Trusted RUBIX™ Version 6

Administrative Commands Reference Guide

Revision 7

RELATIONAL DATABASE MANAGEMENT SYSTEM

Infosystems Technology, Inc.

4 Professional Dr - Suite 118

Gaithersburg, MD 20879

TEL +1-202-412-0152

Infosystems Technology, Inc.
4 Professional Dr - Suite 118
Gaithersburg, MD 20879

INTRODUCTION **1**

ENVIRONMENT VARIABLES 1
NOTATION 2

RXISQL(RX) **2**

↓ FUNCTION 2
↓ SYNOPSIS 2
↓ DESCRIPTION 2
↓ BUILT-IN COMMANDS 3

RXAUDITRPT(RX) **4**

↓ FUNCTION 4
↓ SYNOPSIS 4
↓ DESCRIPTION 4
↓ SECURITY 6

RXAUDITSET(RX) **10**

↓ FUNCTION 10
↓ SYNOPSIS 10
↓ DESCRIPTION 10
↓ SECURITY 12

RXDB(RX) **12**

↓ FUNCTION 12
↓ SYNOPSIS 12
↓ DESCRIPTION 12
↓ SECURITY 14

RXDUMP(RX) **14**

↓ FUNCTION 14
↓ SYNOPSIS 14
↓ DESCRIPTION 14
↓ EXAMPLES 16
↓ SECURITY 16

RXEXPORT(RX) **17**

↓ FUNCTION 17
↓ SYNOPSIS 17
↓ DESCRIPTION 17
↓ SECURITY 17

RXIMPORT(RX) **18**

↓ FUNCTION 18
↓ SYNOPSIS 18
↓ DESCRIPTION 18
↓ LABELED LOADING 19
↓ SECURITY 20

RXLOGS(RX) **21**

↓ FUNCTION 21
↓ SYNOPSIS 21

↓ DESCRIPTION........21
↓ EXAMPLES........22
↓ SECURITY........22

RXRESTORE(RX) 23

↓ FUNCTION........23
↓ SYNOPSIS........23
↓ DESCRIPTION........23
↓ EXAMPLES........24
↓ SECURITY........24

RXSVRMAN(RX) 25

↓ FUNCTION........25
↓ SYNOPSIS........25
↓ DESCRIPTION........25
↓ SECURITY........26

RXPOLMAN(RX) 26

↓ FUNCTION........26
↓ SYNOPSIS........26
↓ DESCRIPTION........27
↓ SECURITY........28

Introduction

Environment Variables

This chapter contains the reference pages to the TRUSTED RUBIX (TR) executable commands. These commands can be invoked from a UNIX prompt. They reside in the **bin** subdirectories of the TR home directory. Soft links are generally created (depending on the host operating system) in the */usr/bin* directory to the utilities. Therefore, these commands can invoked without being preceded by the full path name. If necessary, the TR **bin** directories may be added to the user's executable path.

UNIX provides a general facility called environment variables to assist user's in operating the system. Environment variables can be set in various manners. They can be set temporarily or permanently. A temporary modification will remain in effect only during the current login session. A permanent modification will remain in effect each time the user logs in. To make a modification permanent, the necessary commands have to be placed in the **.profile** or the **.login** files.

The best way to understand how to set environment variables is with an illustration. In the following, the environment variable RXPWID will be set to the value 80.

For the **sh** and the **ksh** use the "command, RXPWID = "80"; export RXPWID. The UNIX command set can be used to display the current value of all environment variables.

For the **csh** use the command, **setenv** RXPWID = "80". The UNIX command **printenv** can be used to display the current value of all environment variables.

The following table is a description of the environment variables used by the TR **rxisql** client. The name, default value and use of the environment variables are listed. Further configuration of the **rxisql** client may be made at runtime. Server side configurations are set using the **rxconfig** file.

Name	Default Value	Description
RXDSPRPORT	NULL	The port used to communicate with the TR Dispatcher.
SQLDATABASE	default_db	The database name to connect to.
EDITOR	vi	Editor invoked by **rxisql** to allow a user to edit SQL commands.
RXSQLHIST	50	Number of commands saved in **rxisql** history.
RXSQLPS1	rxsql : ! >	This variable determines the first prompt in **rxisql.** The **!** character is replaced with a number **n,** which represents that this is the **nth** command to **rxisql.**
RXSQLPS2	! : # >	The command continuation prompt in **rxisql.** The **#** character is replaced with a number depicting the number of continuation lines for the command.

Notation

Each command in this chapter has a FUNCTION, SYNOPSIS, DESCRIPTION, and SECURITY subheading.

The FUNCTION subheading provides a brief description of the command described. The SYNOPSIS subheading summarizes the syntax of the command invocation. The DESCRIPTION subheading is a description of the command, its uses and an explanation of different invocation parameters. The SECURITY subheading presents any restrictions on the use of the command.

The following syntactic notation is used to define the TRUSTED RUBIX command syntax in the SYNOPSIS sections:

→ *Lowercase* words and characters are **literals,** and should be entered **exactly** as they appear. *Commands* are **always literals** and appear as **boldface.**
→ Square brackets **[]** enclose syntax that is optional.
→ Vertical bars (|) separate mutually exclusive options.
→ An Ellipsis (….) means that all syntax enclosed by the immediately preceding pair of brackets may be repeated.

rxisql(RX)

↓ FUNCTION

Interactive execution of SQL statements.

↓ SYNOPSIS

rxisql [OPTIONS] [file]

↓ DESCRIPTION

-a
Ask user before rolling back an uncommitted transaction on exit.

-b
Show SQL command parser debug information.

-c
Do not show number of items changed for insert/delete/update.

-d DATABASE
Set database name where DATABASE is *dbname[@host[:port]]*.

-f
Flush output on each print.

-g
Show SQL command parse tree.

-i
Ignore errors and continue parsing batched files.

-n
Check for syntax only, do not execute.

-o
Show successful outcome of non insert/delete/update statements.

-p
Suppress asking for a password and user/group name.

-r
Rollback pending transaction on exit (transaction commit is the default).

-s
Show comments from SQL script file.

-t LEN
Trim printing of column values to LEN characters.

-v
Display entire statement prior to parsing.

If you would like to run SQL commands from a file, **`rxisql`** may be invoked with a file name as an argument. In this mode, the program will terminate at the first error. If it is invoked with the **`-i`** option, the program will be executed until the end-of-file irrespective of errors. If it is invoked with the **`-v`** flag, the command is printed out on standard output.

If there is any text starting with '**`-`**'**,** it is considered as a comment. It can start anywhere in the line and the rest of the line is ignored. As with any programming language, RUBIX/SQL users are encouraged to liberally comment their code as an aid to future readers. Note also that white space may be used liberally to separate keywords and names; white space is only meaningful within quotation marks, as part of a literal string.

If the **rxisql** program is invoked successfully, a prompt appears on the screen. The prompt can be set by the user as desired, using the environmental variables RXSQLPS1 and RXSQLPS2. RXSQLPS1 refers to the primary prompt which appears at the beginning of the statement. Each command can have multiple lines and therefore a secondary prompt referring to RXSQLPS2 appears at the second line of the statement. If these environmental variables are not set, the default prompt is used (see page 1 of this Guide).

Each SQL statement is delimited by '**`;`**'. One command can have multiple statements. If any line in a command terminates with a delimiter, then it is considered as the end of the command.

The default database name is *default_database*. If the database name is not set using another method, this value is used.

↓ BUILT-IN COMMANDS

rxisql has a number of built-in commands that make the task of creating queries and viewing results easier.

h[istory] [**n**]
A short form '**hn**' can be used which lists the last **n** commands in the command history. If there are fewer commands in the history, only those are displayed. All commands you see in the history are after substitution of edited and rerun commands. The default number of commands in the history is 50 and it can be set to a different number using the environment variable RXSQLHIST (see page 1 of this Guide).

r[erun] [**n**]
Re-executes the **nth** command in the command history. If **n** is not specified, the last command is executed.

t[rim] [**n**]
Limit all column widths to **n** characters and truncate values that do not fit. If a column is truncated, a '**>**' character appears at the right of the column.

If no **n** is specified, column trimming is disabled. This command is useful in preventing columns with large domains (e.g. the INFORMATION_SCHEMA tables) from wrapping. This makes them much easier to view.

q[uit] [**n**]
Exit **isql.**

c[onnections]
Display connection information.

!
Execute subshell.

rxauditrpt(RX)

↓ FUNCTION

Display recorded audit information.

↓ SYNOPSIS

```
rxauditrpt  [-a s|f] [- e [!] event [, ...]
            [-u username [ , ... ] ] [-U userid [ , ... ] ]
            [-l sensitivity label] [-L sensitivity label]
            [-r minimum-maximum]
            [-R minimum-maximum]
            [-s time] [-S time]
```

↓ DESCRIPTION

The **rxauditrpt** command allows a user with the appropriate privileges to display the information placed by TRUSTED RUBIX into the system audit logs.

The following options are defined below:

`-a s | f`
Select records which have an outcome of s (successful) or f (failed).

`-e [!] event [, ...]`
Select all records where the listed events match the specified event list. If the event list is preceded by an !, select all events except those specified in the list.

`-u user [, …]`
Select all records whose user field(s) match on the specified user name(s).

`-U [, ...]`
Select all records whose user field(s) match on the specified UID(s).

`-l sensitivity label`
Select all records where the sensitivity label of the objects(s) match the specified sensitivity label. Multiple sensitivity labels may be specified with one invocation of the **`rxauditrpt`** command by repeated use of the **`-l`** option, up to a system defined maximum.

`-L sensitivity label`
Select all records where the sensitivity label of the user matches the specified sensitivity label. Multiple sensitivity labels may be specified with one invocation of the **`rxauditrpt`** command by repeated use of the **-l** option, up to a system defined maximum.

`-r minimum-maximum`
Select all records where the sensitivity label of the object(s) fall within the range of sensitivity labels specified by minimum and maximum (inclusive). The minimum sensitivity label and maximum sensitivity label must be separated by a **`-`**.

`-R minimum-maximum`
Select all records where the sensitivity label of the user(s) fall within the range of sensitivity labels specified by minimum and maximum (inclusive). The minimum sensitivity label and maximum sensitivity label must be separated by a **`-`**.

`-s start time`
Select all records which were generated on or after the specified start time.

`-S stop time`
Select all records which were generated on or before the specified stop time.

`-z database`
Select all records which were generated for the specified database.

Event Types
The event list provided with the -e option must be separated by commas. The TR audit events are listed in the **Trusted RUBIX Trusted Facility Manual**.

Sensitivity Label Names

The sensitivity label names provided with the `-l`, `-L`, `-r`, and `-R` options must be either the fully qualified sensitivity label name or the sensitivity label alias of a valid sensitivity label. For the sensitivity labels provided with the `-r` and `-R` options, the maximum sensitivity label must dominate the minimum.

Time Identifiers

Time identifiers have the format: YYYY-MM-DD-HH-MM-SS. The time identifier may be truncated to achieve a desired granularity. For example, to retrieve all audit records from July 2008 to August 10, 2008 (inclusive), the command **`rxauditrpt -s 2008-7 -S 2008-8-10`** is issued.

User Identifiers

The user list provided with the `-u` or `-U` option must be separated by commas, and may be either the UID (`-U`) or name (`-u`) of the user for whom the audit records are to be selected.

Display Format

The records will be displayed in the following format:

1. time
2. event id
3. process id
4. status
5. username
6. groupname
7. session id
8. subject lvl
9. event name
10. object name
11. event status
12. object lvl
13. subject lvl
14. subject id
15. transaction ts
16. database name
17. event specific information.

↓ SECURITY

MAC The user must be at the system high session sensitivity label to execute the command.

DAC This command can only be run on the server by those with the *rubix.audit.report.dbname* authorization.

AUDIT The command will be audited using the **`sql_audit_review`** event ID. The audit record will contain the following information:

1. event name
2. user ID
3. group ID
4. database name
5. session sensitivity label
6. operation status, timestamp
7. transaction ID
8. process ID
9. session ID
10. database label
11. command line arguments.

rxauditset(RX)

↓ FUNCTION

Specify or display criteria for setting audit information.

↓ SYNOPSIS

```
rxauditset  [eventlist]
            [-a] eventlist
            [-s] eventlist
            [-m] [-u userlist] [-g grplist]
            [-l sublvllist] [-r subrnglist]
            [-L objlvllist] [-R objrnglist]
            [-d dblist]
            [-i ON/OFF]
            eventlist
            [-h]
```

↓ DESCRIPTION

The **rxauditset** command allows a user with the appropriate privileges to specify or display the criteria which determine whether an audible event is eligible to be recorded by TRUSTED RUBIX in the audit trail. Any updates made to the audit event list will not affect the existing server processes. All new invocations of the database server will use the updated audit event list.

The first format, with no options, allows for displaying the current audit criteria. Specifying the command with no arguments or with just an event list will display the current audit criteria for all events or the specified event list.

The second format **[-a]** allows a list of events to be added to the current audit criteria. The event will be unitized to audit every occurrence of the event.

The third format **[-s]** is used to subtract a list of events from the current audit criteria. When an event is subtracted it will never audit the event.

The last format **[-m]** is used to modify a currently added list of events. The event’s criteria may be specified so as to add, subtract, or to set the various criteria associated with each event.

The following options are defined below:

-d dblist
Specify a list of database names for an event.

-g grplist
Specify the group(s) whose audit information is to be defined. The list of groups will be specified with the group name or ID and separated by commas.

-h
Specifies the list of valid events and provides information on correct syntax.

-l [**+** | **-**] **sublvllist**
Specify a subject label list for an event. If the **lvllist** is preceded by a **+,** the specified subject sensitivity label list is added to the current list of subject sensitivity labels. If the **lvllist** is preceded by a **-,** the specified subject sensitivity label list is removed from the current list of subject sensitivity labels. If the **lvlvlist** is preceded by nothing, the current list of subject sensitivity labels are replaced with the specified subject sensitivity labels. Multiple subject sensitivity labels may be specified with a single **-l** option by separating the sensitivity labels with a semi-colon.

-L [**+** | **-**] **objlvllist**
Specify an object label list for an event. If the **lvllist** is preceded by a **+**, the specified object sensitivity label list is added to the current list of object sensitivity labels. If the **lvllist** is preceded by a **-**, the specified object sensitivity label list is removed from the current list of object sensitivity labels. If the **lvlvlist** is preceded by nothing, the current list of object sensitivity labels are replaced with the specified object sensitivity labels. Multiple object sensitivity labels maybe specified with a single **-l** option by separating the sensitivity labels with a semi-colon.

-r [**+** | **-**] **subrnglist**
Specify a range of subject sensitivity labels for an event. The sensitivity labels specified for the range are separated with a **-**. If the **rnglist** is preceded by a **+**, the specified subject range list is added to the current list of ranges. If the **rnglist** is preceded by a **-**, the specified subject range list is removed from the current list of ranges. If the **rnglist** is preceded by nothing, the current list of subject ranges are replaced with the specified subject ranges. Multiple label ranges may be specified with a single **-r** option by separating the subject ranges with a semi-colon.

-R [**+** | **-**] **objrnglist**
Specify a range of object sensitivity labels for an event. The sensitivity labels specified for the range are separated with a **-**. If the **rnglist** is preceded by a **+**, the specified object range list is added to the current list of ranges. If the **rnglist** is preceded by a **-**, the specified object range list is removed from the current list of ranges. If the **rnglist** is preceded by nothing, the current list of object ranges are replaced with the specified object ranges. Multiple label ranges may be specified with a single **-r** option by separating the object ranges with a semi-colon.

-i on | **off**
Initialize the auditing system on or off. All other audit parameters remain unchanged.

-u userlist
Specify the user(s) whose audit information is to be defined. The list of users will be specified with the user name or ID and separated by commas.

eventlist
List of events provided that are separated by commas.

-h

Online help.

↓ SECURITY

MAC None.

DAC This command can only be run on the server by those with the *rubix.audit.setcrit.dbname* authorization.

AUDIT The command will be audited using the **sql_audit_modify** event ID. The audit record will contain the following information:

1. event name
2. user ID
3. group ID
4. database name
5. session sensitivity label
6. operation status
7. timestamp
8. transaction ID
9. process ID
10. session ID
11. database label
12. command line arguments.

rxdb(RX)

↓ FUNCTION

Drop, list, or rename a database. Change the storage directory for volumes or LOBs. Start or stop the maintenance server.

↓ SYNOPSIS

```
rxdb  -d [-f] [-q] DBNAME
      -l [-a] [DBNAME]
      -r NEWNAME [-f] OLDNAME
      -x NEWPATH DBNAME
      -b NEWPATH DBNAME
      -s DBNAME
      -t DBNAME
```

↓ DESCRIPTION

The **rxdb** command may be used to: drop, list, or rename a database; change the storage directory for volumes or LOBs; or start or stop the maintenance server.

The **-d** option is used to drop a database. The **rxdb** command will drop a database even when the database is too corrupted to open. The options for dropping a database are:

-f
force database to be removed

-q
quiet mode

DBNAME
name of database to drop

The **-l** option is used to list databases. The options for listing database are:

-a
display all available database(s) information

[**DBNAME**]
name of database to list; if not specified all databases are listed

The **-r** option is used to rename a database. The options for moving a database are:

-f
force the database to be moved

NEWNAME
the new database name

OLDNAME
the old database name

The **-x** option is used to move the storage directory for the database volumes. Database volumes store non-LOB table data. The current storage directory may be listed using the ***rxdb*** command with the **–la** option. The options for moving the volume directory are:

NEWPATH
name of database to operate upon

DBNAME
name of database to operate upon

The **-b** option is used to move the storage directory for the database large objects (LOBs). The current storage directory may be listed using the ***rxdb*** command with the **–la** option. The options for moving the LOB directory are:

NEWPATH
name of database to operate upon

DBNAME
name of database to operate upon

The **-s** option is used to start the maintenance server for a specified database. The options for starting the maintenance server are:

DBNAME
name of database to operate upon

The **-t** option is used to terminate the maintenance server for a specified database. The options for terminating the maintenance server are:

DBNAME
name of database to operate upon

↓ SECURITY

MAC
→ To list a database the database sensitivity label must be dominated by the user's session sensitivity label.
→ To rename a database or move a storage directory the database sensitivity label must equal the user's session sensitivity label.

DAC
→ To list a database the user must have the *rubix.admin.db.ls.dbname* authorization.
→ To rename a database or move a storage directory the user must have the *rubix.admin.db.mv.dbname* authorization.
→ To drop a database the user must have the *rubix.admin.db.rm.dbname* authorization.

AUDIT The command will be audited using the **sql_db** event ID. The audit record will contain the following information:

1. event name
2. user ID
3. group ID
4. database name
5. session sensitivity label
6. operation status
7. timestamp
8. transaction ID
9. process ID
10. session ID
11. database label
12. command line arguments

rxdump(RX)

↓ FUNCTION

Perform a full backup of one database.

↓ SYNOPSIS

rxdump [**-c** **-m** SIZE **-n** DUMP_NAME **-p** DUMP_PATH **-qv**] DB_NAME

↓ DESCRIPTION

The DB_NAME argument specifies the database to be backed-up. The options which can appear on the command line are:

-c
Compress the dump files (recommended).

-m SIZE
Restrict each dump file to a maximum of SIZE bytes. The default is 2,147,483,647 bytes.

-n DUMP_NAME
Name of the dump. The dump name is used to create a storage directory for all dump files and to refer to the dump when using the *rxrestore* command. If no dump name is specified one is automatically generated using a format of RXDUMP-DBNAME-YR-MON-DAY-SEQNUM. Dump files contained within the storage directory are named using a format of RXDUMP-SEQNUM.dmp.

-p DUMP_PATH
The full path to the location to store the dump. It must exist and be read/writable to the user issuing the *rxdump* command. The storage directory will be created at this location and will be named the same as the dump name. The default dump path is *RXINSTALL_PATH/backups*, where RXINSTALL_PATH is the installation directory for Trusted RUBIX and is specified in the Trusted RUBIX Installation Guide for the specific install platform.

-q
Quiet mode. Produces output only on error.

-v
Specifies verbose diagnostics for the **rxdump** process.

rxdump performs a backup of exactly one database. The database to be backed-up is specified by the DB_NAME argument.

The backup will be placed into the DUMP_PATH directory, which may be the default value (*RXINSTALL_PATH/backups*) or specified as an option (**-p**). The dump path must exist and be read/writable by the user of the **rxdump** command.

Each backup has an associated DUMP_NAME, which may be an automatically generated value or specified as an option (**-n**). The automatically generated dump name has a format of RXDUMP-DBNAME-YR-MON-DAY-SEQNUM , where DBNAME is the name of the database being backed-up, YR-MON-DAY is the numerical year, month, and day of the backup, and SEQNUM is a sequence number used to distinguish multiple backups performed on the same day. The dump name is used to create a storage directory (in DUMP_PATH) to contain the dump files and to refer to the dump when using the **rxrestore** command. The dump name storage directory must not exist prior to the **rxdump** invocation.

The backup is stored in one or more dump files named using a format of RXDUMP-SEQNUM.dmp, where SEQNUM is a sequence number starting with one. All backup files for a single backup are stored within the DUMP_PATH/DUMP_NAME directory. Because multiple dump files may be used during the backup, there is no database size limitation for a backup to be performed. Data within the dump files may optionally be compressed using the **–c** option. Use of compression may significantly reduce the size of a backup and is highly recommended. Data within the dump files is verified during **rxrestore** using a 32 bit cyclic redundancy check (CRC) code.

Internally, the **rxdump** command creates a database transaction and then reads through all objects in the database, writing them to the dump files. It therefore creates a snapshot of the database as-of the created transaction. The transaction used to perform the backup is identified by the *latest consistent moment* (*lcm*), which is output (as a date-time) during the backup process. If the dump is used to restore the database, using the **`rxrestore`** command, the new database will be equal to the snapshot defined by the *lcm*. To apply updates that occurred after the *lcm* to the new database, restore redo logs must be applied during the restore process. Please see the **`rxrestore`** command for more information.

This command should be run at SYS_HIGH MAC label to ensure that all records are read. A warning message will be printed if the command is run at a label lower than SYS_HIGH. If a lower label is used, then only dominated objects will be included in the backup. Note that this may create errors during a restore that applies redo logs as updates to higher level objects will fail.

If a dump is archived, the storage directory (named by the dump name), all contained dump files, and any relevant restore redo log files should be included in the archive.

↓ EXAMPLES

The following will back up the *MyDb* database, using compression, to the default dump path using an automatically generated dump name:

`rxdump` **–c** *MyDb*

The following will back up the *MyDb* database, using compression, to the */MyBackups* dump directory (must exist and be read/writable) using a dump name of *MyDump*. The dump files will be located in the */MyBackups/MyDump* directory, which will be created:

`rxdump` **–c –p** */MyBackups* **–n** *MyDump MyDb*

↓ SECURITY

MAC Only catalogs, schemas, views, tables, indexes, and rows dominated by the session sensitivity label are dumped to the output device. The session sensitivity label must dominate the database being dumped or the command terminates with an error message. To ensure all objects of a database are dumped the command should be run by a user whose operating system session sensitivity label is SYS_HIGH. A warning message is produced if the command is invoked at a session sensitivity label other than SYS_HIGH.

The operating system files and directories are created with user and group ownership equal to user invoking the command. The operating system permission modes are set to READ, WRITE, EXEC (directory only) for the user. The file is labeled with the session sensitivity label of the user invoking the command.

DAC The user must have the *rubix.restore.backup.dbname* authorization.

AUDIT The command will be audited using the **`sql_dump`** event ID. The audit record will contain the following information:

1. event name
2. user ID, group ID
3. database name
4. session sensitivity label
5. operation status
6. timestamp
7. transaction ID
8. process ID
9. session ID
10. database label
11. command line arguments

rxexport(RX)

↓ FUNCTION

Exports the data in a table.

↓ SYNOPSIS

`rxexport` [**`-d`** DATABASE **`-c`** CATALOG **`-l`** DELIMITER **`-q`** QUOTE **`-s`** SCHEMA **`-f`** FILE **`-mp`**] TABLE

↓ DESCRIPTION

`rxexport` lists current rows in the table to a file or standard output. The data listed by **`rxexport`** is immediately usable by **`rximport`.**

`-d`
if this option is used, then the database name supplied in DATABASE is used.

`-c`
if this option is used, then catalog name in CATALOG is used; Otherwise the catalog "default_catalog" is used.

`-l`
if this option is used, then the delimiter symbol in DELIMITER is used; otherwise, a space is used as the delimiter.

`-q`
if this option is used and the value of QUOTE is 'Y' all data fields will be double-quoted, otherwise with the value of QUOTE as 'N', data fields will not be double-quoted. The default is 'N'.

`-s`
if this option is used, then the schema name supplied in SCHEMA is used; otherwise the schema "**`default_schema`**" is used.

`-f`
if this option is used, then the file name specified in FILE is used to hold the result of the export; otherwise, the file <**`table`**>**`.d`** is used.

`-m`
if this option is used, then data is listed with the row-label as the first column.

`-p`
if this option is used, then all versions of polyinstantiated rows are listed.

↓ SECURITY

MAC All MAC rules apply as if the user were selecting the table using an SQL statement. The operating system session sensitivity label of the invoker must equal the object label of the operating system output file if it already exists or the command terminates with an error message. If the output file does not exist it is created and labeled with the operating system session sensitivity label.

DAC This command can only be run on the server by users with the *rubix.user.export.dbname* authorization. All DAC rules apply as if the user were selecting the table using an SQL statement. The operating system permissions of the invoker must allow WRITE access to the operating system output file if it already exists or the command terminates with an error message. If the output file does not exist it is created with the user/group ownership equal to the invoker and given the operating system permissions READ-WRITE for the user only.

AUDIT The command will be audited using the **`sql_export`** event ID. The audit record will contain the following information:

1. event name
2. user ID
3. group ID
4. database name
5. session sensitivity label
6. operation status
7. timestamp
8. transaction ID
9. process ID
10. session ID
11. relation label
12. catalog name
13. schema name
14. relation name
15. host operating system export file name
16. export data

rximport(RX)

↓ FUNCTION

Load a table with data from a text file.

↓ SYNOPSIS

`rximport` [**`-d`** DATABASE **`-c`** CATALOG **`-l`** DELIMITER **`-s`** SCHEMA **`-f`** FILE **`–C`** #ROWS **`-mP`**] TABLE

↓ DESCRIPTION

`rximport` reads data from a text file, inserting rows into a TABLE. TABLE must be created before **`rximport`** is invoked.

`-d`
if this option is used, then the database name supplied in DATABASE is used.

`-c`
if this option is used, then catalog name in CATALOG is used; Otherwise the catalog "default_catalog" is used.

`-C`
if this option is used, then a COMMIT is automatically issued for ever #ROWS number of rows that

are imported. This option may be used when system resources for active transactions become exhausted during an import of many rows. If #ROWS is 0 (the number zero), then only a single COMMIT is issued after all rows have been imported. The default behavior is for the *rximport* command to calculate the COMMIT rate based upon available system resources. The *rximport* command will always attempt to import all rows under a single COMMIT. Only when resources reach a threshold will more COMMIT operations be used.

-l
if this option is used, then the delimiter symbol in DELIMITER is used; otherwise, a space is used as the delimiter.

-P
if this option is used, then performance information is printed to *stdout* after all rows have been imported.

-s
if this option is used, then the schema name supplied in SCHEMA is used; otherwise the schema "default_ schema" is used.

-f
if this option is used, then the file name specified in FILE is used; otherwise, the file **<table>.d** is used.

-m
if this option is used, then data is loaded with the row label as the first column.

You can create a new relation using the SQL command CREATE TABLE, and then load the data.

Free-Form Input

rximport reads the data file in free form mode, that is:

→ Each record of the data file produces one row of TABLE.
→ Within each record, successive fields are used as the input for successive columns. If there are fewer fields on a record than there are columns, the unfilled columns are marked as Unknown.

Successive fields must be separated by blanks or tabs. Blanks may be included in a field by enclosing the entire field in single or double quotation marks. The following is a valid free-form input file for **projects:**

```
TMK   'Tomahawk Navigation'     EE   500000.00   2006-11-01   2007-11-01
PCS   'Patriot Control System'  EE   60000.00    2006-01-15   2006-06-20
```

↓ LABELED LOADING

If **rximport** is invoked with the -m flag, then a labeled load is performed. In this case, each record in the input **(.d)** file is assumed to be prefixed with a sensitivity label.

The **-m** option of **rximport** may only be exercised by security administrators who are not constrained on the range of labels that they may load. The only constraint is that the session label of the security administrator must dominate the label of the file used as input.

Note that the output of a command such as:

```
rxexport [-m] cinfo
```

is immediately acceptable as input to **rximport** -m.

↓ SECURITY

MAC

→ If the '**-m**' option is specified each row is labeled with the corresponding label contained in the input file.

→ If the '**-m**' option is not used each row is labeled with the operating system session sensitivity label at the time of invocation.

→ The operating system session sensitivity label of the invoker must dominate the object label of the operating system input files or the command terminates with an error message.

DAC

→ This command can only be executed on the server. To successfully use this command, the invoker must have the *rubix.user.import.dbname* authorization.

→ They must also have the INSERT privilege on TABLE or have the *rubix.dac.insert.dbname* authorization.

→ The operating system permissions of the invoker must allow READ access to the operating system input files or the command terminates with an error message.

→ The **-m** option can only be used by those with the *rubix.mac.recl.** authorization.

→ The *rubix.mac.recl.upgrade.dbname* is used to raise the sensitivity label of the row above that of the user's session sensitivity label.

→ The *rubix.mac.recl.downgrade.dbname* is used to lower the sensitivity label of a row below the user's session sensitivity label.

AUDIT The command will be audited using the **sql_import** event ID. The audit record will contain the following information:)

1. event name
2. user ID
3. group ID
4. database name
5. session sensitivity label
6. operation status
7. timestamp
8. transaction ID
9. process ID
10. session ID
11. relation label
12. catalog name
13. schema name
14. relation name
15. host operating system import file name
16. import data

rxlogs(RX)

↓ FUNCTION

`rxlogs` — gives authorized users the ability to list and manipulate audit, restore, and recovery log files for specific databases.

↓ SYNOPSIS

`rxlogs` -[**`a`**|**`r`**|**`y`**|**`s`**] [**`l`**|**`d`** FILENAME|**`ALL`**] [**`m`** NEWPATH] DATABASE

↓ DESCRIPTION

The **`rxlogs`** command gives authorized users the ability to list, delete, and move log files for a specific database. The log files are created as operations are performed on a database. Over time these files may need to be removed to free up disk space. The options define the operation to be performed on the audit log and can be one of the following:

`-a`
: Perform operations on the audit logs. The **–a**, **–r**, **–s**, and **–y** options are mutually exclusive.

`-d` FILENAME | **`ALL`**
: Available only wit the **`-a`** or **`-r`** options. Delete a log file (if an explicit file name is given) or all log files (if **ALL** is specified) . If FILENAME does not specify a valid log file corresponding to the type (**`-a`** or **`-r`**), the file is not deleted and an error message is printed. The keyword **ALL** may be used to delete all log files. This includes the currently active log file. A new, truncated current log file will be created. The **–l**, **–d,** and **-m** options are mutually exclusive.

`-m` NEWPATH
: Move the current log directory to the one specified by NEWPATH. A new directory named DBNAME will be created in NEWPATH which will contain the log files. The directory NEWPATH must be writable for the *rubix:rubixtp* user/group. Additionally, the target file system must have sufficient free space to hold the log files.

`-s`
: Perform operations on the restore shadow logs. The **–a**, **–r**, **–s**, and **–y** options are mutually exclusive.

`-l`
: List logging directory and information about log files. The **–l**, **–d,** and **-m** options are mutually exclusive.

`-r`
: Perform the operation on the restore log files. The **–a**, **–r**, **–s**, and **–y** options are mutually exclusive.

`-y`
Perform the operation on the recovery log files. The **–a**, **–r**, **–s**, and **–y** options are mutually exclusive.

↓ EXAMPLES

The following lists all audit log files in the *MyDb* database:

`rxlogs –al` MyDb

The following deletes the restore log file *MyLog* in the *MyDb* database:

`rxlogs –rd` *MyLog MyDb*

The following moves the recovery log files to the *MyDir* in the *MyDb* database:

`rxlogs –ym` *MyDir MyDb*

↓ SECURITY

MAC TO delete log files or move a logging directory the operating system session sensitivity label of the invoker must equal the object label of the database. To list logs the operating system session sensitivity label of the invoker must dominate the object label of the database

DAC
→ To list audit log files the user must have the *rubix.audit.listlogs.dbname* authorization.
→ To delete audit log files the user must have the *rubix.audit.rmlogs.dbname* authorization.
→ To move the audit log directory the user must have the *rubix.audit.setlogs.dbname* authorization.
→ To list restore log files the user must have the *rubix.restore.listlogs.dbname* authorization.
→ To delete restore log files the user must have the *rubix.restore.rmlogs.dbname* authorization.
→ To move the restore, shadow, or recovery log directory the user must have the *rubix.restore.setlogs.dbname* authorization.

AUDIT The command will be audited using the **`sql_logs`** event ID. The audit record will contain the following information:

1. event name
2. user ID, group ID
3. database name
4. session sensitivity label
5. operation status
6. timestamp
7. transaction ID
8. process ID
9. session ID
10. database label
11. command line arguments

rxrestore(RX)

↓ FUNCTION

Restore a database created with **rxdump**.

↓ SYNOPSIS

rxrestore -d DB_NAME **-n** DUMP_NAME [**-p** DUMP_PATH **-R** REDO_PATH **-qrv**]

rxrestore -d DB_NAME **-n** DUMP_NAME [**-p** DUMP_PATH **-R** REDO_PATH **-rVv**]

rxrestore -l [**-p** DUMP_PATH **-n** DUMP_NAME **-v**]

↓ DESCRIPTION

rxrestore restores exactly one database from the specified dump to a new database. The new database must not already exist. As indicated on the SYNOPSIS section, **rxrestore** operates in three basic modes: normal restoration, verify only, and list dump information only.

The options are as follows:

-d DB_NAME
The name of the new database that is created as a result of the restoration. The database must not exist prior to the restoration.

-l
List information about dumps previously created with **rxdump**. No restoration takes place. If a DUMP_PATH is not specified, then the default is used (*RXINSTALL_PATH/backups*). If DUMP_NAME is given, then only the dump specified by DUMP_NAME is listed; otherwise, all dumps in the dump path are listed.

-n DUMP_NAME
Name of the dump to operate upon. It must already exist and the DUMP_PATH/DUMP_NAME directory must be readable to the user issuing the **rxrestore** command. The DUMP_PATH/DUMP_NAME directory contains the dump files named using a format of RXDUMP-SEQNUM.dmp.

-p DUMP_PATH
The full path to the location of the dump storage directories. It must exist and be readable to the user issuing the **rxrestore** command. The default dump path is *RXINSTALL_PATH/backups*, where RXINSTALL_PATH is the installation directory for Trusted RUBIX and is specified in the Trusted RUBIX Installation Guide for the specific install platform.

-q
Quiet mode. Produces output only on error.

-R REDO_PATH
The full path to the location of the restore redo log to apply to the new database as part of the restoration process. The path must be readable. Mutually exclusive with the **–r** option.

-r
Apply the online restore redo logs from the database that was backed-up using **rxdump**. Mutually exclusive with the **–R** option.

-V
Perform a verification of the dump and redo log (if specified). No restoration takes place.

-v
Specifies verbose diagnostics for the **rxrestore** process.

↓ EXAMPLES

The following will restore the database backed-up by the dump named *RXDUMP-2010-07-04-1* that is located in the default dump path to a new database named *MyNewDb*. Additionally, all redo logs from the backed-up database will be applied:

rxrestore –r –d *MyNewDb* **–n** *RXDUMP-2010-07-04-1*

The following will restore the database backed-up by the dump named *MyDump* that is located in the */MyBackups* dump path to a new database named *MyNewDb*. Note that the */MyBackups/MyDump* directory should contain the dump files and must exist and be readable:

rxrestore –d *MyNewDb* **–n** *MyDump* **–p** */MyBackups*

The following will list information about all dumps located in the default dump path:

rxrestore –l

The following will list information about the dump named *MyDump* that is located in the */MyBackups* dump path. Note that the */MyBackups/MyDump* directory should contain the dump files and must exist and be readable:

rxrestore –l –n *MyDump* **–p** */MyBackups*

The following will verify the dump named *MyDump* that is located in the default dump path and the restore redo logs located in the */MyRedoLog*:

rxrestore –V –R */MyRedoLog* **–n** *MyDump*

↓ SECURITY

MAC All newly created **TRUSTED RUBIX** objects are labeled with the sensitivity label the corresponding object had at the time of the dump or logging action. The operating system session sensitivity label of the invoker must dominate the object label of the operating system files and directories.

DAC This command can only be run on the server machine by those with the *rubix.restore.create.dbname* authorization. All newly created TRUSTED RUBIX objects are given the same TR permissions the corresponding object had at the time of the dump or logging action. The operating system permissions of the invoker must allow READ access to the operating system files and directories.

AUDIT The command will be audited using the **`sql_restore`** event ID. The audit record will contain the following information:

1. event name
2. user ID
3. group ID
4. database name
5. session sensitivity label
6. operation status
7. timestamp
8. transaction ID
9. process ID
10. session ID
11. database label
12. command line argument

rxsvrman(RX)

↓ FUNCTION

Startup and shutdown the TRUSTED RUBIX Dispatcher and terminate instantiated *rxserver* processes.

↓ SYNOPSIS

rxsvrman -s
rxsvrman -t [**-f**]
rxsvrman -k [**-f**] [**-d** DBNAME | **-p** PID]

↓ DESCRIPTION

The rxsvrman command will startup and shutdown the TR Dispatcher and terminate instantiated *rxserver* processes.

The **-s** option is used to startup the TR Dispatcher (*rxdspr* process). This must be performed prior to any database connection requests. There are no command line options for starting the dispatcher.

The **-t** option is used to terminate the TR Dispatcher (*rxdspr* process). All *rxserver* processes that are idle are killed. By default, if there are any active database connections the operation will fail without terminating the dispatcher or any active *rxserver* process. The option for terminating the dispatcher is:

-f
force all *rxserver* processes to be killed even if they have an active database connection.

The **-k** option is used to kill instantiated *rxserver* processes. By default, the *rxserver* process must not have an active database connection or the command will fail without killing the *rxserver* process. By default, all idle *rxserver* processes are killed. The options for moving a database are:

-f
force specified *rxserver* processes to be killed even if they have an active database connection.

-d DBNAME
kill only *rxserver* processes operating on the database DBNAME.

-p PID
kill only the single *rxserver* process identified by the process ID PID.

Note that the TRUSTED RUBIX Dispatcher may also be started, stopped, and restarted by the *root* user using the UNIX *service* command as follows:

```
service rubix start
service rubix stop
service rubix restart
```

↓ SECURITY

MAC None.

DAC
→ To start the dispatcher the user must have the *rubix.admin.dspr.start* authorization.
→ To terminate the dispatcher the user must have the *rubix.admin.dspr.term* authorization.
→ To explicitly kill an *rxserver* process the user must have the *rubix.admin.svr.term* authorization.

AUDIT None.

rxpolman(RX)

↓ FUNCTION

Administer Trusted RUBIX Security Markup Language (RXSML) policy files and their application to RDBMS objects.

↓ SYNOPSIS

The following shows the synopsis for the *rxpolman* command:

```
rxpolman
  -l [-o dbname]
  -c filepath
  -u filepath
  -d policy
  -e policy
  -a policy -o db[.cat][.sch][.tab]
```

↓ DESCRIPTION

The *rxpolman* administrative command is used to manipulate RXSML policy files. It may be used to:

- → Add an RXSML policy file to the Trusted Policy Repository
- → Delete an RXSML policy file from the Trusted Policy Repository
- → Update an RXSML policy file in the Trusted Policy Repository
- → Extract an RXSML policy file from the Trusted Policy Repository
- → List the RXSML policy files in the Trusted Policy Repository
- → Apply an RXSML policy file to a database object
- → Remove the application of an RXSML policy file from a database object
- → List assignments of RXSML policy files to database objects

Using the **-l** option without the **–o dbname** option causes all RXSML policy files in the Trusted Policy Repository to be listed. The RXSML policy files are named the same as their top-level element's *PolicySetId* or *PolicyId* attribute. The *rubix.policy.list* authorization is required for this action.

Using the **-l** option with the **–o dbname** option lists all RXSML Policy file database object assignments for the **dbname** database. The *rubix.policy.list* authorization is required for this action.

Using the **–c filepath** option will create a new RXSML policy file in the Trusted Policy Repository. The policy file to be added must be specified by **filepath**. The syntax for the policy file will be checked. Error messages will be produced and the operation will fail if any errors are found. The RXSML policy file will be named the same as the top-level element's *PolicySetId* or *PolicyId* attribute. All *PolicySetId* and *PolicyId* attribute values (both top-level and non top-level) must be unique after the new policy file has been added. The *rubix.policy.create* authorization is required for this action.

Using the **–u filepath** option will update an existing RXSML policy file in the Trusted Policy Repository. The new version of the policy file must be specified by **filepath**. The syntax for the policy file will be checked. Error message will be produced and the operation will fail if any errors are found. The RXSML policy file that will be updated will be named the same as the top-level element's *PolicySetId* or *PolicyId* attribute of the file specified by **filepath**. All *PolicySetId* and *PolicyId* attribute values (both top-level and non top-level) must be unique after the policy file has been updated. The *rubix.policy.update* authorization is required for this action.

Using the **–d policy** option will delete an existing RXSML policy file from the Trusted Policy Repository. The policy's name, the value of the top-level element's *PolicySetId* or *PolicyId* attribute, must be specified by **policy**. The policy file must not be applied to any database object or the operation will fail. The *rubix.policy.delete* authorization is required for this action.

Using the **–e policy** option will extract an existing RXSML policy file from the Trusted Policy Repository into the user's current working directory. The policy's name, the value of the top-level element's *PolicySetId* or *PolicyId* attribute, must be specified by **policy**. The policy file will be created in the user's current directory. The *rubix.policy.extract* authorization is required for this action.

Using the **-a policy -o db[.cat][.sch][.tab]** option will apply an RXSML policy file that exists in the Trusted Policy Repository to a database object. The policy's name, the value of the top-level element's *PolicySetId* or *PolicyId* attribute, must be specified by **policy**. If **policy** is given as NULL then any policy applied to the database object will be unapplied. The database object must be specified by **db[.cat][.sch][.tab]**. If the database object already has policy applied to it, then the new apply will replace it. The database object need not exist. This allow for policy to be defined before an object is added to the database. The *rubix.policy.apply* authorization is required for this action.

↓ SECURITY

MAC To create, update, and delete policies the session label must equal the label of the database. To apply policy to a database object the session label must equal the object label of the database object. To extract policy the session label must dominate the label of the database.

DAC
→ To list policies the user must have the *rubix.policy.list* authorization.
→ To create policies the user must have the *rubix.policy.create* authorization.
→ To update policies the user must have the *rubix.policy.update* authorization.
→ To delete policies the user must have the *rubix.policy.delete* authorization.
→ To extract policies the user must have the *rubix.policy.extract* authorization.
→ To apply policies the user must have the *rubix.policy.apply* authorization.

AUDIT The command will be audited using the **`sql_polman`** event ID. The audit record will contain the following information:

1. event name
2. user ID
3. group ID
4. database name
5. session sensitivity label
6. operation status
7. timestamp
8. transaction ID
9. process ID
10. session ID
11. database label
12. command line argument