



<http://intevation.net>

GREAT-ER II

Specification GREAT-ER Model Suite

Intevation GmbH
Georgstrasse 4
49074 Osnabrück
Germany

Date: July 28, 2003

This document has been designed with \LaTeX . It is available as source code, PDF- and HTML-format.

Version: 1.0.0

Date: July 28, 2003

Authors: Bernhard Herzog, Frank Koormann

Copyright (c) 2002 Intevation GmbH.

Contents

1	Concept	5
1.1	General Overview	5
1.2	Scheduler - Worker Concept	6
1.2.1	Simulation Run Flow	7
1.2.2	Database Access	7
1.3	Software	7
1.4	System segmentation	8
1.5	Parameter Information	10
1.5.1	Parameter Requirement Information	10
1.5.2	Parameter Distribution Information	10
1.5.3	Information Storage	11
2	Scheduler	13
2.1	Client-Scheduler Network Protocol	13
2.1.1	XMLRPC Methods	13
2.2	Scheduler-Worker Interface	17
2.2.1	The Worker's Perspective on the Scheduler	17
2.3	Configuration	18
2.3.1	Starting the Scheduler	19
2.3.2	Stopping the Scheduler	19
3	Worker	21
3.1	Scheduler-Worker Network Protocol	21
3.1.1	The Scheduler's Perspective on the Worker	21
3.2	Configuration	24
3.2.1	Starting the Worker	25
3.2.2	Stopping the Worker	25
A	GNU Free Documentation License	27

1 Concept

1.1 General Overview

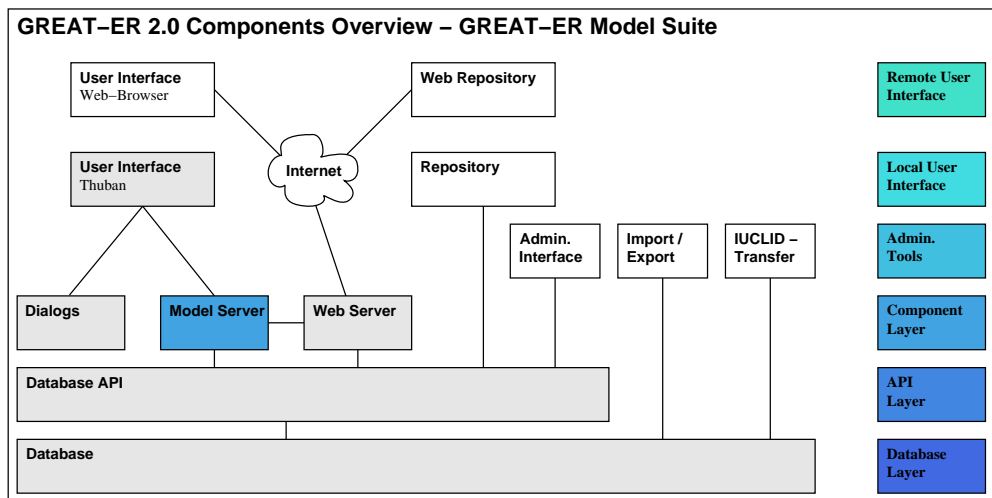


Fig. 1.1: GREAT-ER Model Suite Component Overview

Figure 1.1 gives an overview of the GREAT-ER II architecture and the components covered by this specification. Components with direct relation to GREAT-ER Model Suite are displayed in gray, others in white.

The core component is the GREAT-ER Model Suite, which

- provides required parameter lists
- checks current parameter settings for completeness
- performs simulation runs

Figure 1.2 illustrates the component details of the GREAT-ER Model Suite.

1.2 Scheduler - Worker Concept

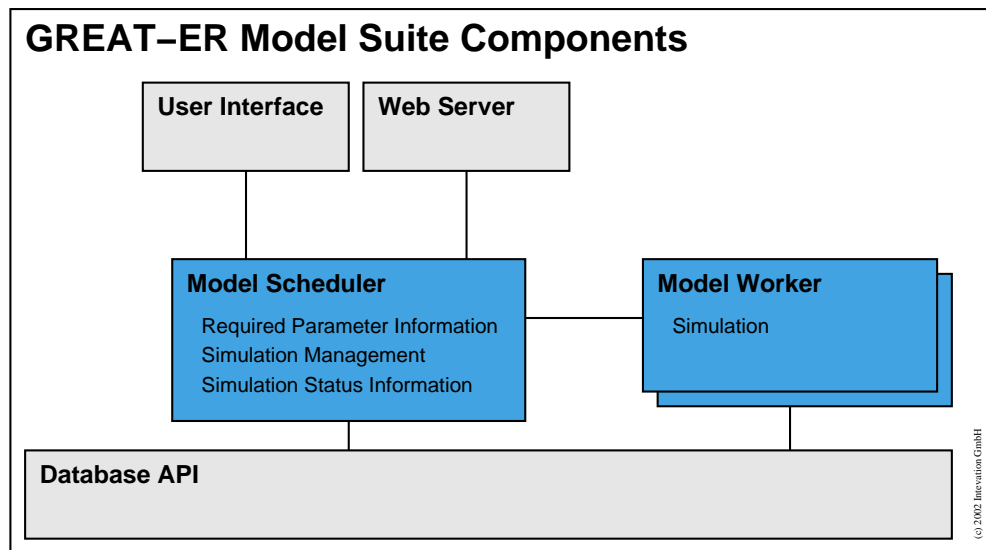


Fig. 1.2: GREAT-ER Model Suite Component Details

The model suite consists of a scheduler process (chapter 2) which communicates with the user interface and the web server and delegates the actual simulation to separate worker processes (chapter 3). Only the scheduler communicates with the worker processes.

The worker processes and the scheduler communicate via the network and may run on separate machines. This architecture allows the simulation to be distributed over several workers at once to speed up the simulation run and concentrates the knowledge about the models in the workers. The workers may of course also run on the same system as the scheduler which in turn could run on the same machine as the client.

The clients — the web-server and the user interface — connect to the server to start and stop simulations, to query the current status of a simulation or to retrieve information about the supported models. The scheduler can manage multiple concurrent simulations for different users and from different clients.

A worker works for only one scheduler process.

For any given simulation the scheduler will determine which workers it will use. It estimates how many workers would be optimal for the amount of work in the simulation and tries to use that many of workers. If at all possible the scheduler will only use workers that are not already working on a simulation. If all workers are already working on some simulation it will reuse one of the workers which then will have to work on two simulations at the same time. The worker accomplishes this by calculating single complete Monte-Carlo shots for each of the simulations it handles in turn.

1.2.1 Simulation Run Flow

A simulation run generally consist of the following steps:

1. The client calls the scheduler to start a simulation
2. The scheduler delegates the simulation to one or more worker processes. The worker processes may run on separate machines or on the same one.

This step includes generation of (pseudo) random numbers for use in the Monte-Carlo shots in the workers. The random numbers are computed in the scheduler to make sure that there's no accidental correlation due to two workers using the same seeds and algorithms in the random number generators. It also allows the use of more systematic approaches in the future like Latin-Hypercube and Hammersley whose implementations need central coordination.
3. When a worker is finished with its part of a simulation they transfer the aggregated results back to the scheduler.
4. When all workers taking part in a simulation have completed their work the scheduler aggregates the results of the individual workers into the final simulation result.
5. The simulation result is written into the database.
6. The client determines that the simulation is completed. The client has to actively poll for the state of the simulation so it can't be informed when the simulation has been finished (see section 2.1.1 for more information)

1.2.2 Database Access

Both the scheduler and the workers need access to the database. Therefore the clients pass the userid and password to use for database access to the scheduler when starting a simulation. All database access for a given simulation run is done with these settings through the DB-API implementation.

1.3 Software

All software tools the GREAT-ER Model Suite are chosen under the conditions of being Free Software and providing platform independence.

The scheduler and worker are implemented in Python using the Twisted framework for network applications. Twisted is built around non-blocking sockets so that a program implemented with Twisted can be run in a single thread that is implemented in an event-oriented way.

1.4 System segmentation

The term system in this section means an environmental system under investigation.

The backbone of the system structure of GREAT-ER 1.0 is the river network. The network is split into stretches of different types: spring, confluence, bifurcation and discharge. Based on these elements the river network can be seen as a directed graph, where the first three stretch types have mainly topological meaning. The latter element is more special because an additional chain of models is embedded: Emission, sewer and WWTP. However, the graph structure is limited to the river network. Figure 1.3 illustrates the situation.

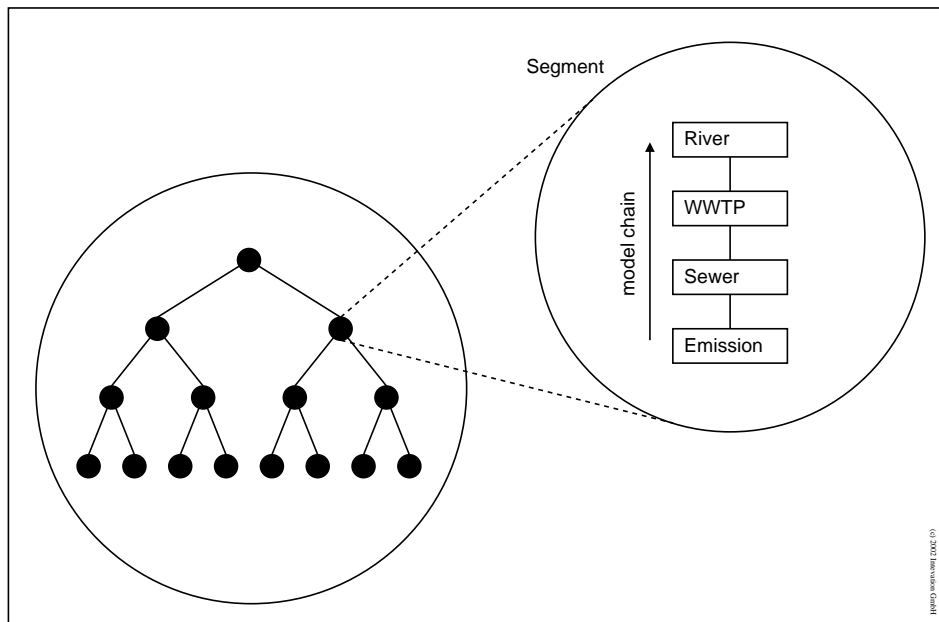


Fig. 1.3: System structure GREAT-ER 1.0

The approach for GREAT-ER II extends the segment concept to all models. The directed graph is not limited to the river network with a static chain of models linked to a discharge point as described above. The different models are represented by specific segments and therewith part of the directed graph (Figure 1.4). This approach provides higher flexibility in extending and combining the model collection.

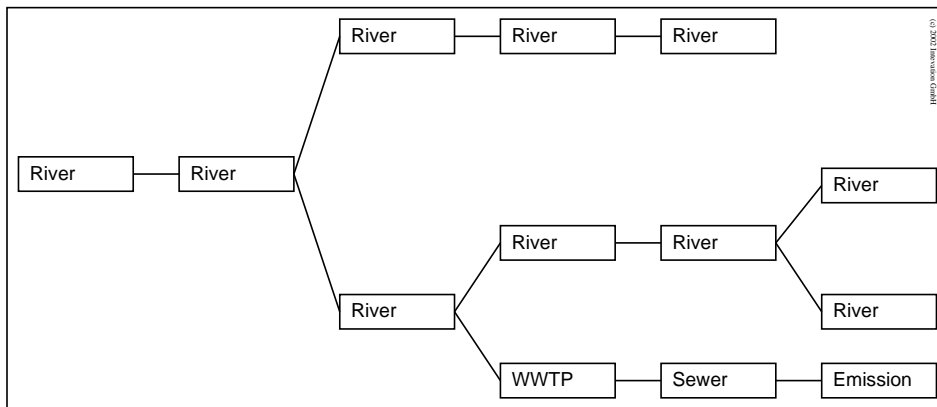


Fig. 1.4: New system structure for GREAT-ER II

1.5 Parameter Information

1.5.1 Parameter Requirement Information

Rationale

The user interface supports the user with filling parameters for a simulation. One aspect of this is an indicator on which parameters actually must be specified to run a simulation with the current model selection (i.e. if volatilization is switched on, certain parameters are required for the model equations - if volatilization is switched off, the user should clearly see that she has not to fill in the corresponding fields).

Component Communication Path

1. A model parameter Dialog is closed with OK.
2. The changes are committed to the database (i.e. to the temporary session).
3. The UI component requests a list of the required parameters from the Model Scheduler for the given session.
4. The Model Scheduler hands over the request to a Worker instance.
5. The Worker instance loads the parameter selection from the database and tests which parameters are required, compiles a list of (field_id, block_id) tuples and returns this list to the Model Scheduler.
6. The Model Scheduler returns the list to the UI component.
7. The UI component uses the list to set the required-indicators for the parameters. The next dialog that is opened uses this information and enables/disables the symbol for necessary parameters accordingly.

1.5.2 Parameter Distribution Information

Rationale

The GREAT-ER Model system provides a Monte-Carlo based simulation approach where some parameters are always distributed and others always fixed. A significant amount of parameters distribution can be determined by the user.

While the Scheduler communicates with the User Interface and has to control the simulation progress, the information about models and their requirements is implemented at the Worker side.

Component Communication Path

1. Selecting the "Simulation/Start" item the client writes the session data to the database and calls the Scheduler to run a simulation.

2. The Model Scheduler requests a Worker instance for a list of parameters to be considered as distributed for the given session.
3. The Worker loads the session settings from the database and based on the model selection and parameter settings compiles a list of (field_id, block_id, distribution type) tuples and returns this list to the Model Scheduler.
4. The Scheduler generates a batch of lists of (field_id, block_id, value from standard distribution¹) tuples representing single Monte-Carlo shots.
5. The batch of shots is split to the registered Worker instances.
6. For a given shot a Worker derives based on the descriptive distribution parameters the specific values for a segment.

1.5.3 Information Storage

The information about required parameters is provided by the models, which are implemented as classes. The base class Model provides methods to analyze the model requirements and the user settings. The requirements have to be implemented by a model developer as a nested list:

1. The first tuple of a list names a parameter pointing out if the current user settings imply requirement of the lists contents: (field_id, block_id, value to set the parameter²).
2. All further items of the list are either (field_id, block_id) tuples describing parameters or lists.

¹Standard distribution in this case means e.g. a standard normal distribution or a [0,1]-uniform distribution, etc.

²This is needed since parameters like RIVER_MODEL can flag different states and are not simply boolean.

2 Scheduler

2.1 Client-Scheduler Network Protocol

The protocol used between the clients and the scheduler process is XMLRPC. The data is communicated through the database. Some of the XMLRPC calls contain ids that refer to records in the database.

2.1.1 XMLRPC Methods

The scheduler implements the following methods:

```
simulation.start(dbname, username, password, sessid)
```

Name	<code>simulation.start</code>
Description	Start a new simulation and return its id.
In	<code>dbname, username, password, sessid</code>
In definitions	<code>dbname:</code> DB instance <code>username:</code> User name <code>password:</code> Password <code>sessid:</code> Session identifier
Return	<code>simulationid</code>
Return definitions	The returned <code>simulationid</code> is a signed integer which identifies the simulation run and has to be used when stopping the simulation or querying the status of the simulation.

```
simulation.stop(simulationid)
```

Name	<code>simulation.stop</code>
Description	Stop the simulation indicated by <code>simulationid</code> . The method has no explicit return value, the calling client can monitor the status of stopping the simulation by using the <code>simulation.status</code> call.
In	<code>simulationid</code>
In definitions	<code>simulationid</code> has to be the value returned by a <code>simulation.start</code> call.
Return	-
Return definition	-

`simulation.status(simulationid)`

Name	<code>simulation.status</code>
Description	Return the current status of the simulation indicated by <code>simulationid</code> .
In	<code>simulationid</code>
In definitions	<code>simulationid</code> references a running simulation.
Return	List [state, auxiliary value]
Return definitions	<p>The string <code>state</code> indicates the state of a simulation, the <code>auxiliary value</code> (which can be either a string or an integer) and its meaning depend on the state:</p> <p>"initializing" The scheduler is initializing the simulation, i.e generating the random numbers to use in the Monte-Carlo shots. This state roughly corresponds to step 2.</p> <p>The auxiliary value is an integer. It is a progress indicator increasing from 0 (just started) to 100 (finished initializing).</p> <p>"waiting" The simulation hasn't begun execution yet because all worker processes are already busy.</p> <p>The auxiliary value has no meaning and should be ignored.</p> <p>"running" The simulation is currently running. This state roughly corresponds to the time between steps 2 and 4.</p> <p>The auxiliary value is an integer. It is a progress indicator increasing from 0 (just started) to 100 (finished the simulation).</p> <p>"stopping" The simulation has told the workers to stop and is waiting for them to actually have stopped. This status is entered when the user told the scheduler to stop the simulation or when an error occurred in one of the workers.</p> <p>The auxiliary value has no meaning and should be ignored.</p> <p>"writing_results" The simulation is currently writing the simulation results into the database. This state corresponds to step 5.</p> <p>The auxiliary value is an integer. It is a progress indicator increasing from 0 (just started) to 100 (finished writing).</p>
<i>Continued on following page ...</i>	

... continued from previous page	
Name	<code>simulation.status</code>
Return definitions	<p>"end_complete" The simulation has finished the simulation successfully. The client can now read the results from the database. This state corresponds to step 6.</p> <p>The auxiliary value has no meaning and should be ignored.</p> <p>"end_error" An error has occurred during the simulation and the simulation has been aborted.</p> <p>The auxiliary value is a string with an error message.</p> <p>"end_stopped" The simulation has been stopped by the client (presumably by calling <code>simulation.stop()</code>).</p> <p>The auxiliary value has no meaning and should be ignored.</p> <p>Note that all end states are denoted by strings that start with "end_" so the client can easily check whether the simulation is still running.</p>

Since the server doesn't call back to the client, the client is responsible for calling `simulation.status` repeatedly to find out when the simulation is finished. Of course, this shouldn't be done more often than a few times per second at most.

```
session.required_parameters(dbname, username, password, sessid)
```

Name	<code>session.required_parameters</code>
Description	Return a list of those parameters <i>that are required</i> for the corresponding session with its current selections e.g. on model modes.
In	<code>dbname, username, password, sessid</code>
In definitions	<p><code>dbname:</code> DB instance</p> <p><code>username:</code> User name</p> <p><code>password:</code> Password</p> <p><code>sessid:</code> Session identifier</p>
Return	List of tuples (<code>blockid, fieldid</code>) or error code.
Return definitions	<ul style="list-style-type: none"> • <code>blockid</code> and <code>fieldid</code> are used as in the <code>PARA_TREE_DEF_TAB</code> and in combination uniquely identify a parameter in the database. • If no parameter is required, an empty list is to be returned.

`session.missing_parameters(dbname, username, password, sessid)`

Name	<code>session.missing_parameters</code>
Description	Return a list of those parameters <i>that are required and missing</i> (i.e. still set to unknown) for the corresponding session with its current selections e.g. on model modes.
In	<code>dbname, username, password, sessid</code>
In definitions	<code>dbname:</code> DB instance <code>username:</code> User name <code>password:</code> Password <code>sessid:</code> Session identifier
Return	List of tuples (<code>blockid, fieldid</code>) or error code.
Return definitions	<ul style="list-style-type: none"> • <code>blockid</code> and <code>fieldid</code> are used as in the <code>PARA_TREE_DEF_TAB</code> and in combination uniquely identify a parameter in the database. • If no parameter is required, an empty list is to be returned.

`scheduler.status()`

Name	<code>scheduler.status</code>
Description	Return a struct describing the status of the scheduler.
In	-
In definitions	-
Return	An XMLRPC struct describing the status of the scheduler.
Return definitions	The struct will have at least these fields: <code>num_workers:</code> Number of workers the scheduler may use

`scheduler.shutdown()`

Name	<code>scheduler.shutdown</code>
Description	Shutdown the scheduler. This method is not available on all schedulers. Typically it's only available if the scheduler was started by e.g. the GREAT-ER client on a single user machine. In that case the client starts the scheduler itself and thus also has to be able to shut it down.
In	-
In definitions	-
Return	-
Return definitions	-

2.2 Scheduler-Worker Interface

The scheduler and the worker processes communicate by the Perspective Broker protocol of the twisted framework. The connection is initiated by the worker process which provides the server with a perspective on itself described in section 3.1.1 and it sees the perspective of the scheduler described in section 2.2.1.

2.2.1 The Worker's Perspective on the Scheduler

The workers perspective on the scheduler has the following methods:

```
report_status(simid, state, parameter)
```

Name	<code>report_status</code>
Description	Tell the scheduler the current state of the calculations. The worker calls this method during a calculation started by the scheduler's call to the <code>start_simulation</code> method.
In	<code>simid, state, parameter</code>
In definitions	<code>simid:</code> id of the simulation the status is reported for <code>state:</code> status of the simulation <code>parameter:</code> parameter to detail the status information Simulation states are: "running" : The simulation is running and the corresponding <code>parameter</code> is a percentage as an integer in the range 0 to 100 indicating how much of the work has been completed.
Return	-
Return definitions	-

Since the simulation start was issued using the perspective broker framework the end of the simulation has not to be reported to the scheduler explicitly.

2.3 Configuration

The scheduler provides various options to configure the model system for an installation:

Option	Description
Client Port	The port to listen on for the XMLRPC calls from clients.
Worker Port	The port to listen on for worker connections.
Port Scan Range	This feature is used especially for the GREAT-ER Desktop Version and should not be used for GREAT-ER Web or computing servers! If specified (with a number of ports to scan) the scheduler will search for free ports for client and worker connections. The scan starts at the specified client port.
Allow Shutdown	This flag is used especially for the GREAT-ER Desktop Version and should not be used for GREAT-ER Web or computing servers! If specified the scheduler can be stopped with XMLRPC calls on the client port.
Admin Port	The port to listen on for administrative XMLRPC calls. The scheduler can be shut down via XMLRPC calls on this port if the calls originating from localhost. This feature is especially for the server usages of the GREAT-ER model systems since common users typically do not have accounts on dedicated servers.
Info File	If specified process name and admin port are stored in the file for later use (e.g. in a stop script, see below). This feature is especially for the server uses of the GREAT-ER model systems.
Log File	If specified the scheduler writes extensive logs into the file for activity monitoring and problem analysis.
Test Port	This feature is especially for the unit testing during development and can be used for special test commands.
DB-API Module	This feature is especially for the unit testing during development and can be used to specify dummy test databases.

2.3.1 Starting the Scheduler

The Scheduler can be started using the `start_scheduler.py` start script. The script provides several command line options to configure the scheduler:

```
Usage: python start_scheduler.py [options]
Options:
  --allow-shutdown    Allow the scheduler.shutdown XMLRPC call (default off)
  -p, --client-port=  The port to listen on for the XMLRPC calls
                      [default: 10101]
  -w, --worker-port=  The port to listen on for the worker connections. If
                      a port-scan-range is given this value will be ignored
                      and the worker port will be chosen automatically.
                      [default: 10102]
  --admin-port=       port to listen on for admin commands
  --info-file=        file to write info about the scheduler process to
                      (pid and admin port).
  --test-port=        Port number to listen on for special commands from the
                      test-suite
  --port-scan-range=  number of ports starting from client-port to scan for
                      a free port. The ports used will be printed to stdout.
  --log-file=         name of file to log to
  --db-api-module=    Name of the db interface implementation module
                      [default: _dagreater]
  --help              Display this help and exit.
```

2.3.2 Stopping the Scheduler

The scheduler can be stopped using the `stop.py` script. The script reads the information from a specified info file and stops the corresponding scheduler.

```
Usage: python stop.py [info-file]
```

The GREAT-ER Desktop Client stops a locally started scheduler automatically on exit.

3 Worker

The worker in the GREAT-ER Model System implements the models. A worker computes serialized simulations requested by the scheduler the worker is connected to. To get the information needed for a simulation a worker reads a session and associated data from the data base. A worker does not write to the database, simulation results are collected and written by the scheduler.

A single worker may be running (parts of) several simulations simultaneously. These parts must belong to different simulations.

3.1 Scheduler-Worker Network Protocol

This section describes the methods of the perspectives mentioned in 2.2 the worker implements.

3.1.1 The Scheduler's Perspective on the Worker

The scheduler's perspective on the worker has the following methods:

```
start_calculation(dbname, username, password, sessid, simid,
stochastic_values)
```

Name	start_calculation
Description	Calculate a number of shots specified by the stochastic values for the given session.
In	dbname, username, password, sessid, simid, stochastic_values
In definition	<p>dbname: DB instance username: User name password: Password sessid: Session identifier simid: Simulation id stochastic_parameters: Dictionary of tuples or None:</p> <ul style="list-style-type: none"> • Dictionary: A tuple of values for each stochastic parameter. • Number of Monte-Carlo shots given by length of tuples. • Dictionary keys are the tuples returned by session_stochastic_parameters. • If stochastic_parameters is None a single deterministic shot is calculated. <p>It is an error if the worker is already working on a calculation with the same simid.</p>
Return	List of aggregated results of the calculations. The number and structure of return values depends on the model selection
Return definitions	

```
stop_calculation(simid)
```

Name	stop_calculation
Description	Stop the currently running calculation.

```
session_stochastic_parameters(dbname, username, password,
sessid)
```

Name	<code>session_stochastic_parameters</code>
Description	Return a list of parameters that have to be varied throughout the simulation. Whether this variation is done by random numbers or a more systematic way is up to the scheduler.
In	<code>dbname, username, password, sessid</code>
In definitions	<code>dbname:</code> DB instance <code>username:</code> User name <code>password:</code> Password <code>sessid:</code> Session identifier
Return	List of Python tuples: <code>((blockid, fieldid, distribution, objectid), correlation)</code>
Return definitions	<code>blockid:</code> Parameter Block ID as defined in <code>PARAM_TREE_DEF_TAB</code> . Together with <code>fieldid</code> identifies a parameter. <code>fieldid:</code> Parameter Field ID as defined in <code>PARAM_TREE_DEF_TAB</code> . Together with <code>blockid</code> identifies a parameter. <code>distribution:</code> Indicating the distribution of the parameter, "uniform", "normal" or "lognormal". <code>objectid:</code> Used when a parameter has to be varied independent for each object it is related to. These are identified by the <code>objectid</code> . If the parameter is applied general the <code>objectid</code> is set to '-1'. <code>correlation:</code> Either None, if no correlation to be applied, or a numerical value [-1.0,1.0] or a (BlockID, FieldID) tuple to identify a parameter describing the correlation.

```
session_required_parameters(dbname, username, password, sessid)
```

Name	<code>session_required_parameters</code>
Description	Return the parameters required for a simulation in the given session.
In	<code>dbname, username, password, sessid</code>
In definitions	<code>dbname:</code> DB instance <code>username:</code> User name <code>password:</code> Password <code>sessid:</code> Session identifier
Return	List of Python tuples: (blockid, fieldid)
Return definitions:	blockid and fieldid are used as in the <code>PARAM_TREE_DEF_TAB</code> and in combination uniquely identify a parameter in the database.

```
session_missing_parameters(dbname, username, password, sessid)
```

Name	<code>session_missing_parameters</code>
Description	Return the parameters required for a simulation that are not yet specified in the given session.
In	<code>dbname, username, password, sessid</code>
In definitions	<code>dbname:</code> DB instance <code>username:</code> User name <code>password:</code> Password <code>sessid:</code> Session identifier
Return	List of Python tuples: (blockid, fieldid)
Return definitions:	blockid and fieldid are used as in the <code>PARAM_TREE_DEF_TAB</code> and in combination uniquely identify a parameter in the database.

3.2 Configuration

The worker provides various options to configure the model system for an installation:

Option	Description
Port	The port a running scheduler listens for connections.
Host	The host the scheduler is running on. By default this is localhost; with this option a larger computing can be set up distributing a simulation for large catchments or a high number of Monte-Carlo shots over several computers.
Admin Port	The port to listen on for administrative XMLRPC calls. The worker can be shut down via XMLRPC calls on this port if the calls originate from localhost. This feature is especially for the server usages of the GREAT-ER model systems since common users typically do not have accounts on dedicated servers.
Info File	If specified process name and admin port are stored in the file for later use (e.g. in a stop script, see below). This feature is especially for the server uses of the GREAT-ER model systems.
Log File	If specified the worker writes extensive logs into the file for activity monitoring and problem analysis.
Test Port	This feature is especially for the unit testing during development and can be used for special test commands.
DB API Module	This feature is especially for the unit testing during development and can be used to specify dummy test databases.

3.2.1 Starting the Worker

The Worker can be started using the `start_worker.py` start script. The script provides several command line options to configure the worker:

```
-p, --port=           The port the scheduler listens on for workers [default: 10102]
-h, --host=          The host the scheduler is running on [default: localhost]
--admin-port=        port to listen on for admin commands
--info-file=         file to write info about the worker process to (pid and
                    admin port).
--test-port=         Port number to listen on for special commands from the
                    test-suite
--log-file=          name of file to log to
--db-api-module=     Name of the db interface implementation module
                    [default: _dagreater]
--help              Display this help and exit.
```

3.2.2 Stopping the Worker

The worker can be stopped using the `stop.py` script. The script reads the information from a specified info file and stops the corresponding worker.

Usage: `python stop.py [info-file]`

The GREAT-ER Desktop Client stops a locally started worker automatically on exit.

A GNU Free Documentation License

Version 1.2, November 2002

Copyright © 2000,2001,2002 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, \LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in

another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section A.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin

distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections A and A above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- State on the Title page the name of the publisher of the Modified Version, as the publisher.
- Preserve all the copyright notices of the Document.
- Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- Include an unaltered copy of this License.
- Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document

as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
- Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
- Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section A above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section A is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate,

or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section A. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section A) to Preserve its Title (section A) will typically require changing the actual title.

TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.