
# Installation and Administration Guide

rasdaman version 9.2

rasdaman Version 9.2 Installation and Administration Guide.

# Preface

## *Overview*

This guide provides information about how to use the rasdaman multidimensional database system, in particular: installation and system administration.

rasdaman interoperates with conventional database systems, be they relational or object-oriented, in an integrated manner. There is a clear, natural distribution of work between the two different database systems according to the data types: multidimensional data are managed by rasdaman, whereas the alphanumeric data remain in the conventional system[1]. At the bottom line, however, all data - multidimensional or

[1] In some disciplines, multidimensional data are referred to as *raw data* or *processed data*, depending on their status, whereas the accompanying alphanumeric data are called *meta data*.

alphanumeric - end up in the same physical database, thereby considerably easing database maintenance wrt. consistency, backup, etc. To this end, rasdaman makes use of the storage management facilities of the database system it is coupled to.

For the purpose of this documentation, we will call the conventional database system to which rasdaman is interfaced the *base DBMS*, understanding that this base DBMS is in charge of all alphanumeric data maintained as relational tables or object-oriented semantic nets.

rasdaman operates on the PostgreSQL relational DBMS as its so-called "base DBMS" storing the actual data. This manual covers only base DBMS independent issues. You need to additionally consult the *External Products Integration Guide*, as well as the other rasdaman guides for features of the rasdaman system which are common to all platforms.

## *Audience*

The information in this manual is intended primarily for database and system administrators.

## *Rasdaman Documentation Set*

This manual should be read in conjunction with the complete rasdaman documentation set which this guide is part of. The documentation set in its completeness covers all important information needed to work with the rasdaman system, such as programming and query access to databases, guidance to utilities such as the graphical-interactive query tool *rView*, and release notes.

In particular, current restrictions, known bugs, and workarounds are listed in the Release Notes. All documents, therefore, always have to be considered in conjunction with the Release Notes.

The rasdaman Documentation Set consists of the following documents:

- Installation and Administration Guide
- Query Language Guide
- C++ Developer's Guide
- Java Developer's Guide
- raswct Developer's Guide
- rView Guide

# Table of Contents

# 1 Getting Started

This section outlines the hardware and software requirements and describes the rasdaman distribution directory. Make sure you are familiar with this before proceeding to the next section which describes the installation procedure.

## *1.1 Hardware Requirements*

It is recommended to have at least 3 GHz processor frequency and 8 GB main memory.

Disk space depends on the size of the databases, as well as the requirements of the base DBMS of rasdaman chosen. The footprint of the rasdaman installation itself is around 400 MB.

## 1.2 Software Requirements

You need to download and install the following packages which are required by rasdaman:

- **tools:** git-core, make, autoconf, automake, libtool, gawk, flex, bison, ant, g++, libstdc++, doxygen, openjdk-6-jdk, Tomcat (or another suitable servlet container)
- **java:** Java Runtime Environment (JRE) 1.6 or higher
- **general libraries:** libreadline-dev, libssl-dev, libncurses5-dev
- **database:** PostgreSQL 8.x, libecpg-dev
- **image formats:** libtiff-dev, libjpeg-dev, libhdf4g-dev, libpng12-dev, libnetpbm10-dev

Note: this list may change and unintendedly become outdated. It is recommended to additionally consult the rasdaman website, www.rasdaman.org.

During installation, presence of these packages will be checked. See Section 2.1 for more information on the configuration steps.

**Optional packages**

For accessing documentation, an Acrobat Reader is needed (available, e.g., from www.adobe.com/products/acrobat/readstep2.html), plus some Web browser capable of HTML 4.0. However, this is not a “hard” requirement, it is not checked by the installation procedure.

Some packages, such as HDF4, are optional. This means that the feature (in the case of HDF4: support for the HDF4 data format) is not available unless its use is specified during configuration; see

```
./configure --help
```

for options available.

See Section 2.1 for more information on the configure script.

**Warning**

Do **not** use versions PG 8.3.0 through 8.3.6, it **won't work**.

## 1.3 Server Types

Historically, rasdaman internally uses three different client/server communication protocols: `RPC`, `HTTP`, and `RNP`. The `RPC` and `HTTP` types are deprecated, and `RNP` is default and recommended (exception: rview, which still requires `RPC`).

- SUN RPC. This mechanism is deprecated. Only rview still needs an RPC type server running.

- HTTP. This has been used by rasdaman applications programmed in Java, such as rasogc. It is deprecated.
- RNP. This protocol substitutes both the other protocols. It is th default for all tools except rview.

To accommodate the different clients, an appropriate server has to be started as the counterpart. For example, a Java application needs an RNP or HTTP type server and is not able to communicate with an RPC type server.

The communication type of a rasdaman server is set via rascontrol (see Section 5.6).

The communication protocol of a rasdaman client can be set through shell environment variable `RMANPROTOCOL` as follows:

```
RMANPROTOCOL=RNP       -- use RNP (default)

RMANPROTOCOL=HTTP      -- use HTTP (deprecated)

RMANPROTOCOL=RPC       -- use RPC (deprecated)
```

## *1.4 Support*

Installation information, FAQs, and troubleshooting information is available on www.rasdaman.org.

For support in installing rasdaman and any other question you may contact rasdaman GmbH at www.rasdaman.com.

# 2 Getting It Up: System Installation and Database Creation

This section describes how to install rasdaman, software package dependencies, and how to initialise rasdaman and the base DBMS so that the rasdaman system finally is up and running.

Please read this in conjunction with the rasdaman *External Product Integration Guide* for your particular base DBMS brand and the relevant manuals of the base DBMS vendor.

This section outlines the procedure for installing rasdaman from scratch. For incremental installation, such as from patch files and updates or upgrades follow the particular instructions there (e.g., in the release notes).

## *2.1 Download and Install rasdaman*

### Create Dedicated User

While rasdaman can be installed and run under any operating system user, for security reasons it is strongly recommended to create a

dedicated user to shield rasdaman activity (e.g., log files) from the rest of the system.

This user can be named `rasdaman`, but any other (pre-existing or newly established) user will do as well; in this case, adjust the commands listed in the sequel where necessary. In the sequel it will be assumed that a user account named `rasdaman` has been created and that you are logged in as user `rasdaman`, e.g., by using this command:

```
$ su - rasdaman
```

Note that the dollar sign ("$") symbolizes the command line prompt and is not to be typed in.

### Download

You can get a complete *rasdaman Community* file set from www.rasdaman.org by executing the following command:

```
$ git clone git://kahlua.eecs.jacobs-university.de/rasdaman.git
```

This will create a sub-directory `rasdaman` in your current working directory.

### Installation

**Step 1:** Change into this directory:

```
$ cd rasdaman
```

**Step 2:** The following command will prepare compilation for your system.

```
$ ./configure
```

Any missing components will be reported; if this is the case, then install the missing packages and retry configuration.

Configuration can be customized, see the `--help` option. In particular, to change the default installation directory, `/usr/local`, use option `--prefix`.

Example for an alternative location:

```
$ ./configure --prefix=/usr/local/rasdaman-8.0.0
```

**Step 3:** Next, compile and link rasdaman:

```
$ make
```

**Step 4:** Install rasdaman at the place specified before:

```
$ make install
```

Note that the user executing this command must have write access to the target directory specified.

**Step 5:** For your convenience you can add the executable path location to the `$PATH` definition.

### Example

Assuming variable `$RASDAMAN` is pointing to the installation directory. Then, the following setting will allow invoking rasdaman tools without any path indication:

```
$ export PATH=$RASDAMAN/bin:$PATH
```

Note that all paths *inside* rasdaman scripts and binaries are adjusted automatically during generation, so you do not need to edit any script.

In the remainder of this document the installation path is referred to as `$RASDAMAN`.

## 2.2 Installation Directories

As described in the previous section, the installation directory can be chosen at compile time. Inside this installation directory (referred to as `$RASDAMAN`), substructure is as follows.

### Important directories

**bin/**
: binaries

**etc/**
: configuration files

**include/**
: include files for compiling C++ rasdaman applications

**lib/**
: libraries for linking C++ rasdaman applications

**log/**
: server logs

**share/rasdaman/**
: examples, documentation, error message text files, etc.

## 2.3 Base DBMS Set-Up

### Create Relational Database

For PostgreSQL a database instance needs to be created. Currently two of the PostgreSQL authentication schemes are supported:

- **Trust authentication.** In this case, rasdaman must provide a user name and password acceptable to PostgreSQL. This is set in `$RASDAMAN/etc/rasmgr.conf` before server start.
- **Ident-based authentication.** PostgreSQL must run under under the same operating system user under which rasdaman will be operated

(such as `rasdaman` as recommended above). No further rasdaman setup action is required.

Following PostgreSQL recommendation, the database cluster supposed to hold the rasdaman database should be owned by the `rasdaman` operating system user, meaning that the cluster is generated under the `rasdaman` login and the PostgreSQL server is running under this login.

See the PostgreSQL documentation for details and consequences of choosing a particular scheme.

**Example**

The following commands create a PostgreSQL database sitting in directory `$DATADIR` and with owner `rasdaman`, assuming the commands are executed as effective user `rasdaman`:

```
$ initdb –D $DATADIR

$ pg_ctl –D $DATADIR start
```

**Verify database**

Roughly speaking, base DBMS configuration is ok if the SQL interpreter can successfully connect to the base DBMS when logged in to the target user account, in the above discussion: `rasdaman`. If this does not work then you may have to reconsider `$PATH` and `$LD_LIBRARY_PATH` settings.

**Delete a Database**

To delete a database *db*, use the pertaining PostgreSQL command as operating system user `rasdaman`:

```
dropdb db
```

## *2.4 Database Initialization*

**PostgreSQL Variables**

The following PostgreSQL variables have to be set for the `rasdaman` user.

```
export PGSQLDIR=/usr/local/pgsql
export PATH=$PGSQLDIR/bin:$PATH
export LD_LIBRARY_PATH=$PGSQLDIR/lib:$LD_LIBRARY_PATH
```

Note: Depending on your installation, variable `PGSQLDIR` may have to be adapted accordingly to point to the local PostgreSQL installation directory used.

**Create rasdaman database**

The `create_db.sh` script creates and initializes a rasdaman database named `RASBASE` by instantiating a set of MDD types. It has no parameters and is invoked as:

```
$ create_db.sh
```

This script assumes that a database cluster has been generated (according to Section 2.1 and the PostgreSQL documentation) and that the pertaining server process is accessible from user `rasdaman`. The script performs creation of a PostgreSQL database named `RASBASE` does and rasdaman initialization via `rasdl`.

**Notes**

For creation of new databases, rasdaman servers have to be restarted, otherwise the databases may not be recognized. Updating a rasdaman database schema (that is: creating new types and the like), however, does not need a server restart.

Running `rasdl` does not require the rasdaman server to be up.

## *2.5 Server Configuration (optional)*

Rasdaman is a multi-server multi-user system. The server processes available must be configured initially, which is done in file `$RASDAMAN/etc/rasmgr.conf`. For distribution, this configuration contains only one server process going by a name like, for example, `N1`. If this is fine then you can just leave it as it is. If you want to change this by modifying server startup parameters or increasing the number of server processes available then see Section 4.3 for details on how to do this; probably you want to change this only once the initial, single-server installation has been verified to be operational.

## *2.6 Server Start*

Make sure that the ports rasdaman uses are not blocked in your system. These are 7001 for the scheduler (rasmrg9 and 7002, 7003, etc. for each worker process. Ports used can be reconfigured, cf. Section 3.2.

Start rasdaman by invoking

```
$ start_rasdaman.sh
```

**Notes**

- Messages printed by `start_rasdaman.sh` will not always show the detailed system state. If, for example, the rasdaman servers fail to contact the base DBMS then nevertheless a message “Server started” may appear.

  Workaround: use this to get the actual server state, as user `rasdaman`:

```
rascontrol –e –x ”list srv”
```

## 2.7 Demo Database

The rasdaman distribution contains a demo database which serves as a first test of successful installation.

Inserting demo data into the fresh database is done through

```
$ rasdaman_insertdemo.sh localhost 7001 \
  $RASDAMAN/share/rasdaman/examples/images rasadmin rasadmin
```

Note that repeated invocations are not harmful – each of the sample collection will simple receive additional objects made of the same images.

After successful completion, you can check whether the three rasdaman collections containing the example images have been created through:

```
$ rasql –q "select r from RAS_COLLECTIONNAMES as r" \
        --out string
```

This command shows a list of all collections existing in the database. There should be `mr`, `mr2`, and `rgb`.

Further, you can inspect these data using, for example, rview (see the *rasdaman rView Guide*).

Congratulations! At this point, if everything completed successfully, rasdaman is up and running and prepared for data definition, data import and retrieval, and any other suitable task.

# 3 rasdaman Architecture

The parallel server architecture of rasdaman offers a scalable, distributed environment to efficiently process even very large numbers of concurrent client requests. Yet, server administration is easy to accomplish, with only few things to do to have a smoothly running, highly performant installation. Moreover, the system is implemented in a special high availability technique where most server management operations can be done with the server up and running, limiting the need for a server shutdown to the absolute minimum.

In this Section the general rasdaman server architecture is outlined. It is recommended to study this section so as to understand server administration terminology used in the next Section.

## *3.1 Executables Overview*

The following executables are provided in the `bin/` directory, among others:

- `rasmgr` is the central rasdaman request dispatcher;

- `rasserver` is the rasdaman server engine, it should not (and actually cannot) be invoked in a standalone manner;
- `rascontrol` allows to interactively control the rasdaman server by communicating with `rasmgr`;
- `raspasswd` allows to administrate rasdaman database user logins;
- `rasdl` is the command-line based schema maintenance tool; this is (currently) not a client application, but connetxts directly to the relational database manager.
- `rasql` is the command-line based query tool.

The `rasdl` and `rasql` tools are explained in detail in the *rasdaman Query Language Guide*.

## *3.2 Server Manager and Server*

### Overview and Terminology

The rasdaman server configuration consists of one dispatcher process per computer, `rasmgr` (we will refer to it as *manager* in the sequel), and server processes, `rasserver` (referred to as *servers*), of which at a given time none, one, or several ones can be running. All server processes are under control of the manager. Server manager and rasdaman server(s) all run on the same physical hardware, the *rasdaman host*.

The servers resolve requests, thereby generating calls to the relational database system which in turn accesses its database files. For the purpose of this manual, the relational server together with the database it maintains are collectively called the *database*. The machine the relational database server runs on is referred to as *database host* (Figure 2).

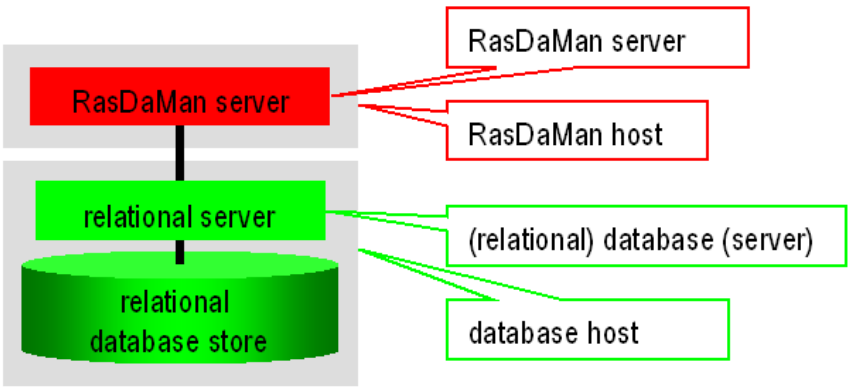


**Figure 2: Overall server hierarchy, introducing the terminology for rasdaman hardware and software environment**

### Server Structure in General

The manager accepts client requests and assigns server instances to them, taking them from the pool of server processes it maintains. In distributed installations, it keeps contact to the managers on other machines to further dispatch client requests across all the rasdaman servers available. Whenever needed, the administrator can launch further server instances, or shut them down again.

Upon system configuration definition (see Section 4.3), a unique name is assigned to each server identifying it to the manager.

Each rasdaman server is assigned to a relational database server, laid down in the manager configuration file. Databases can be registered and associated to particular rasdaman servers at any time.

rasdaman hosts and database hosts are identified by their resp. host name in common domain address form, e.g., `martini.rasdaman.com` or `199.198.197.50`.

`Rascontrol` is the interactive front-end to `rasmgr` and, as such, the main utility for user and system management. It provides the necessary functions to manage the whole system configuration, to add and remove user, to change their rights, and to obtain information about system activity.

The rasdaman server, i.e., `rasserver`, is controlled by the manager which starts and stops server instances. Hence, the `rasserver` executable should not (and actually cannot) be invoked directly.

### Dynamic Server Assignment

The process of client/server communication and server scheduling is done as follows (see numbers in Figure 3).

1. The client starts every `OPENDB` and `BEGIN TRANSACTION` request with an HTTP call to the manager, providing the required service type (RPC, HTTP, etc.) and the database name, together with user name and password.
2. The manager's answer is the server ID of a free server, or an error message in case no server is available or access is denied for the given login.
3. Client-Server communication to perform the database requests.
4. Upon `CLOSEDB` and `ABORT/COMMIT TRANSACTION` the server informs the manager that it is available again. This is also done upon a client timeout.

These negotiation steps are performed between client library and server, hence transparent to the application.

The rasdaman server system is started by invoking the server manager `rasmgr` (see Section 5.2). If it finds a configuration file, them autopmatically all servers indicated will be started; alternatively, server configuration can be done directly through `rascontrol` (see Section 5.3).

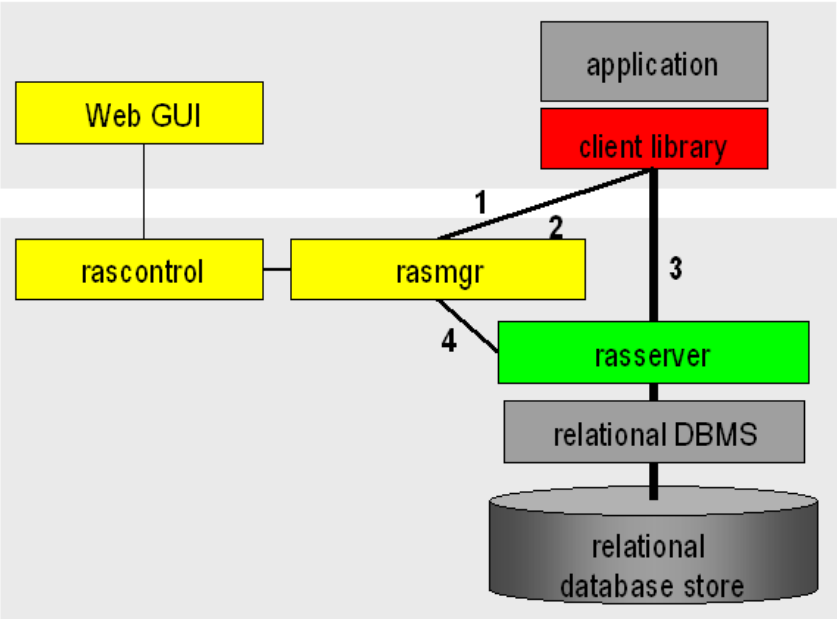


**Figure 3: Internal server management**

### System Start-up

Invocation of the `rasmgr` executable must be done under the operating system login under which the rasdaman installation has been done, usually (and recommended) `rasdaman`.

### Server Federation

rasdaman servers running on different computers can be coupled so as to form one single server network. To this end, the dispatcher processes, `rasmgr`, running on each node exploits knowledge about other nodes in the network. This is accomplished via `inpeer` and `outpeer` directives, best written into `rasmgr.conf`; see Section **Error! Reference source not found.** for details.

Whenever a local dispatcher finds that a new session cannot be served as there is no more free server process available currently it will attempt to acquire a free server from a peer `rasmgr`. Upon success, this server is transparently communicated to the client.

Any server in the network can forward requests this way (depending on the administrator controlled security policy on each node). Hence, there is **no single point of failure** in such a rasdaman peer network.

All peers in a rasdaman federation are assumed to access the same underlying database, or a database with identical contents.

### Authentication

On every machine hosting rasdaman servers a separate manager has to run. The manager maintains a configuration and an authorization file. None of them should be changed by the administrator, as they are generated, maintained, and overwritten by the manager.

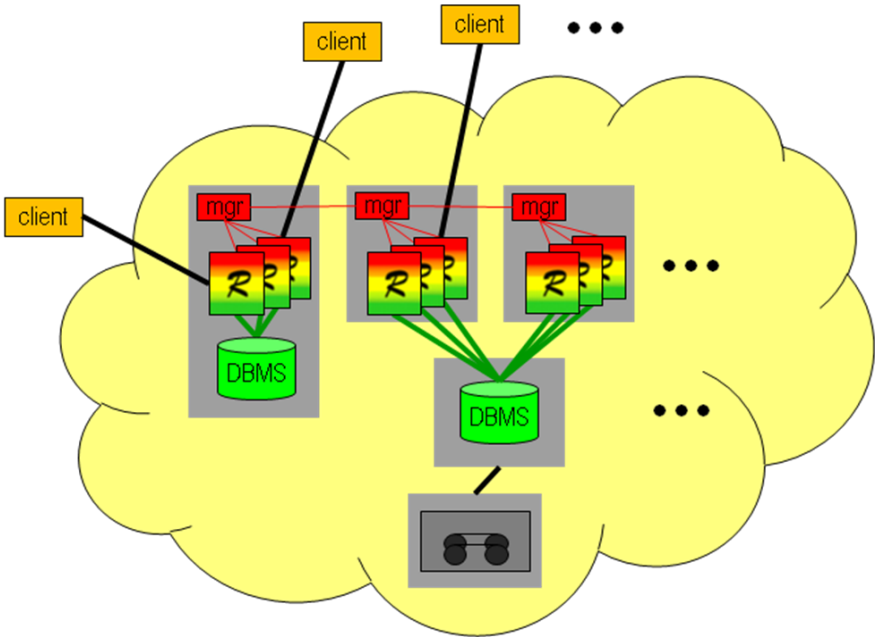


**Figure 4: rasdaman federation**

#### rasdaman Manager Defaults

The manager's default name is the hostname (the one reported by the UNIX command `hostname`), but it can be changed (see the `change` command). By default, it listens to port 7001 for incoming requests and uses port 7001 for outgoing requests:

#### Port Number Recommendations

To keep overview of the ports used, it is recommended to use the following schema (there is, however, no restriction preventing from choosing another schema – just keep an overview…):

- use port number 7001 for the server manager;
- use port numbers 7002 to 7999 for rasdaman servers.

## *3.3 Tile caching*

To speed up tile access, a caching mechanism is implemented in rasdaman.

This cache has proven advantageous for access performance in particular during bulk ingestion, in order to work around inefficiencies of BLOB handling in PostgreSQL in presence of certain tiling schemes.

### How to use

The cache can be enabled in `rasmgr.conf` by indicating a cache limit. The corresponding parameter, `--cachelimit`, has to be placed after the `-xp` parameter (cf. Section 5.6). By default, tile caching is disabled.

| | |
|---|---|
| `--cachelimit` *c* | upper limit of cache area in bytes (default: 0) |

The specified maximum amount of memory will be used for tile caching. When this limit is reached, tiles in the cache get replaced.

When enabled, `INSERT` and `UPDATE` rasql statements can be issued in the normal way. Upon `COMMIT`, the cache flushed to PostgreSQL and cleared.

### Known limitations

The tile cache is experimental. Therefore, it should be used during bulk ingestion *only*, and disabled again afterwards. Leaving the tile cache on will cause selection queries to fail randomly. Furthermore, it is best to have only one ingest server active in rasmgr.conf, and `countdown` and `timeout` parameters set to very large values.

### See also

Latest information, as well as usage hints, are provided on http://rasdaman.org/wiki/Performance.

# 4 Access Interfaces

Rasdaman services can be invoked in several ways: through command line, through Web services, and through C++ and Java APIs.

## *4.1 Command Line Tools*

Queries can be submitted to the command line tool `rasql`. Complete control over the server is provided through several utilities, in particular `rasmgr`; see Section 4.3 for details. All tools can communicate with local and remote rasdaman servers (current exception: rasdl).

## *4.2 Web Services*

Several Web services are available with rasdaman. They are implemented as servlets, hence independent from the array engine and only available if

started in a servlet container such as Tomcat or jetty. They can be accessed under the common context path `/rasdaman` as follows:

The corresponding war files by default are located in installation directory `share/rasdaman/war/`. During the configuration step and before compilation (cf. Section 2.1) this directory can be set with `configure` option `--with-wardir`. For example, this allows indicating the directory where war files are installed (such as Tomcat's `webapps/` directory). Note that the effective user invoking "make install" (as available in shell variable `$USER`) has to have write permissions in that directory.

### 4.2.1 rasql Queries

Submission of rasql queries is possible through path `/rasdaman/rasql`.

This requires deployment of war file `rasdaman.war`.

#### Invocation syntax

The request has three mandatory parameters:

| | |
|---|---|
| `username` | rasdaman login name under which the query will be executed |
| `password` | password corresponding to the login |
| `query` | rasql query string, properly encoded for URI embedding |

#### Example

```
http://www.acme.com/rasdaman/rasql
    ? username=rasguest
    & password=rasguest
    & query=select%20encode(mr,"png")%20from%20mr
```

### 4.2.2 Geo Web Services

A series of geo Web services is available at the following endpoints:

- Geo Web Services based on the interface standards of the Open Geospatial Consortium (OGC Web Services, OWS):

  | | |
  |---|---|
  | `/rasdaman/ows/wms`: | OGC Web Map Service (WMS) |
  | `/rasdaman/ows/wcs`: | OGC Web Coverage Service (WCS) suite |
  | /`rasdaman/ows/wcps` | OGC WCPS (deprecated, now with WCS) |
  | /`rasdaman/ows/wps` | OGC Web Processing Service (WPS) |

  This requires deployment of war file `petascope.war`.

- A Coordinate Reference System (CRS) Resolver service, SECORE, which is identical to the one deployed by OGC) is available under path `/def`. This path is reflecting the OGC resolver architecture where http://www.opengis.net/def/crs is the branch for CRSs served by SECORE.

  This requires deployment of war file `def.war`.

### *4.3 APIs*

Programmatic access is available through self-programmed code using the C++ and Java interfaces; see the C++ and Java Guide for details.

# 5 Server Administration

This Section explains how to start up and shut down servers, as well as how to monitor and influence server state.

It is recommended to first study the previous section so as to understand server administration terminology used here.

## *5.1 General Procedure*

**`rasmgr` VS. `rascontrol`**

It is important to distinguish between the manager, `rasmgr`, and its control front-end, `rascontrol`. The manager runs as a background process, supervising activity of local (and possibly remote) rasdaman servers. Interaction between user (i.e., administrator) and the manager takes place through the interactive control front end.

In the sequel, it is first described how to launch the manager `rasmgr`, then `rascontrol` commands are detailed.

#### Important Security Note

To remain compatible with older rasdaman versions, clients use login "rasguest" / password "rasguest" by default (i.e., when no user and password are explicitly set by the application). In the distribution configuration, this user is defined to have read-only access to the databases – in plain words,

- According to the default configuration,
- users can access,
- but not manipulate databases
- without authentication.

Therefore, the administrator is strongly urged to adapt authentication settings to the local security policy before switching databases online.

See Section 5.10 to learn more about user management mechanisms.

## *5.2 Running the Manager*

#### Manager Startup

Starting up the rasdaman system is done by invoking the rasdaman manager, `rasmgr`, from a shell under the `rasdaman` operating system login. Usually the manager will be sent to the background:

```
rasmgr &
```

Starting `rasmgr` is the only direct action to be done on it. Any further administration is performed using `rascontrol`.

Note that, unless a server configuration has been defined already, no rasdaman server is available just by starting the manager.

#### Invocation Synopsis

Manager invocation synopsis:

```
$ rasmgr [--help] [--hostname h] [--port p]
```

where

| | |
|---|---|
| `--help` | print this help |
| `--hostname h` | host on which the manager process is running is accessible under name / IP address *h* (default: output of Unix command `hostname`) |
| `--port p` | manager will listen to port number *p* (default: `7001`) |

#### Examples

To start a manager which will listen at port 7001:

```
$ rasmgr --port 7001
```

## 5.3 `rascontrol` Invocation

The manager front end, `rascontrol`, is a command-line interface used for rasdaman administration. It allows to define the whole rasdaman system configuration, including start up and shut down of server instances and user logins and rights.

To secure access to the server administration facilities, `rascontrol` performs a login process requesting login name and password similar to the Unix `rlogin` command. User name must be one of the users defined in the rasdaman authentication list (see Section 5.10).

### `rascontrol` Synopsis

```
$ rascontrol [-h|--help] [--host h] [--port n] [--prompt n]
             [--quiet]
             [--login|--interactive|--execute cmd|--testlogin]
```

where

`--host h` — name of the host where the manager runs (default: `localhost`)

`-h`
`--help` — this help

`--port n` — port number at which the manager listens to requests (default: `7001`)

`--prompt n` — change rascontrol prompt as follows:
`0` - prompt `'>'`
`1` - prompt `'rasc>'`
`2` - prompt `'user:host>'`
(default: 2)

`--quiet` — quiet, don't print header (default for `--login` and `--testlogin`)

`--login` — print login and password, obtained from interactive input, to `stdout`, then exit (see *Script Use* below)

`--interactive` — read login and password from environment variable `RASLOGIN` instead of requesting it interactively

`--execute cmd` — execute single *cmd* and exit (batch mode); all text following `-x` until end of line is passed as `command`; this option implicitly assumes `-e`

`--testlogin` — just do a login and nothing else to check whether the login/password combination provided in the `RASLOGIN` variable is valid

### Interactive Use

In interactive use, `rascontrol` will be invoked with the host parameter only. Following successful authentication, `rascontrol` accepts command line input from `stdin`.

Here is an example session (*`mypasswd`* will not be echoed on screen):

```
$ rascontrol
Login name: mylogin
Password: mypasswd
mylogin:localhost> define dbh h1 -connect /
mylogin:localhost> define db  d1 –dbh h1
mylogin:localhost> define srv s1 -host localhost
                           -type h -dbh h1
mylogin:localhost> up srv s1
mylogin:localhost> save
mylogin:localhost> exit
$
```

### Script Use

Alternatively to interactive login, user and password information can be taken from the environment variable `RASLOGIN`. This variant is suitable for batch scripting in conjunction with the `–x` option.

The following example shows how first the `RASLOGIN` is set appropriately:

```
$ export RASLOGIN=`rascontrol --login`
```

…and then a sample Unix shell script which starts all rasdaman servers defined in the system configuration, performing implicit login from the environment variable contents which has been obtained from the previous command and pasted into the shell script:

```
#!/bin/bash
export RASLOGIN=rasadmin:mytotallyencryptedpassword
rascontrol -x up srv -all
```

## Comments in Scripts

To enhance legibility of scripts, `rascontrol` accepts comments in the usual shell syntax: Lines beginning with a hash sign '`#`' will be ignored, whatever they may contain. An example is usage in shell *here documents* (type `man sh` in your favourite shell for further information on this feature):

```
$ rascontrol <<EOF
# this is the command submitted to rascontrol:
list srv -all
# now terminate rascontrol:
exit
# the following line terminates rascontrol input:
EOF
$
```

#### Prefabricated Script

To ease system installation, a shell script, `rasconfig.sh`, is delivered in the `bin` directory which contains initial server definitions to set up a rasdaman Classic Edition configuration (i.e., one server process).

Don't simply run this script as is; prior to running it you may want to first understand it and then adapt it to your local situation.

## 5.4 `rascontrol` Command List

#### Command Synopsis

| | |
|---|---|
| `help` | display information (general or about specific command) |
| `exit` | exit `rascontrol` |
| `list` | list info about the current status of the system |
| `up` | start server(s) |
| `down` | stop rasdaman server(s) or server manager(s) |
| `define` | define a new object |
| `remove` | remove an object |
| `change` | change parameters of objects |
| `save` | make configuration changes permanent |

In the remainder of this section, commands are explained in detail, sorted by the targets they affect.

## 5.5 Server Hosts

#### Define Server Hosts

`define host` *h* `-net` *n* `-port` *p*

| | |
|---|---|
| *h* | symbolic host name |
| `-net` *n* | set network host name to *n* |

`-port p` port on which the rasdaman manager will listen

#### Change Server Host Settings

```
change host h [-name n] [-net x] [-port p]
      [-uselocalhost [on|off] ]
```

`h` host name whose entry is to be updated

`-name n` change host name to `n`

`-net x` change network name to `x`

`-port p` change port number to `p`

`-uselocalhost [on|off]` use domain name `localhost` (IP address 127.0.0.1) instead of regular network host name; usually this speeds up communication a little (default: `on`)

Note that it is not possible to change network name or port for a host while this server is running.

#### Remove Server Host Definitions

```
remove host h
```

`h` host name whose entry is to be deleted

Remove host `h` from the definition table.

It is not possible to remove a host definition while the corresponding host has active servers.

#### Status Information

```
list host
```

List all hosts currently defined.

## *5.6 rasdaman Servers*

#### Define rasdaman Servers

```
define srv s -host h -type t -port p -dbh d
      [-autorestart [on|off] [-countdown c]
      [-xp options]
```

`s` a unique, not yet used name for the server

`-host h` name of the host where the server will run

`-type t` communication type: `t` is `r` for RPC, `h` for http

`-port p` the RPC *program number* for RPC servers (recommended: 0x2999001 - 0x2999999), TCP/IP port for http servers (recommended: 7002 - 7999)

`-dbh d` database host where the relational database server to which the rasdaman server connects will run

`-autorestart a`
for `a`=`on`: automatically restart rasdaman server after unanticipated termination
for `a`=`off`: don't restart
(default: `a`=`on`)

`-countdown c`
for `c`>`0`: restart rasdaman server after c requests
for `c`=`0`: run rasdaman server indefinitely
(default: `c`=1000)

`-xp options` pass option string `options` to server upon start
(default: no options, i.e., empty string)

Option `-xp` must be the last option. Everything following "`-xp`" until end of line is considered to be "`options`" and will be passed, at startup time, to the server; see Section 5.11 *Server Control Options* below for the list of options available.

**Change Server Settings**

```
change srv s [-name n] -type t [-port p] [-dbh d]
       [-autorestart [on|off] [-countdown c]
       [-xp options]
```

`s` change settings for server `s`

`-name n` change server name to `n`

`-port p` change port number to `p`

`-dbh d` new database host where the relational database server runs to which the rasdaman server connects

`-autorestart a`
for `a`=`on`: automatically restart rasdaman server after unanticipated termination
for `a`=`off`: don't restart

`-countdown c`
for `c`>`0`: restart rasdaman server after c requests
for `c`=`0`: run rasdaman server indefinitely

`-xp options` pass option string `options` to server upon start

Option `-xp` must be the last option. Everything following "`-xp`" until end of line is considered to be "`options`" and will be passed, at startup time, to the server; see Section 5.11 *Server Control Options* below for the list of options available.

Restrictions:

- The server host cannot be changed.
- The server name cannot be changed while the server is up.

- The new settings will be used only next time the server starts.

#### Remove rasdaman Server Definitions

`remove srv s`

| | |
|---|---|
| *h* | server name whose entry is to be deleted |

Remove server *s* from the definition table.

It is not possible to remove a server definition while the corresponding server is up and running

#### Status Information

`list srv [ s | -host h | -all ] [-p]`

| | |
|---|---|
| *s* | give information about server *s* |
| `-host` *h* | give information about all servers running on host *h* information is requested |
| `-all` | list information about all servers on all hosts (default) |
| `-p` | additionally list configuration information |

The first is variant prints status information of the currently defined server(s); if *s* is provided, then only server `s` is listed.

## *5.7 Database Hosts*

#### Define Database Hosts

`define dbh h [-connect c]`

| | |
|---|---|
| *h* | a unique symbolic database host name, usually the host machine name |
| `-connect` *c* | the connection string used to connect `rasserver` to the database server |
| `-user` *u* | the user name (optional) used to connect `rasserver` to the base DBMS server; for PostrgreSQL, using this parameter automatically implies trust authentication. |
| `-passwd` *p* | the password (optional) used to connect `rasserver` to the base DBMS server; for PostrgreSQL, using this parameter automatically implies trust authentication. |

#### Change Database Host Settings

`change dbh h [-name n] [-connect c]`

| | |
|---|---|
| *h* | database host whose entry is to be changed |
| `-name` *n* | change symbolic database host name to *n* |
| `-connect` *c* | change connect string to *c* |

`-user u` the user name used to connect `rasserver` to the base DBMS server; using this optional parameter automatically implies ident-based authentication.

`-passwd p` the password used to connect `rasserver` to the base DBMS server; using this optional parameter automatically implies ident-based authentication.

The connection parameters can be changed at any time, however the servers will get the information only when they are restarted.

#### Remove Database Host Definitions

`remove dbh h`

`h` database host name whose entry is to be deleted

Remove database host `h` from the definition table.

It is not possible to remove a database host definition while this database host has active servers connected to it.

#### Status Information

`list dbh`

List all relational database hosts currently defined.

## *5.8 Databases*

Databases represent the physical database itself, together with the relational database server accessing them. It is possible to have multiple database definitions in the rasdaman server environment which are distinguished by the database host; the interpretation, then, is that the same contents (be it the same physical database or a mirrored copy) is available through relational servers running on the different hosts mentioned. In other words, when a client opens a database, the server manager can freely choose any of the database hosts on which the database indicated is defined.

The pair (database,database host) must be unique.

#### Define Databases

`define db d –dbh db`

`d` define database with name `d`

`-dbh db` set database host name to `db`

#### Change Database Settings

`change db d -name n`

`d` database whose name is to be changed

`-name n` change to new database name `n`

#### Remove Database Definitions

```
remove db d –dbh db
```

| | |
|---|---|
| *d* | name of database to be removed |
| -dbh *db* | host name of database to be removed |

Remove definition of database *d* from the definition table. The database itself remains unchanged, it is not physically deleted.

It is not possible to remove a database definition while the corresponding database has open transactions.

#### Status Information

```
list db [ d | -dbh h | -all ]
```

| | |
|---|---|
| *d* | give information about servers connected to database *d* |
| -dbh *h* | give information about all servers connected to database *d* via database host *h* |
| -all | list information about all servers connected to any known database (default) |

List relational database(s) defined.

## *5.9 Server Start-up and Shutdown*

#### Server Start

```
up srv [ s | -host h | -all ]
```

| | |
|---|---|
| *s* | start only server *s* |
| –host *s* | start all servers on host *h*; this requires that a manager has been started on this host previously. |
| –all | start all servers defined; note that only those servers can be started on whose host a manager is currently running. |

Look up the named server(s) in the definition list, and start the specified one(s) using the previously defined individual startup parameters.

At least one of the options *s*, -host *s*, and –all must be present.

#### Server Shutdown

```
down srv [ s | -host h | -all ] [-force] [-kill]
```

| | |
|---|---|
| *s* | name of the server to be stopped |
| –host *s* | terminate all servers on host *h* |
| –all | terminate all servers |

| | |
|---|---|
| `-force` | `send SIGTERM` immediately, don't wait for transaction end |
| `-kill` | `send SIGKILL` immediately, don't wait for transaction end |

This command shuts down the indicated server(s). At least one of the options *s*, `-host` *s*, and `-all` must be present.

Without `-force` and `-kill`, the server is marked for shut down and will actually be terminated by sending `SIGTERM` after completing the current transaction. With `-force` and `-kill`, the server is terminated instantaneously; this should be handled with extreme caution, as experience shows that relational database systems react differently on such a situation: usually a running transaction is aborted (which is the desired behavior), but sometimes the running transaction is committed (most likely leaving the database in an inconsistent state). See a Unix manual for the difference between `SIGTERM` and `SIGKILL` signals.

The manager on host *h* is not terminated.

## *5.10 Users and Their Rights*

Similarly to operating systems, rasdaman knows named users with access rights associated to them. Each rasdaman client must log in to the system under a specific login name using its specific password; this holds for database clients as well as for database administration. With each login name, a set of rights is associated which determines the set of actions admitted to the user under this login.

To this end, the rasdaman administrator manages user login names (user names) equipped with a password and rights to access the databases.

Attention: There is no way to retrieve a lost password!

The set of known logins as well as the associated rights all are under administrator control; the `define` and `remove` commands serve to add or delete user logins, the `change user` command allows to individually assign rights to a login.

In the rasdaman system's initial state after installation, user `rasadmin` is defined owning all possible rights (see below). A further user `rasguest` is defined which owns read-only access ("`R`") rights.

For both users, the password initially is identical with the user name. It is highly recommended to change this immediately using the `raspasswd` utility (See Section 6).

### Define New User

```
define user u [-passwd p] [-rights r]
```

| | |
|---|---|
| *u* | login name, must be unique (i.e., not yet existing) |
| *-passwd p* | set login password to pass<br>(default: user name) |
| -rights *r* | rights associated with this login<br>(default: R, i.e., read-only) |

The user's password can be changed at any time using the `raspasswd` utility (see Section 6).

### Remove User

```
remove user u
```

| | |
|---|---|
| *u* | login name to be removed |

The user is removed from the login list and henceforth cannot login to the rasdaman system any more.

### User Rights

User rights are indicated by upper case letters. They are divided into two categories: *system rights* and *database rights*. System rights apply to the whole system configuration of a server machine, whereas database rights can be specified individually for a database.

The following system rights are defined:

| | |
|---|---|
| C | user may change the system configuration |
| A | access control: the user may perform user management |
| S | start/stop right: the user may start and stop the system, in particular: rasdaman servers |
| I | info retrieval: the user may retrieve server status information |

The following database rights are defined:

| | |
|---|---|
| R | user is allowed read data (`select...from...where`) from rasdaman databases |
| W | user is granted write access (`update`, `insert`, `delete`) to rasdaman databases |

### Notation of Rights

In the `change user` command used for user rights administration, a user's rights set is described by a *rights string*. It is built from letters denoting the rights to be granted.

To revoke a right, leave out the corresponding character. To grant no rights at all, use – (minus sign).

No blanks or other characters are allowed in a rights string.

Examples of valid rights strings are:

grant all rights: `CASIRW`

grant read access only: `R`

grant no rights at all: `-`

These are examples for *invalid* rights strings:

Blanks between rights: `CA SIR  W`

Invalid characters I: `AXYZS`

Invalid characters II: `"A_+S`

#### Change User Attributes

`change user u [-name n | -passwd p | -rights r]`

Options:

| | |
|---|---|
| *u* | user login to be updated |
| `-name` *n* | change user name to *n* |
| `-passwd` *p* | change password to *p* |
| `-rights` *r* | change rights of user *u* according to rights string *r* |

Change name of user, login password, or user rights.

To change the password it is recommended to use the `raspasswd` utility instead of the `change user` command, as `raspasswd` does not echo the plain password on screen while `rascontrol` does.

#### Status Information

`list user [-rights]`

| | |
|---|---|
| `-rights` | additionally list rights assigned to each user |

List all user names currently defined, optionally with their rights.

## *5.11 Server Control Options*

The following options can be passed to the server when it is started by the server manager using the `up srv` command. Option settings are defined for a particular server using the `rascontrol` command `change srv -xp` which passes the rest of the line after `-xp` on to the server upon starting it (see Section 5.6).

`--enablefs` store new tiles in operating system files
(default: store new tiles in database).

`--log` *logfile*
print log to *logfile*.
If *logfile* is `stdout`, then log output will be printed to

standard output.
(default: `$RASDAMAN/log/serverid.log`)

`--mgmntint [m]`
: server management interval in seconds for garbage collection (default: 120 sec)

`--notimeout` server does not check for client timeouts (default)

`--timeout [t]` client time out in seconds for sign-of-life signal.
If no `t` indicated: 300 sec; if set to 0, no sign-of-life check is done.
Activated only if `--mgmntint` is also set.

`--transbuffer b`
: set maximum size of transfer buffer to `b` bytes
(default: 4 MB = 4,194,304 bytes)

`--cachelimit c`
: upper limit of cache area in bytes
(default: 0)

`--enable-tilelocking`
: perform tile-level locking on insert / update / delete
(default: whole database is locked)

## *5.12 Distributed Query Processing*

Rasdaman can form a federation network for query answering. In such a setup, `rasmgr`s facing congestion (i.e., all `rasserver` worker processes busy) will try to acquire a free server from some other `rasmgr`'s holding in the federation.

### Session-based server assignment

As always in rasdaman, acquisition and release of server processes is done on session level: when a client opens a new connection, it gets a server assigned; when it closes the connection, this server is released and put back into the pool of available processes. Hence, for optimal load balance clients should strive to have short-running sessions and not keep open connections unduly for a long time.

### Federation network

The federation network is defined in a decentralized way: each `rasmgr` knows peers from which it accepts requests, and to which it can send requests. To this end, each `rasmgr` maintains an `inpeer` and `outpeer` list:

- The `inpeer` list contains those hosts from which this node's `rasmgr` will accept requests.
- The `outpeer` list contains those hosts which this node's `rasmgr` will ask for server processes on local session overflow.

By manipulating these two lists administrators can exercise fine-grain security policy in a rasdaman federation network.

Note that the federation connectivity graph is not necessarily symmetric: a `rasmgr` may send requests to some other `rasmgr`, but not accept requests, and vice versa, depending on the individual configuration.

Each host individually respects these statements, there is no global rasdaman federation configuration.

### Federation node addressing

Addressing is based on hostnames, where a hostname in the sequel is one of

- a domain name, resolvable by this `rasmgr`'s host
- an IP address

All `inpeer` and `outpeer` statements accumulate so that host identifiers can be added and removed incrementally.

#### Security

A `rasmgr` request for a server process on another host is treated by the incoming host in the same way as any such incoming client request. The requesting `rasmgr` authenticates via the login and password which the originating client used for authenticating against rasdaman in the first place.

This implies that a client approaching such a federation must be known in all federation nodes. See Section 5.10 for details on users and the various permissions they can have on a database.

If neither any `inpeer` nor any `outpeer` is defined (either interactively through `rascontrol` or by way of settings in `rasmgr.conf`) then this rasdaman instance will act completely standalone and will neither send nor accept peer requests.

#### Define peers

`define inpeer` *`hostname`*

| | |
|---|---|
| *`hostname`* | host from which requests for rasdaman server process assignment will be accepted by this `rasmgr` |

`define outpeer` *`hostname`* `[-port` *`portnumber`*`]`

| | |
|---|---|
| *`hostname`* | host from which this `rasmgr` may request a rasdaman server process |
| *`portnumber`* | port number at which the `rasmgr` on that host is listening (default: `7001`) |

**List peers**

```
list inpeer
```

```
list outpeer
```

These commands list all currently defined inpeers and outpeers, respectively.

**Remove peers**

```
remove inpeer hostname
```

```
remove outpeer hostname
```

These commands remove hostname *hostname* listed from the list of peers.

**Examples**

```
define inpeer www.acme.com
```

```
define inpeer 192.168.28.10
```

**Caveat: fluctuating IPs**

In cloud environments, IP addresses are maintained dynamically and can change for a given host between reboots. Hence, when growing a rasdaman federation by launching new VMs care must be taken that the in- and outpeers received the proper current IP address.

**Restrictions**

In the current version, the queries are distributed only if the receiving rasmgr has no locally assigned rasservers. This limitation will be removed in the next release.

## *5.13 Miscellaneous*

**Help**

```
help
```

Display top level help page

```
help [command]
command help
```

Display information specific to *command*

(both syntax variants are equivalent)

**Version Information**

```
list version
```

`version` display rasdaman server version.

### Save Changes to Disk

```
save
```

The `save` operation writes the current configuration and authorization values to disk. All changes done during the session thus become permanent.

### `rascontrol` Termination

```
exit
```

terminates `rascontrol`.

# 6 Database User & Password Administration

The `raspasswd` utility allows users to interactively change their password. After requesting the user name and the current password, the user is prompted for the new password is twice to exclude typing errors.

### Invocation Synopsis

```
$ raspasswd [--help] [--port p]
```

where

| | |
|---|---|
| `--help` | this help text |
| `--host` *h* | server manager host name<br>default: `localhost` |
| `--port` *p* | port number of server manager |

### Example

```
$ raspasswd localhost
Login name: mylogin
Password: mypasswd
New password: mynewpasswd
```

```
Retype new password: mynewpasswd
$
```

Note that none of the passwords actually is echoed.

# 7 Example Database and Programs

## *7.1 Example Database*

A demonstration database is provided as part of the delivery package which contains the collections and images described in the *Query Language Guide*. To populate this database, first install the system as described here in the *Installation Guide* and the base DBMS specific *External Products Integration Guide*, and then invoke `rasdaman_insertdemo.sh` in the `bin` directory. This script makes use of the example images sitting in the `examples` directory, as well as the `insertppm` executable described below.

It is recommended to populate this demo database – it occupies only marginal disk space – first: Successful generation of this database shows overall successful rasdaman installation.

Before the test programs can be used, the demo database has to be created and schema information has to be imported. The following command line creates the database *`RASBASE`* (see also Section 2):

```
$ rasdl --basename RASBASE --createdatabase
```

The following imports schema information:

```
$ rasdl --basename RASBASE
        --read examples/rasdl/basictypes.dl --insert
```

Finally, the following line establishes the demo database (using a script from the `bin` directory which itself relies on the `insertppm` executable):

```
$ rasdaman_insertdemo.sh base
```

It is not important whether the rasdaman server is running during `rasdl` execution, however, the server is required for the `rasdaman_insertdemo.sh` script, as this is a client application.

For further information on `rasdl` see the *C++ Programming Guide*.

## *7.2 Example Programs*

Several example programs are provided in the `c++` and `java` subdirectories of `$RASDAMAN/share/rasdaman/examples`. Each directory contains a Makefile plus several `.cc` and `.java` sources, resp.

**Makefile**

The `Makefile` serves to compile and link the sample C++ sources files delivered. It is a good source for hints on the how-tos of compiler and linker flags etc.

**`insertppm.cc`**

The `insertppm` program inserts a PPM / PGM / PBM image into a rasdaman database. This program serves a dual purpose: On the one hand, it shows how a custom application can be created which inserts single images into a database from a file in some data exchange format. On the other hand, it is the utility to establish the sample database used in the *Query Language Guide.*

A compiled version of this program can also be found in the `bin` directory to avoid that this important utility is overwritten by exploring the sample code.

**`lookup.cc` and `lookup.java`**

The `lookup` program reads a specified collection and prints the MDDs and their content. Any collection from the demo database can be inspected, but be warned of the data volume generated by ASCII printouts.

**`avg-cell.cc` and `avg-cell.java`**

This program computes the average cell value from all images of a given collection.

Note that it requires grayscale images! A good candidate collection is `mr` from the demo database.

**`avg-cell-red.cc` and `avg-cell-red.java`**

Same as `avg-cell`, but takes the red component of an RGB image for averaging.

A good candidate collection is `rgb` from the demo database.

**Note**

All programs, once compiled and linked, print a usage synopsis when invoked without parameter.

# 8 Troubleshooting

## *8.1 Installation*

### 8.1.1 Cannot start rasdaman server

**Question:**

Upon startup of the server, I keep on getting a message

```
Checking if I am the only rasmgr on this
machine....Error: rasmgr instance already active.
```

…however, no rasmgr process is active.

**Answer:**

A previous forced termination of rasmgr may have left open the portmapper connection. Find it out via

```
rpcinfo -p
```

…and then delete it by issuing, as superuser:

```
rpcinfo -d n 1
```

where `n` is the pertaining address.

## *8.2 PostgreSQL Known Issues*

### 8.2.1 Transaction Warnings

Upon committing or aborting a PostgreSQL transaction (e.g., in the course of committing or aborting a rasdaman transaction, or upon closing a database or a connection) a warning is issued in the rasdaman server's log file:

```
commitTA...Warning/error in TransactionIf::abort() ROLLBACK:
SQLSTATE: 25P01 SQLCODE: -604
```

Simultaneously, the PostgreSQL server may issue one of the warnings below:

```
WARNING:  there is already a transaction in progress
```

```
WARNING:  there is no transaction in progress
```

Both warnings can be safely ignored.

### 8.2.2 Long Transaction Open

In very rare circumstances it has been observed that a PostgreSQL "begin transaction" command sometimes can take up to several seconds. Currently no reason is known for this phenomenon.

It is recommended to reuse transactions as much as possible to avoid the overhead of opening new transactions.

### 8.2.3 Disk Space

The PostgreSQL database grows as data are inserted. Make sure that sufficient disk space is available.

It seems that, when needed, parts of the database can be relocated to other file systems using symbolic links with the same names (with the DBMS server duly being shut down during this reorganization, of course); however, this has not been verified systematically yet.

In any case, it is recommended to regularly run

```
vacuumlo
```

### 8.2.4 3D ingest performance

While usually very satisfactory, under certain circumstances PostgreSQL BLOB performance degrades. Specifically, this has been observed when BLOBs undergo a heavy update frequency.

This happens, for example, when an image timeseries is updated time-slice by timeslice, in presence of directional tiling to favour temporal access: tile cubes are updated with every slice, leading to excessive tile traffic between rasdaman and PostgreSQL. As a consequence, PostgreSQL gets particularly slow during ingest.

As a workaround, a tile cache has been implemented in rasdaman (cf. Section 3.3).

Performance is at high speed for 2D data and for 3D climate data, for example.