EGEE

R-GMA command line tool

Document identifier:	**EGEE-JRA1-TEC-cli-manual**
Date:	**February 17, 2007**
Activity:	**JRA1: Middleware Engineering and Integration (UK Cluster)**
Document status:	**DRAFT**
Document link:	

Abstract: This document describes the R-GMA command line tool

CONTENTS

1 INTRODUCTION

The R-GMA command line tool provides simple shell-like access to the R-GMA distributed information and monitoring system. R-GMA uses a relational model to publish and query information using the SQL language.

For more information about R-GMA and for detailed installation and configuration instructions, see the R-GMA web page:

```
http://hepunx.rl.ac.uk/egee/jra1-uk/glite/doc/index.html
```

For more information about the SQL language, try the following tutorials:

```
http://www.firstsql.com/tutor.htm
http://www.w3schools.com/sql/default.asp
```

1.1 INSTALLATION

The R-GMA command line tool requires the R-GMA Python client library and its dependencies. You will need to install the following packages:

```
glite-rgma-base
glite-rgma-api-cpp
glite-rgma-api-c
glite-rgma-api-python
glite-rgma-command-line
```

To configure the R-GMA client library, first ensure that the environment variable `RGMA_HOME` is set to the prefix of your local R-GMA installation, e.g.

```
export RGMA_HOME=/opt/glite
```

Then run the `rgma-setup.py` script to configure the locations of the local R-GMA server, registry and schema:

```
$RGMA_HOME/share/scripts/rgma/rgma-setup.py
```

1.2 STARTING THE R-GMA COMMAND LINE TOOL

To start the R-GMA command line tool, run the following command:

```
$RGMA_HOME/bin/rgma
```

Alternatively if your system is set up to include `$RGMA_HOME/bin/` in the `$PATH`, you can just use:

```
rgma
```

On startup you should a message similar to the following:

```
Welcome to the R-GMA virtual database for Virtual Organisations.
================================================================

Your local R-GMA server is:

  https://<server-host>:8443/R-GMA

You are connected to the following R-GMA Registry services:

  https://<registry-host>:8443/R-GMA/RegistryServlet
  https://<registry-replica>:8443/R-GMA/RegistryServlet

You are connected to the following R-GMA Schema service:

  https://<schema-host>:8443/R-GMA/SchemaServlet

Type "help" for a list of commands.

rgma>
```

The number of registry replicas may vary depending on how R-GMA has been deployed.

1.3 ENTERING COMMANDS

Commands are entered by typing at the `rgma>` prompt and hitting “enter” to execute the command. A history of commands executed can be accessed using the Up and Down arrow keys. Reverse incremental search of command history is also supported - use CTRL-R and type the first few letters of the command you want to recall. Commands can be entered in lower or upper case.

Command autocompletion is supported – hit the “Tab” key when you have partly entered a command and it will either be completed automatically if possible. If there is more than one possibility, hit “Tab” again to distplay a list of matching commands. Table names will be autocompleted in appropriate places using the list of tables obtained at startup or in the most recent `show tables` command. [1]

The `#` character can be used to add comments to commands - any text following this character will be ignored.

2 COMMANDS

2.1 GENERAL COMMANDS

`help`	Displays general help information
`help <command>`	Displays help for a specific command
`help examples`	Display list of example commands
`exit`	Exit the R-GMA command line
`quit`	Same as “exit”

[1] In some circumstances, files in the current working directory may be presented as options for command completion! This is caused by a bug present in some versions of the Python standard library.

2.2 Querying data

Querying data uses the standard SQL `SELECT` statement, e.g.

```
rgma> SELECT * FROM ServiceStatus
rgma> SELECT Name, Endpoint FROM Service
rgma> SELECT Service_Name, Message, Status FROM ServiceStatus ORDER BY Status
```

The behaviour of `SELECT` varies according to the type of query being executed. In R-GMA there are three basic types of query:

`LATEST`	Queries only the most recent tuple for each primary key
`HISTORY`	Queries all historical tuples for each primary key
`CONTINUOUS`	Returns tuples continuously as they are inserted

The type of query can be changed using the `SET QUERY` command:

```
rgma> SET QUERY LATEST
rgma> SET QUERY CONTINUOUS
```

The current query type can be displayed using the `SHOW QUERY` command.

The maximum age of tuples to return can also be controlled. By default there is no limit. To limit the age of latest or historical tuples use the `SET MAXAGE` command which takes a value and a time unit (seconds, minutes, hours, days - default is seconds). For example, the following are equivalent:

```
rgma> SET MAXAGE 2 minutes
rgma> SET MAXAGE 120
```

In the case of a continuous query, the `MAXAGE` parameter has a slightly different effect. If a maximum age is specified, the query will initially return a history of matching tuples up to the specified maximum age. It will then return new tuples as they are inserted.

To disable the maximum age, set it to `none`:

```
rgma> SET MAXAGE none
```

The current maximum tuple age can be displayed using the `SHOW MAXAGE` command.

The final property affecting queries is the timeout. This controls how long the query will execute for before exiting automatically. For a latest or history query the timeout exists to prevent a problem (e.g. network failure) from stopping the query from completing. For a continuous query, the timeout indicates how long the query will continue to return new tuples. The default timeout is 1 minute and it can be changed using the `SET TIMEOUT` command, which has a similar syntax to `SET MAXAGE`, i.e. the following are equivalent:

```
rgma> SET TIMEOUT 3 minutes
rgma> SET TIMEOUT 180
```

The current timeout can be displayed using the `SHOW TIMEOUT` command.

2.3 INSERTING DATA

The SQL INSERT statement may be used to add data to the system:

```
rgma> INSERT INTO Table VALUES ('a', 'b', 'c', 'd')
```

In R-GMA, data is inserted into the system using a Producer component which handles the `INSERT` statement. Using the command line tool you may work with one producer at a time. If you change the properties of the producer, a new one is created and all `INSERT` statements will go to the new producer.

Producers may be configured to answer the three different types of query. All producers can answer continuous queries, but ability to answer latest and history queries is optional and is changed using the `SET PRODUCER` command:

```
rgma> SET PRODUCER latest
rgma> SET PRODUCER latest history
rgma> SET PRODUCER continuous
```

The default setting is for a producer that only answers continuous queries. The current producer type can be displayed using the `SHOW PRODUCER` command.

An alternative to enabling latest and history queries for the producer is to use a secondary producer to provide these capabilities. See the section on secondary producers for more details.

A producer may have a predicate associated with it describing the subset of a table it provides. This is not compulsary but it enables queries to be answered more efficiently. For example, if a table Service has the column `Site` which for your producer will always have the value `myhost.com`, you can express this restriction using:

```
rgma> SET PRODUCER PREDICATE Service WHERE Site = 'myhost.com'
```

A producer predicate may only specify fixed values for columns using = and `AND`. Other operations (e.g. `<, >=, <=, OR`) should not be used (although they can be used in queries).

The current producer predicate for a table can be displayed using:

```
rgma> SHOW PRODUCER PREDICATE <table name>
```

If you define a producer predicate you must ensure that all inserted tuples comply with it or an error will result. To remove the predicate use:

```
rgma> SET PRODUCER PREDICATE <table name> none
```

For a producer that can answer latest and/or history queries, tuples must be deleted after some time interval to prevent the system becoming full of old data. This is controlled by the latest and history retention periods for a producer. The producer will return tuples to Latest queries provided they are younger than the latest retention period (LRP), and similarly for history queries and the history retention period (HRP). These times are 10 minutes by default, and may be controlled by:

```
rgma> SET PRODUCER latestretentionperiod 30 minutes
rgma> SET PRODUCER historyretentionperiod 2 hours
```

To show the current producer HRP/LRP, use the commands `SHOW PRODUCER HRP` and `SHOW PRODUCER LRP`.

2.4 SECONDARY PRODUCERS

A Secondary producer does not insert new data to the system, but collects data from individual Producers and makes it available via its own Producer component. The main uses of a Secondary producer are:

- To allow latest and/or history queries on a table even though the individual producers only support continuous queries.
- To allow a join query on two tables that may be produced by different producers.
- To bring data from R-GMA into a particular database so it can be worked on by other tools.

Using the R-GMA command line you can work with one Secondary producer at a time, which can consume from multiple tables. If the properties of the secondary producer are changed a new one is created, and is automatically set up to consume from the same tables as the previous one.

To instruct the secondary producer to consume from the table `MyTable`, use the following command:

```
rgma> SECONDARYPRODUCER MyTable
```

Like the producer, the secondary producer may be configured to answer latest and/or history queries:

```
rgma> SET SECONDARYPRODUCER latest
```

By default the secondary producer can only answer latest queries. The current secondary producer type can be displayed using the `SHOW SECONDARYPRODUCER` command.

If the secondary producer can answer history queries, it has an associated history retention period, as for a primary producer. This is controlled in the same way:

```
rgma> SET SECONDARYPRODUCER historyretentionperiod 1 day
```

To show the current secondary producer HRP, use the command `SHOW SECONDARYPRODUCER HRP`. Secondary producers do not have a latest retention period as this is a property of each tuple and is inherited from the individual producers it consumes from.

2.5 ADMINISTRATION COMMANDS

To create a table called `MyTable`: [2]

```
rgma> CREATE TABLE MyTable
```

To drop a table called `MyTable`:

```
rgma> DROP TABLE MyTable
```

[2]The create/drop table commands are not yet implemented

2.6 INFORMATION COMMANDS

To show a list of all R-GMA producers that produce the table `MyTable`:

```
rgma> SHOW PRODUCERS OF MyTable
```

To show a list of all table names:

```
rgma> SHOW TABLES
```

To show information about a table `MyTable`:

```
rgma> DESCRIBE MyTable
```

To show a summary of properties for the current session:

```
rgma> SHOW PROPERTIES
```

2.7 DIRECTED QUERIES

Normally a component of R-GMA called the “mediator” selects which Producers are contacted to answer a query. For debugging purposes it may be useful to specify a particular Producer to use instead. This is called a “directed query” and can be specified with the `USE PRODUCER` command:

```
rgma> USE PRODUCER <url> <resource id>
```

All future `SELECT` queries will be directed to this Producer. Only one producer may be specified. The `<url>`and `<resource id>` should correspond to a valid Producer that can answer the type of queries you put to it or no results will be returned. The `SHOW PRODUCERS` command displays urls and resource IDs of registered Producers.

To revert back to using the Mediator to select producers, use the command:

```
rgma> USE MEDIATOR
```

2.8 SESSION HISTORY

A history of commands executed in a session is recorded in order to allow a session to be replayed. To display the current session history, use the `SHOW HISTORY` command.

To clear the current session history, use the `CLEAR HISTORY` command. To write the session history to a file, use the `WRITE HISTORY` command.

Example:

```
rgma> SET QUERY latest
Set query type to latest
rgma> SELECT * FROM userTable

+--------+---------+-------+-------+-----------------+-----------------+
| userId | aString | aReal | anInt | MeasurementDate | MeasurementTime |
+--------+---------+-------+-------+-----------------+-----------------+
| my_id  | a       | 1.2   | 3     | 2005-06-24      | 12:56:03        |
| my_id2 | a       | 1.2   | 3     | 2005-06-24      | 12:56:19        |
| my_id3 | a       | 1.2   | 3     | 2005-06-24      | 12:56:29        |
+--------+---------+-------+-------+-----------------+-----------------+
3 Rows in set

rgma> SELECT * FROM JobStatusRaw
No results returned
rgma> SHOW HISTORY
SET QUERY latest
SELECT * FROM userTable
SELECT * FROM JobStatusRaw
rgma> WRITE HISTORY commands.rgma
Session history written to file 'commands.rgma'
rgma> CLEAR HISTORY
Session history cleared
rgma> SHOW HISTORY
rgma>
```

Note that `HELP` commands are not included in the history, and nor are the `SHOW HISTORY`, `CLEAR HISTORY`, or `WRITE HISTORY` commands. However, comments, unrecognized commands and commands which produced errors are preserved in the history. Furthermore, this session history is independent of the command history accessed using the `up` and `down` keys. This command history cannot be cleared and preserves all commands.

2.9 Abbreviations

The following abbreviations may be used:

p	producer
sp	secondaryproducer
lrp	latestretentionperiod
hrp	historyretentionperiod
q	quit

2.10 Other commands

```
rgma> read <filename>
```

Read and execute commands in the specified file. Only one command may be specified per line.

```
rgma> set output <format>
```

Set the output format for the results of queries. Supported formats are `table` (the default), `csv` (comma-separated) and `tsv` (tab-separated).

```
rgma> write results <filename>
```

Write the results of all SELECT queries to a file. If `<filename>` is `stdout` then results will be written to standard output (i.e. the screen by default).

```
rgma> USE VO MyOrganization
```

Multi-VO support is not implemented yet.

3 BATCH MODE

The command line tool can be used in batch mode in three ways:

3.1 EXECUTING COMMANDS DIRECTLY

```
rgma -c <command>
```

Executes `<command>` and exits. The `-c` option may be specified more than once in which case the commands are executed in the order they appear on the command line.

3.2 EXECUTING COMMANDS FROM A FILE

```
rgma -f <file>
```

Executes commands in `<file>` sequentially then exits. Each line should contain one command.

3.3 EMBEDDING IN A SHELL SCRIPT

Commands may be embedded in a shell script and executed as follows:

```
#!/bin/sh

$RGMA_HOME/bin/rgma <<EOF
set query latest
select Service_Name, Status FROM ServiceStatus WHERE Status != 0
EOF
```