materialization from the primary copy of a replicated table, using a holdlock to maintain serial consistency.

rs_set_isolation_level3
Turns on transaction isolation level 3 locking in ASE.

rs_setproxy
Modifies the login name in a data server, using the ASE **set proxy/set session authorization** command.

rs_textptr_init
Allocates a text pointer for a **text** or **image** column.

rs_triggers_reset
Turns off triggers in the ASE database.

rs_truncate
Truncates a table in a replicate database.

rs_update
Updates a single row in a table in a replicate database.

rs_update_threads *@rs_id , @rs_seq*
Updates the sequence number for the specified entry in the **rs_threads** system table. *@rs_id* is the ID of the entry to be updated. *@rs_seq* is the new sequence number for the entry.

rs_usedb
Modifies the database context in a data server.

rs_writetext
Modifies **text** or **image** data for a column.

Troubleshooting Tips

When troubleshooting replications problems, check the following:

- Always check the RepServer errorlog for any error messages. The errorlog location can be found using **admin log_name**.
- Run **admin who** to determine if any replication threads are down; for a warm standby, also run **admin logical_status**.
- When having problems with RepAgents, check the ASE errorlog for error messages.
- Check the RSSD transaction log is not full; ensure the RSSD accessible using the RSSD server logins. The RSSD location is shown by **admin rssd_name**.
- Check the stable devices are not full (**admin disk_space** or **rs_helppartition**).
- When finding error messages, look up the error numbers in the files in the **`./doc`** subdirectory in the RepServer installation directory.
- When nothing else works, try restarting RepServers, RepAgents or ASE servers -- sometimes this helps!
- For more troubleshooting tips, see the *Replication Server Troubleshooting Guide* or check out `http://www.sypron.nl/rs_tips.html`

RepServer resources on the web

Here's an arbitrary selection of some RepServer-related WWW links:

The Sybase FAQ: `http://www.isug.com/Sybase_FAQ/REP`
Various RepServer tips: `http://perso.wanadoo.fr/dbadevil/`
On-line Sybase manuals (including downloadable manuals in PDF format): `http://www.sybase.com/support/manuals/`
RepServer newsgroups exist at Sybase's own news server: `forums.sybase.com`
RepServer Tips & Tricks: `http://www.sypron.nl/rs_tips.html`
This QuickRef Guide can be found at :`http://www.sypron.nl/rs_qref.html`

How to assemble the pages into a booklet:

1. Print the file (paper size should be A4). You may need to adjust the page margins or the space between the columns if it doesn't print correctly; there should be about 7 millimeters (~ 5/12 inch) free space at the left and right margins (if you can't get it printed correctly, you can always change the lay-out to 1 column/page, and so some manual cut-and-pasting).

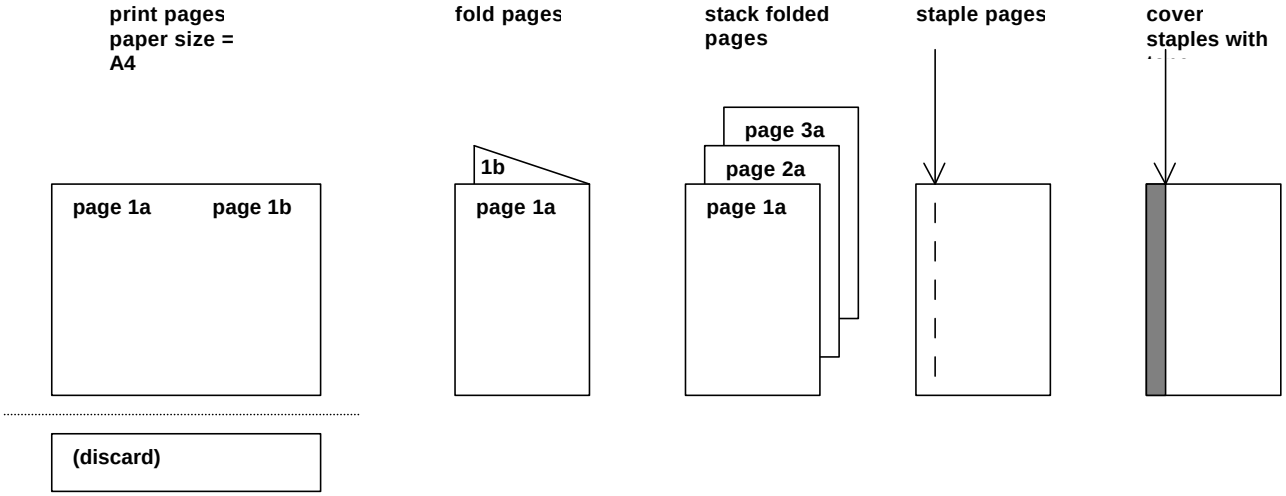
2. Cut off the blank bottom part of the pages

3. Fold the printed pages, so that there is one column on each folded side.

4. Stack the folded pages

5. Staple the folded pages together at the left-hand margin

6. If you don't like the sight of those staples, cover them with a bit of sticky tape.



Do you find this Quick Reference Guide useful ?

Then check out ***"The Complete Sybase ASE Quick Reference Guide"*** , the only truly complete ASE Quick Reference Guide that exists. This 96-page, fit-in-your-pocket book contains all commands, functions, procedures and other information you need most often when working with ASE 11.9 or 12.0. It includes full command syntax, command descriptions and examples, as well as a selection of useful undocumented commands. There's also a handy 10-page index. It's probably the most useful book any ASE DBA or developer could want to have.

More information, including a preview of some of the actual pages of the book, is available at www.sypron.nl/qr . Online orders are accepted (using a secure payment interface).