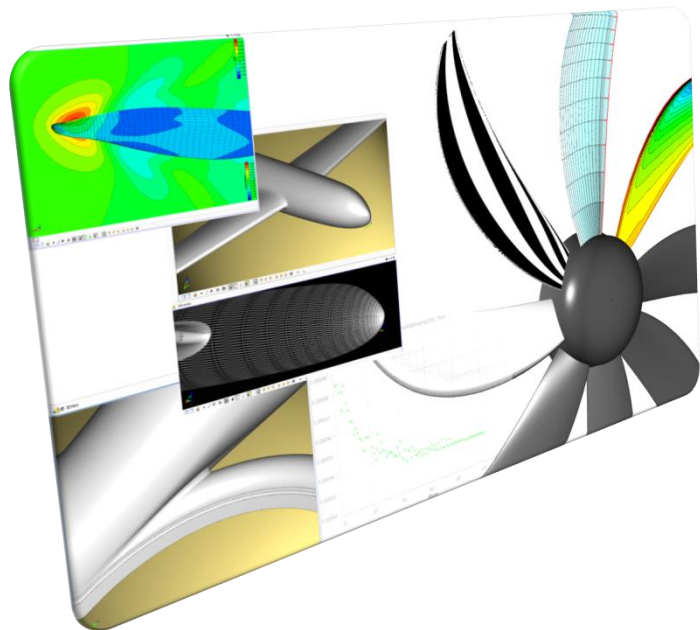




SshResourceManager

A grid engine for CAESES

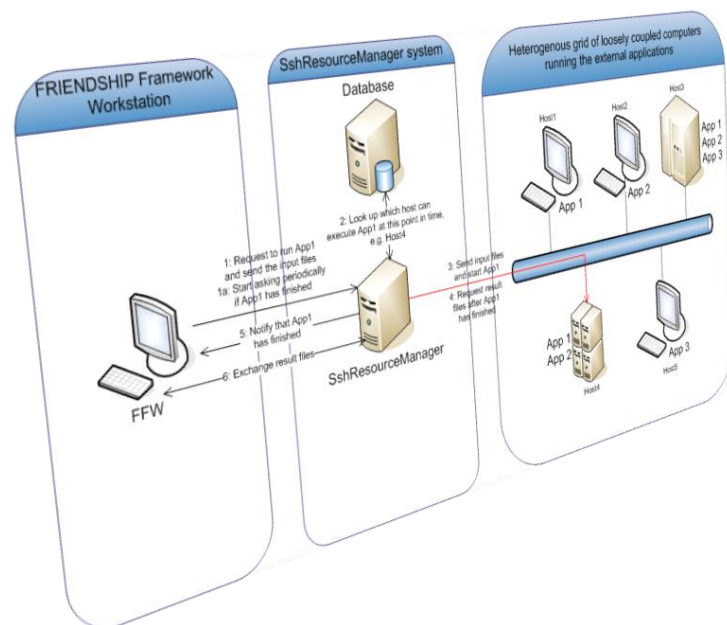
Admin Guide



Part A SshResourceManager	4
A.1 Introduction	5
<i>SshResourceManager</i>	5
About this guide	6
A.2 Installation	7
Head Node	7
Deployment	7
Configuration	7
Recommended additional setup	8
A.3 Configuration	9
Database Settings	9
Security Settings	9
General settings	9
Host management settings	10
Logging settings	11
Mailer settings	11
X forwarding settings	12
Scheduler settings	12
Advanced settings	13
A.4 Additional Tools	15
SshConfigurationManager	15
Administration Interface	15
FLockHost	15
Part B SshConfigurationManager	16
B.1 Introduction	17
B.2 Installation	18
Prerequisites	18
Deployment	18
B.3 Usage	19
Login	19
Operating Systems	19
Applications	20
Floating Licenses	21
Hosts	21
Users	24
State	24
Logout	26
B.4 Administration	27
Update	27
Database	27

General	27
General	27
Logging	28
Mailer	28
B.5 Notes	29
Part C Administration Interface	31

Part A | SshResourceManager



A.1 Introduction

SshResourceManager

This software is a light weight grid engine designed to enable CAESES to start external applications on remote computers based on the SSH protocol. It is a platform independent program that can run on any operating system for which Java is available. It allows to reduce the idle time of software licenses and to utilize available hardware resources fully. Additionally, it enables CAESES to pass operating system boundaries, when, for example, the software you want to integrate has to run on a certain operating system but your CAESES users work on a different operating system.

Prerequisites

There are some requirements to the computers that are part of the setup and the network infrastructure in order to run the SshResourceManager. The computer running the SshResourceManager (the *Head Node*) needs to be running Java 1.6 or higher. It is also necessary for the computers running the CAESES to be able to communicate with the Head Node via TCP/IP.

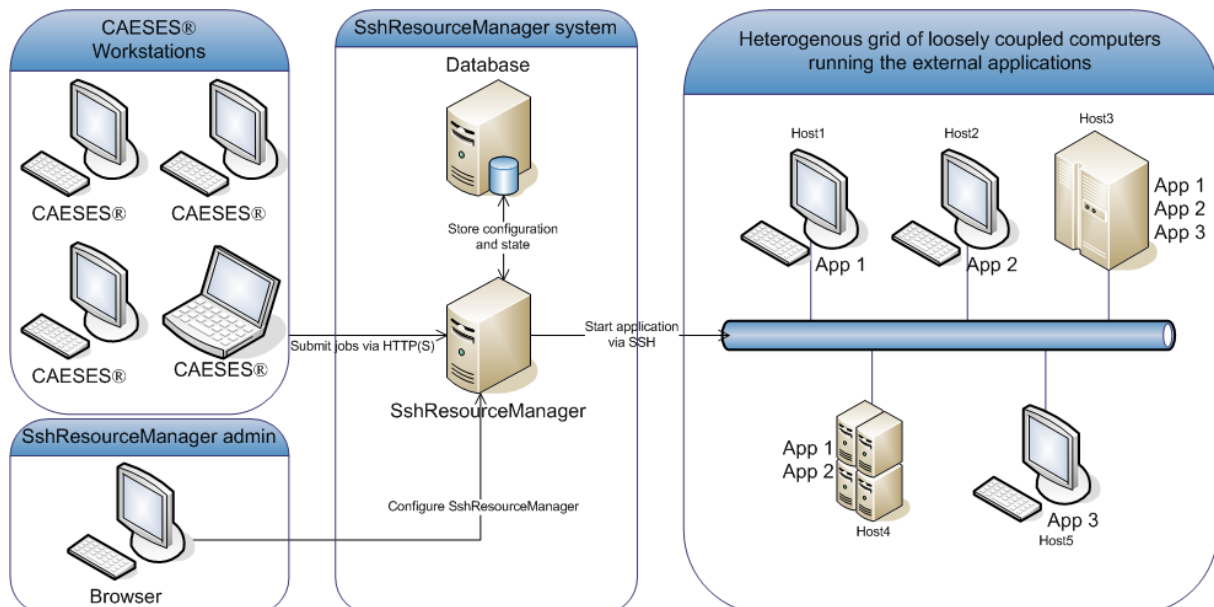
The SshResourceManager stores its configuration in a database, so a database server must be accessible from the Head Node.

The computers running the actual applications (the *Computation Nodes*) need to be accessible from the Head Node via SSH. They need to allow shell and SFTP access using password authentication.

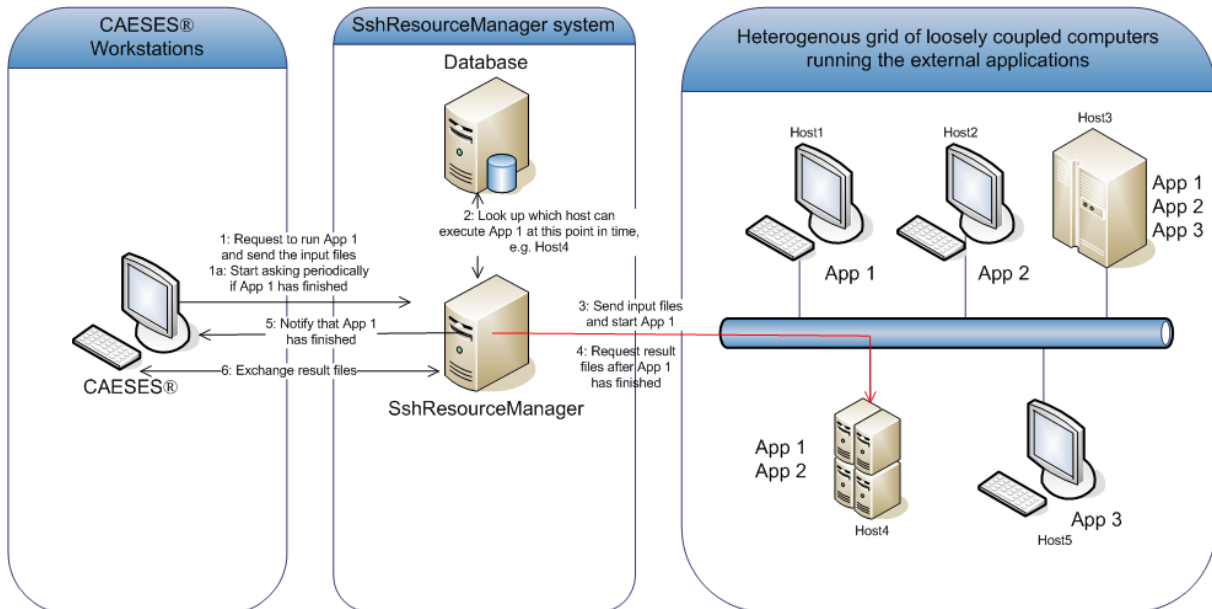
Finally, the workstations that run CAESES and should use the SshResourceManager need to be able to communicate with the computer running the SshResourceManager via TCP/IP.

Setup

A sample setup is depicted in the image below. In this example there are four computers running CAESES that use the SshResourceManager in order to execute three different applications installed on five computation nodes. The workstations need no further information about where to run the actual application, all they do is submit their request to the SshResourceManager which then selects the computation node that is able to fulfill it at that moment in time. The input files are transferred to the selected host before the external program is run. After it has finished the results are copied back to the workstation where they are processed by CAESES. To the user it behaves the same as if the application was executed on the local machine.



The data- and workflow that results from this sample setup when, for example, one of CAESES workstations requests to start the application called “App1” is depicted in the image below.



About this guide

The first part of this guide describes the setup process and the possible settings of the *SshResourceManager*. The configuration process using the web application is detailed in the second part. The usage from within CAESES is documented in the inline documentation of CAESES.

A.2 Installation

Head Node

The machine running the *SshResourceManager* (the *Head Node*) needs a Java™ Virtual Machine version 1.6 and higher installed.

For information regarding the *SshConfigurationManager* which is the web application used to configure the *SshResourceManager* please follow the documentation that can be found in the second part of this guide.

Recommended: For production use it is strongly advised to have a database server running for the *SshResourceManager* to store its configuration. See [Recommended additional setup](#) below for an explanation and the required set up steps.

Deployment

Using the built-in server

Extract the zip file <CAESES installation directory>/tools/sshresourcemanager.zip to a directory on the Head Node.

Change in that directory and issue the command

```
$>java -jar sshresourcemanager.jar
```

This will start the built-in server of the *SshResourceManager* to listen on port 8080. You can change the port by supplying it as the (last) command line argument, e.g.

```
$>java -jar sshresourcemanager.jar 9090
```

starts the server on port 9090.

The *SshResourceManager* understands some additional command line arguments that mainly influence the server behavior:

- -h : Displays the command line option help (alias: help)
- -bip <address>: Tells the *SshResourceManager* to bind to the given address only (alias: bindip)
- -nr : If given, processors and licenses that are marked as being in use in the database are not reset. This may be a problem if the *SshResourceManager* was previously shut down while jobs were running (alias: noreset)
- -nd : By default, the *SshResourceManager* uses the UDP multi-cast address 239.255.43.21 to provide a discovery service for CAESES users. It will try to bind to one of the following

UDP ports on start (in the given order): 49152, 52128, 55123, 61254. If this option is used, the discovery service is disabled. (alias: nodiscovery)

- -sd <description> : If the discovery service is enabled, the provided description will be provided to the CAESES user in order to be able to distinguish between multiple *SshResourceManagers*. Use quotes, if spaces are part of the description. (alias: servicedescriptor)
- -sdbg : If the discovery service is enabled, this will enable additional output when service discovery requests are received. (alias: servicedbg)

Using a servlet server (obsolete)

Starting with version 3.0 the *SshResourceManager* comes with a built-in Tomcat 7 servlet server. It is no longer necessary to use an external servlet server.

Configuration

The configuration of the *SshResourceManager* is stored in the *SshResourceManager.properties* file which is included in the zip package. It needs to reside in the base directory of your *SshResourceManager* installation. Please note that the configuration file contains some sensitive data, e.g. the database password. Make sure to set the file permissions correctly so only authorized users can access it.

The configuration file consists of key-value pairs in the form of "key=value". Lines starting with the '#' character are ignored.

In general, the configuration file does not need to be changed to run the *SshResourceManager* as all important settings can be configured from with the administration web interface. However, in order to restrict the access to the administration interface, please open the *SshResourceManager.properties* file in a text editor and adjust the settings **administratorUser** and **administratorPassword**. The login can also be changed later inside the administration web interface.

The following sections explain how to do the configuration using the configuration file directly. For a description of the administration interface, please go the [third part](#) of this guide.

Recommended additional setup

To use the *SshResourceManager* for production use, it is strongly advised to perform an additional setup. By default the *SshResourceManager* uses a file based SQLite database. It is recommended not to use that setup for production use as there are issues with concurrent access to the database file. The following steps are necessary to configure the *SshResourceManager* to work with the database of your choice:

1. Acquire a jdbc (Java database connector) driver for your database. Jdbc drivers are available for most common database systems.
2. Put the Jar-file containing the jdbc driver in the "dbdriver" sub-directory of your *SshResourceManager* installation.
3. Open the *SshResourceManager.properties* file in a text editor.
4. Adjust the database settings to match your database setup:
databaseJdbcDriver: This should be set to the JDBC class for your database system. (Should be stated in the jdbc driver documentation).
databaseDialect: This is the SQL dialect understood by your database system. A list of supported dialects can be found here: <http://docs.jboss.org/hibernate/orm/3.5/api/org/hibernate/dialect/package-summary.html>
databaseConnectionUrl: This is the Url to the database. Use the format: jdbc:[databasesystem]://[databaseserver]/[databaseName].
 For example when running a PostgreSQL database with the name *fsshrm* on the server "myServer" the url would be:
 jdbc:postgresql://myServer/fsshrm
databaseUsername: This is the username to access the database with.
databasePassword: This is the password for the user.

Your setup should be complete now and you can start the *SshResourceManager* as described before. To test the database configuration, log in to the *SshResourceManager* using the provided web application.

Notes

The most common database systems used with the *SshResourceManager* are *MySQL* and *PostgreSQL*. The correct values for the settings described above for these two database systems are:

MySQL:

databaseJdbcDriver=com.mysql.jdbc.Driver

databaseDialect=org.hibernate.dialect.MySQLDialect

databaseConnectionUrl=jdbc:mysql://localhost/<dbName>
 (assuming the database server is running on the same machine as the *SshResourceManager*)

PostgreSQL:

databaseJdbcDriver=org.postgresql.Driver

databaseDialect=org.hibernate.dialect.PostgreSQLDialect

databaseConnectionUrl=jdbc:postgresql://localhost/<dbName>
 (assuming the database server is running on the same machine as the *SshResourceManager*)

The Jdbc drivers for those two database systems can be downloaded here:

MySQL: <http://dev.mysql.com/downloads/connector/j/>
 (extract the .jar file from the archive after downloading)

PostgreSQL: <http://jdbc.postgresql.org/download.html>

A.3 Configuration

The mentioned configuration file *SshResourceManager.properties* includes some additional settings that can be adjusted to suit your needs. This section describes the possible settings in detail. Note that manual changes to the configuration file are not applied until the *SshResourceManager* is restarted. Most of these settings can also be changed using the [administration interface](#) at runtime.

Database Settings

Please refer to the section [Recommended additional setup](#) for an explanation of the settings in this section of the configuration file.

Security Settings

The *SshResourceManager* uses 1024 Bit RSA encryption to store passwords that are configured. This includes the passwords of the *SshResourceManager* users and passwords for SSH credentials. On first start the *SshResourceManager* generates the required RSA public/private key pair. You can configure the location where these keys are stored as well as whether the private key should be protected with a passphrase.

The keys can be stored as binary files on the hard disc or in the database that is used by the *SshResourceManager*. If passphrase protection is turned on, you have to supply that passphrase in the properties file. Note that there is no way to retrieve that passphrase if you forget it!

Options

rsaKeyStore – Obsolete. This setting is no longer used. The RSA keys are always stored in the database.

rsaPubKeyFile – Obsolete (see above).

rsaPrivKeyFile – Obsolete (see above).

rsaUsePassPhrase – This setting determines, whether the private key should be protected with a pass phrase. Please note that, unlike prior versions of the *SshResourceManager*, versions 3.0 and up require you to set the *rsaPassPhrase* property (see below) if using this feature. Possible values are:

- **true** Passphrase protection is turned on
- **false** Passphrase protection is turned off

rsaPassPhrase – If passphrase protection is on, this property MUST be used to configure the passphrase.

rsaCheckForUnencryptedPasswords – In version 1.x of the *SshResourceManager* passwords were stored unencrypted. After migration to a newer version these passwords will remain in the database as plaintext. By setting this property to true, the *SshResourceManager* is advised to scan the database for those plaintext passwords on startup and encrypt them. Once the *SshResourceManager* was started with this option after the migration process it can be removed from the configuration file. There is no need to use this option if no migration from a pre 2.x version was done.

enableAdministration – The settings in the properties file can also be changed at runtime through the web interface. To enable this feature, set this property to true.

administratorUser – When the *enableAdministration* setting is activated, this property determines the user name of the administrator.

administratorPassword – When the *enableAdministration* setting is activated, this property determines the password of the administrator.

General settings

This section allows you to configure basic settings that are used by the *SshResourceManager* like paths and memory usage.

Options

jobDataDir – This setting lets you choose the path where the *SshResourceManager* stores temporary job data like input and output files.

knownHostsFile – The *SshResourceManager* checks the identity of remote hosts using their fingerprint when establishing an SSH connection. This setting lets you configure the path to the file that is used to store the fingerprints. If the file cannot be created the known hosts feature will be disabled. This may pose a security risk as the identity of the hosts is not verified.

synchronizeKnownHostsOnStartup – If this setting is set to “true” the *SshResourceManager* will synchronize the information stored inside the known-hosts file with the information stored in the database. This means that for all hosts with the known-hosts flag set, it will be checked whether there is an according entry in the known-hosts file and vice-versa. This is especially helpful when sharing the

known-hosts file with other applications (e.g. when a user's `.ssh/known_hosts` file is used).

verifyKnownHostsOnStartup – The *SshResourceManager* can verify the entries stored in the known hosts file on startup. To do so it fetches the fingerprint for all active known hosts during the start procedure. If the stored fingerprint does not match the host's current fingerprint the entry will be removed from the known hosts file and the host will be marked as being no known host.

disableKnownHostsCheck – By setting this property to “true” the fingerprint mechanism can be disabled completely. Since this poses a potential security risk this is not advised!

inputFileServerPort – The *SshResourceManager* uses additional network connections to transfer input and output files. To do so, additional ports need to be opened. This property allows to configure the port that is opened to allow CAESES to connect for sending the input files for a program. The specified port needs to be in the range of 0 and 65535, where a setting of 0 means to use a random port.

resultFileServerPort – This is the port that is opened for sending result files of the applications. Just like the previous setting it needs to be in the range of 0 and 65535, where a setting of 0 means to use a random port.

fileTransferTimeout – During the transfer of input and output files connection problems can be detected through timeouts. This property allows to configure this timeout interval. If the *SshResourceManager* does not receive any information for the amount of seconds configured here, the communication is considered to have failed.

outputBufferSize – This determines the size in MB of data that is transferred to CAESES at once when sending result files. A higher value will result in better performance but will also increase the memory footprint of the *SshResourceManager*. Setting it 0 will always allocate the buffer in the size of the current file. The maximum value for this property is 2096MB.

maximumNumberOfJobs – This determines the maximum number of jobs to be run simultaneous. This setting lets you control the resources used by the *SshResourceManager*. A setting of “0” means that an unlimited number of jobs can run. Note that each job spawns a thread and each thread opens a file on the Head Node to log the application's console output. Depending on the operating system settings the maximum number of open files may be limited which may cause errors if too many jobs are started simultaneously.

deleteTemporaryRemoteFiles – This determines whether files are deleted from the remote machines after the computation is done. You should leave this set to *true* except for diagnosis purposes in case of problems.

timeZone – This determines which time zone is used when displaying dates especially for logging output. The time zone is identified by its Id (e.g. GMT, UTC, PST, America/Los_Angeles). To use the local time zone of the machine running the *SshResourceManager*, set this property to “local”.

configurationInterfaceAddress – Obsolete. The configuration interface (*SshConfigurationManager*) is available at the same address as the *SshResourceManager* itself.

Host management settings

The *SshResourceManager* includes functionality regarding the management of the configured hosts, including automatic monitoring of hosts, static configuration tests, and runtime error checks. These functions can be configured in this section.

testExecutable - The *SshResourceManager* can perform a test run on all configured hosts in order to check whether all host configurations are correct and whether login, file transfer and command execution work on all configured hosts. By default the program that is executed on the hosts as a test is `echo`. If a different program should be executed this property can be set to the path of the executable. Note that the executable path entered here needs to be present on all computation hosts otherwise the test run will return false positives. Test runs can be triggered through the [SshConfigurationManager](#). They are only possible when no other job is running on the *SshResourceManager* and no job is queued. Also, the *SshResourceManager* will not accept new jobs while in test mode.

testArguments - This property holds the argument string that is passed to the program configured in the *testExecutable* property.

disableFailedHosts - When a host is selected as the executor for a job and the connection cannot be established, the *SshResourceManager* can automatically disable that host, so it will not be selected by the scheduler anymore until that problem is resolved. The occurring problems may be that the host is not reachable in the network, the host's fingerprint does not match the stored fingerprint, or login with the given credentials fails. If the automatic availability check is enabled, the host will be re-activated automatically once the problem is fixed. Possible values for this property are:

- **unreachable** - A host will only be disabled if no network connection can be established to the configured SSH port.
- **fingerprint** - A host will only be disabled when the fingerprint does not match the stored fingerprint.
- **credentials** - A host will only be disabled if login fails with the given login data (only applied if global authentication settings are configured for the host - see [global authentication settings](#)).
- **all** - A host will be disabled if any of the above three settings would cause a host to be deactivated.
- **none** - Hosts will not be disabled automatically.

Note that you can also combine these values (except for all and none) by separating them with a dot ('.'), e.g. the value `unreachable.fingerprint` will disable hosts if no network connection can be established or if there is a mismatch of fingerprints.

rescheduleJobsWithFailedHosts - If a job could not be started due to connection problems the *SshResourceManager* can reschedule the job instead of just failing it. It will then be executed another time. Note that depending on the value of *disableFailedHosts* the same host could be selected the next time the job is scheduled. Additionally, setting this property to true may cause jobs to be queued forever if all hosts that could execute it have been disabled automatically.

activateAutomaticAvailabilityCheck - The *SshResourceManager* can periodically check whether all configured hosts which are potentially ready for executing jobs (i.e. they are active and have a confirmed fingerprint) are currently available in the network. In order to activate that function set this property to true.

automaticAvailabilityCheckInterval - This is the interval in seconds in which the hosts are checked for availability if *activateAutomaticAvailabilityCheck* is set to true.

automaticAvailabilityFingerprintCheck - By default the automatic availability check checks the fingerprint to match the stored fingerprint for a host. This may take some time. If you want to deactivate the fingerprint check, set this property to false. Note, that some SSH servers are configured to only send the fingerprint, when valid login data is provided. In that case this fingerprint check can only work when [global authentication settings](#) are configured for all hosts.

disableNotificationAddress - In case any of the automatic disable mechanisms has been applied to a host, the *SshResourceManager* can send a notification email to e.g. an administrator. This property holds the

email address of the the person to notify. Note that this setting will only have effect if the Mailer function has been setup correctly (see section [Mailer settings](#)).

Logging settings

The *SshResourceManager* includes a verbose logging mechanism. It produces logs on the console and as logfiles. This section shows how to configure the logging behavior of the *SshResourceManager*

consoleLogging - This setting determines whether logging output is printed to the console. If you set this setting to *false* the *SshResourceManager* will only produce prompts when it needs to ask for passwords (for the RSA keys or the mailer function if necessary). Note that you will have no indication at what time the *SshResourceManager* is actually running. It is recommended to leave this setting set to *true*.

consoleLoggingLevel - By default the *SshResourceManager* creates very verbose logging information. This is very helpful for the setup and testing phase of the *SshResourceManager*. Once this phase is over, it may be desired to reduce the verbosity and to only log information of higher priority levels. This setting is the verbosity level of console output. Possible values are:

- **debug** - The highest verbosity. All information is logged.
- **info** - Medium verbosity, output of level *INFO* and higher is logged.
- **warn** - Low verbosity, only messages of the levels *WARN* and *ERROR* are logged.
- **error** - Lowest verbosity, only output of level *ERROR* is logged.

maxLogFileSize - The *SshResourceManager* creates logfiles in HTML format. This setting determines how large a logfile can become before it is overwritten (in KB).

maxLogFileBackups - The logfiles are created as "rolling logfiles". This means that when the size limit is reached a backup of the current logfile will be created before a new one is started. This setting configures how many backups are kept before existing backups are overwritten.

fileLoggingLevel - This is the level of output that is written to the logfiles. The levels are the same as described for the *consoleLoggingLevel* property.

Mailer settings

The *SshResourceManager* is capable of sending notification emails to the user that submitted a job once the job is finished. In order to use this feature, you need to provide information about a SMTP server the

SshResourceManager can use to send the mails. Additionally, only those users can be notified by that have an email address configured.

smtpEnable - This setting enables and disables the mailer functionality. The mailer will only be active if this setting is set to "true".

smtpHost - This is the address of the SMTP server the *SshResourceManager* uses to send emails.

smtpPort - This is the port the SMTP server is listening at.

smtpUser - In case the SMTP server requires authentication using a user name and password, this is the user name used to log in to the SMTP server. If this setting is empty or not present, no authentication will be used.

smtpPass - If the SMTP server requires authentication, this setting should hold the password.

smtpFromAddress - This is the address that will be used as the "from" address for emails sent by the *SshResourceManager*.

smtpDefaultToAddress - Optionally, you can enter an address where the *SshResourceManager* sends notifications in case the user has no email configured.

smtpTestMail - If this setting is set to *true*, the *SshResourceManager* will send a confirmation email to the address configured in the setting "*smtpFromAddress*" when it starts up. This is to check whether the mailer function is set up correctly and sending emails works.

smtpDebug - If you experience problems with sending email, this setting can be set to *true* to help diagnosis. Note that debug information of the mailer will only be printed on the console and will not be present in the *SshResourceManager*'s logfiles.

X forwarding settings

Some applications may need an X server to work. In order to use such an application, the host that executes the application needs to forward the X server to a computer running an X server. The *SshResourceManager* has two ways to set up the X server that is forwarded to. The first one is to set up a default X server where all requests for X server forwarding are sent. This is done using the properties file. As an alternative, you can configure the hosts to have X server forwarding support. In that case you can also configure multiple X server forwarding ports for each host. You can also create hosts for the sole purpose of X forwarding.

Both ways can be configured in parallel. By default an application that needs X server support will forward the X to the default server configured here. That behavior can be changed through the *SshConfigurationManager*.

The following settings configure the X default server which is forwarded to.

XServerHostName - This is the host name of the computer where the X server is forwarded to.

XServerPort - This is the port the X server is running on.

Scheduler settings

The scheduler is the *SshResourceManager*'s module that selects the next job to run and on which host it will run. Hints can be passed to the scheduler how it should prioritize during this process.

schedulerHint - This is the hint for the scheduler how to prioritize job and host selection. Since you can replace the built-in scheduler with your own implementation, this setting may be arbitrary strings that the scheduler implementation can then process. The built-in scheduler can handle the following hints:

- **cpu** - The scheduler will always select the host with the highest number of free CPUs.
- **license** - This setting will cause the scheduler to first use up all node locked licenses (*hard licenses*) before using any float licenses. It will also try to minimize the fragmentation of the node locked licenses, i.e. it will select the host with the smallest difference between needed licenses and available licenses. This is the default setting.
- **priority** - This means that the scheduler will select the host based on the priority that was configured for the current application on a host. For hosts with the same priority the second criterion is licenses (see above).
- **priority.cpu** - If the second criterion should be CPUs instead of licenses, this setting can be used.

schedulerFairnessCriterion - The scheduler selects jobs based on a fairness algorithm. This means that when different users submit jobs to the *SshResourceManager* the scheduler will alternate between the jobs submitted by different users so no single user will block other users' jobs for a potentially very long time. This property determines whether that selection is based on the user's login name into the *SshResourceManager* or based on the user's actual username on the computer (the workstation running CAESES). Possible values for this property are:

- **owner** – Use the username used to log in to the computer.
- **submitter** – Use the username used to log in to the *SshResourceManager*

schedulerDebug - This setting enables the debugging output of the scheduler.

See also the section [Advanced settings](#) on how to replace the scheduler with your own implementation.

Advanced settings

The *SshResourceManager* provides interfaces that lets you extend or alter its functionality and/or behavior.

The *SshResourceManager* can dynamically load and even compile java classes on startup and during runtime. This makes it possible for you to implement special functionality that suits your needs and setup. The following settings configure that functionality. Javadoc for the interfaces that need to be implemented can be found in the javadocs subdirectory of the ResourceManager's documentation directory. All Java source code that is provided can be found in the same directory.

Note that in order to make use of any of the features that compile Java source code at startup and/or runtime, you will need a Java Development Kit (JDK) installed. Also, the *SshResourceManager* needs to run from the Java Runtime Environment that is part of the JDK.

Custom Scheduler

scheduler – The built-in scheduler can be replaced with your own implementation. Your custom scheduler class needs to implement the

`com.friendshipsystems.fsshresourcemanager.FIScheduler` interface. That interface is provided in the *javadocs* subfolder of the documentation folder. The class is looked for in the classpath. If it fails to load a warning will be displayed on start-up and the built-in scheduler will be used.

Alternatively, the *SshResourceManager* can compile the scheduler provided as Java source code on start-up. The next to properties configure that functionality.

schedulerToCompile - This is the path to the Java source code file that is supposed to be compiled.

schedulerToCompileClass - This is the fully qualified classname of the class defined in the source file that is provided in the *schedulerToCompile* setting.

Custom Job Canceler

Canceling SSH jobs is not straight-forward as the SSH protocol does not provide a mechanism to do so. In order to provide the possibility to quit running jobs, the *SshResourceManager* executes several commands on the remote computer through a second SSH connection. These commands have the goal to find out the process ID of the remote process and to issue the termination command when a process should be killed.

The *SshResourceManager* is shipped with classes that should work on most Windows hosts and on Linux hosts running a bash shell. If the computation nodes run different operating system custom classes to fulfill the canceling task need to be implemented. To simplify that process we provide the implementations of the two built-in classes as a starting point for the custom implementations. These custom implementations need to implement the

`com.friendshipsystems.fsshresourcemanager.FIJobCanceler`

interface which is also provided in the *javadocs* folder.

Similar to the custom scheduler class the *SshResourceManager* can either handle precompiled Java classes or compile them from Java source code on start-up. New Job Canceler classes can also be implemented from the configuration web application.

Loading custom Job Cancelers

The setting(s) to load such a canceler class is (are) **cancelerToLoad[number]** where [number] is consecutive number starting with 1. Again the classes are tried to load from the classpath. If you want to load 3 additional canceler classes, for example, the corresponding part of your property file would look like this:

cancelerToLoad1=some.package.CustomCanceler

cancelerToLoad2=some.package.AnotherCustomCanceler

cancelerToLoad3=also.some.other.package.AThirdCustomCanceler

Compiling Job Canceler class on start-up

Similar to the scheduler you can provide Java source file(s) and the corresponding class names to the *SshResourceManager* to compile on start-up. It follows the same mechanism as loading canceler classes, as the setting(s) need to be followed by a consecutive number starting with 1. These settings are:

cancelerToCompile[number]

cancelerToCompileClass[number]

Example:

cancelerToCompile1=C:/myjavasource/src/some/package/CustomCanceler.java

cancelerToCompileClass1=some.package.CustomCanceler

cancelerToCompile2=C:/myjavasource/src/some/package/AnotherCustomCanceler.java

cancelerToCompileClass2=some.package.AnotherCustomCanceler

Compiling canceler classes at runtime

The webinterface provides you with the possibility to enter Java source code, compile it and assign it to an operating system. The source code will be stored in your database and will be reloaded the next time you run the *SshResourceManager*

A.4 Additional Tools

SshConfigurationManager

This is the web application that is used to configure the *SshResourceManager*. Here you need to set up the hosts, applications and licenses that are managed and can be then used by CAESES users. It is included in the *SshResourceManager* and is available at the same address the *SshResourceManager* was started at. The webinterface can then be used from any computer that can communicate with that webserver using a web browser that supports Javascript. We recommend using Firefox, Opera or Chrome. The *SshConfigurationManager* is described in detail in the second part of this guide.

Administration Interface

Most settings that were described in the previous sections can also be configured at runtime using the administration interface. The third part of this guide explains how to use that part of the *SshResourceManager*'s web interface.

FLockHost

This application can be used to lock hosts and resources from workstations.

If you have configured a host that is also used as workstation, the user of that computer can use this application to set CPUs as being used in the *SshResourceManager*.

Part B | SshConfigurationManager



B.1 Introduction

The *SshConfigurationManager* is the web application that configures the *SshResourceManager*. In order for the *SshResourceManager* to start applications on remote computers, it needs to know about the topology of the grid it works on. This information includes the remote hosts, the operating systems that run on those hosts, and the applications that are installed on the hosts. Additionally, the *SshResourceManager* includes a user management to limit the access to its functionality.

The web application is included in the *SshResourceManager*. As opposed to previous versions an additional HTTP server is no longer needed. Any computer that has access to the *SshResourceManager* can be used to access the web application through a browser.

This part of the admin guide describes the process of setting up the web application as well as its usage. It is assumed that the *SshResourceManager* is already set-up and running.

B.2 Installation

Prerequisites

The *SshConfigurationManager* is part of the *SshResourceManager*. As opposed to previous versions an external webserver is no longer needed. It is available as soon as the *SshResourceManager* is started.

Once the *SshResourceManager* is set up, a browser is needed to access the application. The browser needs to support Javascript and cookies enabled for the application to work correctly. It has been tested with all current browsers. Only the Internet Explorer has shown major problems when interacting with the application. We are currently working on a solution for these problems and advise you to use a different browser until the problems are solved. Please see the following table for browser compatibility.

Browser	Version	Caveats
Firefox	3.6.3	-
Firefox	3.0.3	-
Opera	10.53	Sorting indication cursor not working correctly
Chrome	5.0	Tab navigation within a page not working correctly, State auto refresh does not work
Safari	4.0.5	Tab navigation within a page not working correctly, State auto refresh does not work
Internet Explorer	8.0	Multiple display and functionality errors. Use currently not recommended.

Deployment

Once the *SshResourceManager* is set up and running, no additional steps are required to use the *SshConfigurationManager*. Open a web-browser and enter the address (including the port) where the *SshResourceManager* is running.

B.3 Usage

In order to use the *SshConfigurationManager*, open a browser of your choice (see the [browser support comparison](#)) and navigate to the application. For example:

`http://yourSshResourceManager:8080`

The key components of the *SshResourceManager* that need to be configured so it can do its work are the applications it can start on remote computers and the remote computers themselves so it knows where to start which application. Those computers are called "hosts".

All configuration pages are set up in a similar general structure. On the top you will find a list of the current configuration objects of the particular kind, where you can change the existing objects. On the bottom you will have the chance to create new configuration objects. Mostly, you can change the same properties for existing and newly created objects. However, there are some cases where certain attributes can only be changed after creating the object and others are not changeable anymore after first creation.

Login

On the front page you will be presented a login screen. Initially you can use the login "admin" with an arbitrary password. Once you have created a user with admin privileges this account will be disabled.

Additionally, the front page gives users the possibility to change their password.

All users have the possibility to log into the *SshConfigurationManager*. However, only users with the SuperUser level (see section [Users](#)) are able to edit the configuration of the *SshResourceManager*. Normal users have the possibility to

change their user settings (see section [Users](#)) and to view the current state of the *SshResourceManager* (see section [State](#)).

After logging in you will be presented the main menu of the *SshConfigurationManager* from where you can access all configuration pages which will be explained in detail in the following sections.

Operating Systems

Some operations that need to be executed on hosts in order to start applications on them are dependent on the operating system each host runs. Therefore, each host that you configure needs to have an operating system assigned. This page allows setting up those operating systems.

On first start-up the *SshResourceManager* will create and preconfigure the two most common operating systems, namely Linux and Windows. This section explains the settings that need to be configured for each operating system using the two default ones as examples.

Name

The name is a mere identifier to simplify the identification of an operating system when assigning it to a host. The two default operating system use "Linux" and "Windows", but can be changed if you desire.

Parent

Operating systems can be set up in a hierarchical structure. An operating system can have a parent operating system whose settings apply to the child as well. For example, having the Linux operating system in place, you could create two children called "Linux32" and "Linux64" representing 32 and 64 bit versions of Linux. When creating those children the settings of the parent will automatically be inherited by each child, simplifying the setup process.

Change dir command

This is the command that needs to be executed on a computer running this operating system in order to change directories within a path. Although this is the same for most current operating systems when working with the console on the computer directly (the "cd" command) the internal execution may differ. The *SshResourceManager* does need the actual internal execution as it would be performed by the operating system. For the pre-configured operating systems Linux and Windows the command is "cd".

If you want to configure other operating systems than Linux and Windows, please consult the documentation of the operating system to find out the internal command that is executed to change directories.

Path Delimiter

When displaying paths some operating systems do not use the standard delimiter "/" as it is specified in the RFC for URLs (see RFC 1738). On Windows the path delimiter is the backslash character ("\"), for example. The *SshResourceManager* needs to know which delimiter is used by your operating system.

Canceler class

The SSH protocol does not specify a procedure how to quit remotely executed processes once they are started. Therefore, the *SshResourceManager* uses other mechanisms in order to provide the possibility to stop remote execution. This setting lets you choose the implementation of those mechanisms that best suits your operating system.

The *SshResourceManager* is shipped with two implementations, one that should work on Windows computers and one that should work on Linux computers with a bash shell.

For new operating systems the *SshConfigurationManager* lets you either choose one of those two or let you create a new one. For a more in depth guide how to create your own Canceler Classes please see the section [Custom Job Cancelers](#) in part A of this guide.

Create new JobCanceler

This button allows you to implement a new Canceler class using the webinterface. After pressing that button you will be presented an editor where you can create your new Java class that implements the appropriate interface. Besides the actual code you will also need to provide the name of the class. The editor allows you to compile the written code in the webinterface. Any errors will be presented to you. When implementation is done you can submit your Java code which is then compiled and the newly created class is assigned as the new JobCanceler for this operating system.

After that you can also assign the new class to other operating systems.

Delete

This allows you to delete an operating system. You cannot delete an operating system that is currently assigned to a host.

Show hosts

This button gives you a view of the hosts that are currently running that operating system.

Applications

The applications are the interface between the *SshResourceManager* and *CAESES*. The application you set up here are the ones that will show up for the *CAESES* user when he connects to the *SshResourceManager*. They are also the container that connects multiple hosts with each other and holds the actual executables that will be run on the hosts.

Name

Like for operating systems, this is a mere identifier that allows you and the *CAESES* user to identify a certain application.

Description

Besides the name this can hold additional information that gives further details about an application.

Required licenses

The *SshResourceManager* needs to keep track of the resources that are currently used based on the applications that have been started so far. The required licenses field allows you to set the number of licenses an application needs per running instance.

In general this will be one, however especially some multi-threaded applications may need multiple licenses per running instances.

Required CPUs

Similar to the number of required licenses per instance, this allows you to specify how many CPUs are needed by one instance of an application. This way the *SshResourceManager* can ensure that it will only start an application on a host that has enough processors available to run an application to ensure best performance.

Needs X

Some applications may need an X server in order to run. This setting lets you mark an application to have such a requirement.

If an application is marked as needing X server support, you will be requested to select the X server that will be forwarded to. The *SshResourceManager* can be configured to have a global default X server to forward to and/or you can configure hosts to take the X server forwarding. You can only mark an application to need X support if either the global X server is configured or if at least one host exists that supports X forwarding (see also section [Host X forwarding](#)).

Floating license

Here you can assign a floating license to an application. Assigning a floating license to an application causes the *SshResourceManager* to consider that floating license when trying to determine whether or not there are licenses available for an application to start. Floating licenses can be created either on the floating license page (see section [Floating Licenses](#)) or directly on the application page by pressing the button "Create a new floating license for this application". One floating license can be assigned to multiple applications.

IsCaeses

If an application is labeled as being label, it enables the possibility to pause a Design Engine run on a workstation and send it to a remote computer and have it continue there.

Delete

With this button you can delete an application. After deletion it will also be removed from all hosts that have this application configured.

View hosts

Pressing this button gives you a view of all hosts that have this application configured.

New Floating license

As mentioned [before](#) this button allows you to create a new floating license that is automatically assigned to the application without having to change to the floating licenses page.

Floating Licenses

Most vendors offer floating license models for their applications. Floating licenses have the advantage that they are not bound to a certain execution host, but can be used by multiple computers. The *SshResourceManager* can keep track of the currently used floating licenses that have been checked out by the applications that were started by the *SshResourceManager*.

Note that it does not interact with the actual floating license server. So using a floating license outside of the *SshResourceManager* will make it impossible for the *SshResourceManager* to know the actual number of licenses that are in use which may cause applications to attempt to check out more than the available licenses. We recommend to start and manage all application executions that make use of such a floating license that is configured in the *SshResourceManager* by the *SshResourceManager* through CAESSES.

This does not limit you to execute all of those applications remotely via SSH as the *SshResourceManager* can also be applied to monitor floating licenses used by locally executed programs through CAESSES. This can be achieved by connecting a local application in CAESSES with a remote application of the *SshResourceManager*. CAESSES will then check out floating licenses from the *SshResourceManager* when running the computation locally. More information can also be found in the documentation shipped with CAESSES.

Name

A floating license's name is an identifier that helps you in finding a floating license when assigning it to an application.

Number

This is the number of available licenses for this floating license.

In use

This gives you information about how many slots of this floating license are currently being used. When this value is larger than zero, a button will be shown that allows you to manually reset the number of used licenses to zero.

Note that you should only do this if you are sure that is what you want. There are some occasions which make this needed. If a local application uses a floating license and the computer crashes, the *SshResourceManager* will not receive the signal to free that floating license again. This causes the number of used licenses to increase even though no application is running that is actually using the license. In that case this button needs to be used. Please use this function with caution. You should make sure that there is no other application instance running that uses this license.

Delete

This button deletes a floating license. It will also be removed from any application it is assigned to.

View applications

Pressing this button gives you a view of all applications this floating license is assigned to.

Hosts

The hosts are the core of your grid setup. They represent the computers that are contacted by the *SshResourceManager* via SSH in order to start remote processes. There are two prerequisites for computers in order to be able to use them for this purpose. The first one is that the computer needs to be reachable by the computer running the *SshResourceManager* via a

network. The network topology does not matter, for the *SshResourceManager* it makes no difference whether it is ethernet or internet for example. The second condition that has to be met (which directly requires the first one) is that the computer needs to be accessible through SSH using password authentication. This means that it needs to run an SSH server that supports shell execution and SFTP. While most Linux distributions are shipped with an SSH server by default, other operating systems (e.g. Windows) are not. There are, however, several SSH server solutions available for all common operating systems, both free and commercial. Note that for Windows we have experienced problems when using certain versions of the Cygwin SSH daemon as it has a bug in its implementation that prevents multiple SSH channels being opened on the same session which should be possible according to SSH specifications.

Name

While for the other configuration objects the name is mere identifier to ease the setup process the name for a host is also the computer's hostname, i.e. the name which identifies the computer in the network. This can be the either the name that was given to the computer or the computer's IP address. Note that in dynamic network configurations using DHCP the IP address may change when a computer is rebooted for example.

This setting cannot be changed after a host was added. You will need to create (or clone) a new host and delete the old one if the hostname/IP address changes.

Port

This is port the SSH server on that host is running at. The standard port for SSH is 22.

Number of CPUs

Modern computers often have more than one processor. This setting allows you to set the number of processors available on the particular host. It is needed so the *SshResourceManager* knows how many computational resources are available on a host and how many applications can be run on it at the same time. This relates to the number of CPUs needed by an application as configured in the application's settings.

MB Memory

This is the amount of memory a host has. At the moment this is a pure informational property and has no influence on how the *SshResourceManager* selects a host. It is included for future use during the scheduling process.

Operating System

Here you can select the operating system this host is running. You can select from the operating systems that were configured on the [Operating Systems page](#).

Sftp Root

Input files for applications are copied to the remote hosts using SFTP. So the actual process will be started in a directory that is accessible through SFTP. In order for the *SshResourceManager* to be able to do so, it needs to know which actual path files transferred via SFTP are located in. The Sftp root directory is the directory a user that connects to host via SFTP is in initially. This usually depends on the configuration of the SFTP server (i.e. the SSH server). In some cases the SFTP root depends on the user that connects to the host via SSH. If that is the case it is advisable to configure [global authentication](#) settings for that host. Alternatively you can configure Sftp root to be the system's root directory ("/") and set up the *Temp Dir* (see below) to a directory which is accessible for all users.

Temp dir

The *SshResourceManager* will create a special directory on the remote host when using it to execute applications. This directory will be created relative to the path specified in *Sftp Root* (see above).

Host X forwarding

As mentioned in the [applications section](#), some applications may require X server forwarding to work correctly. By setting this flag, the host will be selectable for an application to forward X to. If activated you can enter a comma separated list of ports the X server can be forwarded to. This allows forwarding multiple applications to the same host while giving each application its own X.

Note that this is independent of the "active" flag of a host, i.e. a host that is set to be an X host and that is entered as the X forwarding host of an application will be used to forward X to, even if the host is marked as being inactive.

Active

This flag determines if a host is currently active. Since the setup of computers may change dynamically, this allows you to take certain hosts out of the list of available computers at any time. When setting a previously active host inactive, however, applications that are already running will continue to run until finished, but no new processes will be started there.

Only those computers set to active and with a [confirmed fingerprint](#) are used to start computations. This in turn means that only applications installed on at least one host

that is active and has a confirmed fingerprint will be available in the list of remote applications for the CAESES user.

Global Authentication settings (Auth)

In the `ResourceManagerSetupSsh` object in CAESES, the user can set an SSH username and password that will be used to log in to remote hosts. As an alternative, which eases the usage for the actual CAESES user, you can set that info for each host individually ("global authentication settings"). This setting overrides any credentials that are entered in CAESES. When the user enters no SSH authentication information in CAESES only those applications that are installed on at least one host with global authentication settings are returned in the list of applications. When setting the global authentication settings for a host, you can optionally tell the `SshResourceManager` to check whether an SSH connection can be established using the given username and password. Passwords that are entered here will be saved in the `SshResourceManager's` configuration database using RSA encryption.

This setting is only available for existing hosts and cannot be configured during the initial adding of a host.

Fingerprint

In order to confirm the authenticity of a host, the `SshResourceManager` stores the hosts' fingerprints. Before a host can be used for execution, you need to confirm the fingerprint once. Then every time an SSH connection to that host is established, the fingerprint is compared to the stored one and the `SshResourceManager` will only continue, if the fingerprints match. This ensures that no confidential data is transferred to unauthorized computers.

This setting is only available for existing hosts and cannot be configured during the initial adding of a host.

Please note that for some SSH servers you will need to configure global authentication settings at least temporary in order to obtain the fingerprint.

Delete

By pressing this button you can delete a host.

Test Configuration

Press this link in order to start a test run on all configured hosts. The `SshConfigurationManager` allows you to enter optional SSH login data before starting the actual run, in order to also test hosts that have no global authentication settings configured. Note that a test run is only possible when the `SshResourceManager` is currently idle, i.e. no

jobs are running and no job is queued. During a test run all jobs sent from CAESES are rejected.

Applications on Hosts

Once a host is configured, you will need to assign applications that are installed on that particular host. Pressing this button will present you with the interface that allows you to manage the applications for a particular host. As on all pages, the top part gives you an overview of the applications currently installed on that host and the bottom part allows you to add new applications to the host. Pressing the "back" link will take you back to the normal hosts view.

Priority

When using priority based scheduling (see the [Scheduler Settings](#) documentation), this allows you to assign the priority for an application to be executed on that particular host. When priorities are used, the `SshResourceManager` will try to use the host with the highest priority for an application to start the application on first and only resort to other hosts if the one with the highest priority does not have enough resources available.

Hard licenses

The `SshResourceManager` knows two kinds of licenses, floating licenses and hard licenses (also known as node locked licenses). Here you can set how many hard licenses for that application are available on the particular host.

Executable path

This is the path of the executable for that application as it is installed on the host.

Remove

Pressing this button will remove the application from the host.

Clone host

In many cases the IT infrastructure consists of many computers that are very similar to each other. To ease the setup process you can take one host as the basis for new hosts so you do not need to enter the same information over and over. This is what the "Clone host" feature is for. When pressing this button, you will be taken to a view similar to the "new host" section of the Host page except that it has all settings preconfigured as they were on the original host. You do need to set the hostname for the new host and you can change the settings that may differ. Note that all applications are cloned as well.

Add clone

The "add clone" link at the top allows you to create multiple identical hosts at a time. For each time you press the link a new host row is entered where you can set the hostname and alter the information necessary. Once done, you can press any of the "add" buttons to create all hosts at once. Only those hosts will be created for which a valid hostname was given. So if you clicked the "add clone" host too many times, you do not need to start over.

Clone applications

When you do not want to clone a complete host but only want to apply the settings you have made for an application on a particular host to other hosts this allows you to do so.

You will be presented with a list of hosts and a list of applications that are installed on the source host. You can then select the destination hosts and the applications to clone. When pressing "perform", the selected applications will be added to the selected hosts with the same settings (priority, hard licenses and path to executable) as on the source host. When an application is already installed on a destination host, the existing configuration will not be overwritten, but the host will be skipped for that application.

Users

The *SshResourceManager* has a user management feature. This ensures that only authorized users are allowed to use the *SshResourceManager* for computations. It also protects the configuration of the *SshResourceManager* from changes by unauthorized persons. There are three levels of users: **LockedUser**, **User** and **SuperUser**.

A **LockedUser** is not allowed to actually use the *SshResourceManager*. It is basically a disabled user account. This allows you to activate and deactivate user accounts in a comfortable way if you need to grant access to the *SshResourceManager* on a temporary basis, for example.

A **User** can use the *SshResourceManager* to start remote computations through CAESES but cannot change the configuration of the *SshResourceManager*.

A **SuperUser** has full access to the webinterface and can, therefore, alter the *SshResourceManager*'s configuration.

All users are able to access the user page when logging in to the webinterface, but users of the levels LockedUser and User are only allowed to change their own settings and cannot change their access level.

Username

This is the name that is used to log in to the *SshResourceManager*, both, using the *SshConfigurationManager* and when submitting requests to start a remote computation through CAESES.

This setting can only be changed when initially adding a new user and cannot be changed later.

Role

This allows you to select the access level of the particular user (see above). This can only be changed by users with the access level *SuperUser*.

Email

The *SshResourceManager* can be configured to send [information emails](#) to the user that started a computation when it is done. If that feature is set up and the particular user wants to make use of it, an email address needs to be supplied.

LDAP only

This option is only available if LDAP authentication is configured in the administration interface. When it is activated for a user, all authentication will be done through LDAP and not through the *SshResourceManager*'s internal database.

Password

Either when pressing the button or when adding a new user, you can set a password for that user. This needs to be supplied when logging in to the *SshResourceManager* either through the *SshConfigurationManager* or when using it with CAESES. Each user can change his password at any time. Note that passwords of users that are set to *LDAP only* cannot be changed from within the *SshResourceManager*.

Delete

This deletes a user. This is only available for users with the access level *SuperUser*.

State

The state page gives you a quick overview of the configuration of the *SshResourceManager* and the current state of computations running.

Clean up

Pressing this link will reset all resources to the state unused. I.e. it will reset all licenses and processors of all hosts to a used count of zero. This should only be necessary if an error occurred, e.g. the Computer running the *SshResourceManager* was terminated in an

uncontrolled way. You should only use this if you are really sure that you need it. This action needs confirmation and is, of course, only available to *SuperUsers*.

Refresh

This refreshes the state page. This is basically the same as pressing the "reload" button in the browser.

Automatic refresh

This button enables an automatic update. When activated you can enter the rate in seconds how often the state page should be refreshed.

Summary

This gives a very brief overview of the current configuration.

Scheduler State

This shows the current state of the scheduler. At certain times the *SshResourceManager* blocks the scheduling of new jobs. This is the case, for example, when a DesignEngine is being canceled to avoid starting jobs that will be canceled in the next second. The state of the scheduler blocked should only exist for a short time period. However, if during that time an error occurs (e.g. network failure), the scheduler may not be unlocked. You can then use this setting to unblock it again.

On the other hand, you can also use this to block scheduling manually for a certain period of time.

Jobs

This section gives you information about the jobs (computations) that are currently managed by the *SshResourceManager*. It is separated in three parts: *Running Jobs*, *Waiting Jobs* and *Finished Jobs*. When clicking on a particular job, you can view the address of the computer that is running CAESES that submitted the job, the name of the project that the ExternComputation this job belongs to is part of, and the name of the design in which this ExternComputation is executed.

Running Jobs

These are the computations that are currently being run on a remote host. The JobId is the GUID (global unique identifier) of the job as it was assigned by CAESES.

The Task shows how many tasks are part of this job and which one is currently executed. A task can be copying of input/output files and the actual execution of a process.

Submitter is the user of the *SshResourceManager* that logged in to start this job.

The Owner is the actual computer user (i.e. the user that was logged in to the computer that is running CAESES) which submitted the job.

The Submit Time tells you the time at which the *SshResourceManager* received the request to run the job, which is basically the same time CAESES sent the job plus the time that was needed to transfer the request over the network.

The Application is the name of the application that is or will be executed by the job.

The Execution Host is the host that actually runs the remote process.

The State is the state of the current task. For running jobs this should always be the state "Running".

Waiting Jobs

When a job is submitted to the *SshResourceManager* and there currently no resources available to run it, the job will be queued. Once the resources become available again, waiting jobs will be selected to run. This gives an overview of the current waiting queue. The information given is basically the same as for a running job. However, since it is not yet running it was not assigned to an actual host to execute it and therefore there is no information about the "Execution host". As additional information for scheduled jobs you can see whether the job needs a host that has [global authentication](#) settings as there were no such settings set in CAESES.

Finished Jobs

Jobs that have completed but were not yet acknowledged by the corresponding CAESES instance will be displayed here. A job will be removed from this list once CAESES has confirmed that it successfully received the output files of the job.

Hosts

Here you can see an overview of the hosts that are configured in the *SshResourceManager*.

It gives information about configuration settings and the current CPU usage. By clicking on one of the hosts you can view information about the applications installed on that host and the current state of hard licenses if available.

Applications

This is an overview of the applications that are configured in the *SshResourceManager*.

You can see on how many hosts each application is installed and how many of them are currently active. You will also see the available licenses for this application, both floating and hard licenses, and how many of them

are currently in use. Finally you see how many instances of the application are currently running and how many can be executed at the same time based on the licenses and CPUs available on the hosts this application is installed on.

Logout

This ends the current session and will leave the configuration interface.

B.4 Administration

If the *SshResourceManager* was configured to allow remote administration (see the [enableAdministration](#) property) most settings stored in the properties file of the *SshResourceManager* can be configured here. Since there is only one administration account which is not related to the configured users, an additional login is required. Note that all changes to the configuration are only possible while the *SshResourceManager* is in idle state, i.e. no jobs are running, waiting, or finished.

Update

When a new version of the *SshResourceManager* is available an update can be performed at runtime. This page shows you the current versions of the three components the *SshResourceManager* system consists of:

- The *SshResourceManager* itself is the component that manages the jobs
- The server is the inbuilt Tomcat server
- The administration interface

The zip file containing the new release of the *SshResourceManager* system can be either uploaded through the browser by pressing the *Upload zip file* button, or, if it already is located on the computer running the *SshResourceManager*, it can be specified by pressing the *Browse* button.

After uploading or selecting a file, it will be analyzed for the versions contained in it and they will be displayed. If one of the three components can be updated the buttons to start the update will be enabled (depending on which components can be updated).

The button *Update SshResourceManager* will update the actual *SshResourceManager* component only. The button *Update all* also updates the server and the administration interface. This will take a little bit longer than updating the *SshResourceManager* only. If the checkbox *Dry run* is checked the update will only be simulated and no files will be changed.

This page also allows you to restart the *SshResourceManager* component. This may be desired if manual changes to the properties file were made which are not applied until before the *SshResourceManager* is restarted.

Database

This page of the administration interface allows to change the database configuration of the *SshResourceManager*. The available settings are the same as in the configuration file. If the new database was previously used with the *SshResourceManager* and contains RSA keys that are protected with a different passphrase than the one provided in the properties file, changing to that database will fail.

All changes on this page will not be saved on the *SshResourceManager* until the *Submit* button was pressed.

General

This page of the administration interface allows to change the general settings of the *SshResourceManager* this includes

- The directory where temporary data is stored
- The ports of the input and output file servers
- The timeout for file transfers
- The buffer size for transferring result files
- The maximum number of simultaneously running jobs
- Whether to delete temporary files from remote computers
- The time zone for log messages

All inputs on this page are saved right away.

General

This page allows to configure the automatic host management settings. The settings are the same as in the configuration file:

- Whether to synchronize the known hosts file and the database on start-up
- Whether to verify the stored fingerprints of known hosts on start-up
- The executable used on all hosts when triggering configuration tests through the web application
- The arguments for configuration tests.
- The strategy to disable hosts when jobs failed
- Whether or not to reschedule jobs that failed
- Enable or disable automatic host availability checks

- The interval in which automatic host availability checks are performed
- Whether or not fingerprints should be verified during automatic host availability checks
- The email address to send a notification to, in case a host was automatically disabled

All changes made on this page are automatically stored.

Logging

This page allows to change the logging settings.

- Enable or disable console output
- Change the logging level of the console output
- Change the logging level of the log file output
- Change the maximum number of log file backups before old logs are overwritten
- Change the maximum size of individual log files

All changes on this page are automatically saved.

Mailer

This page allows to change the settings for the *SshResourceManager*'s mailer functionality.

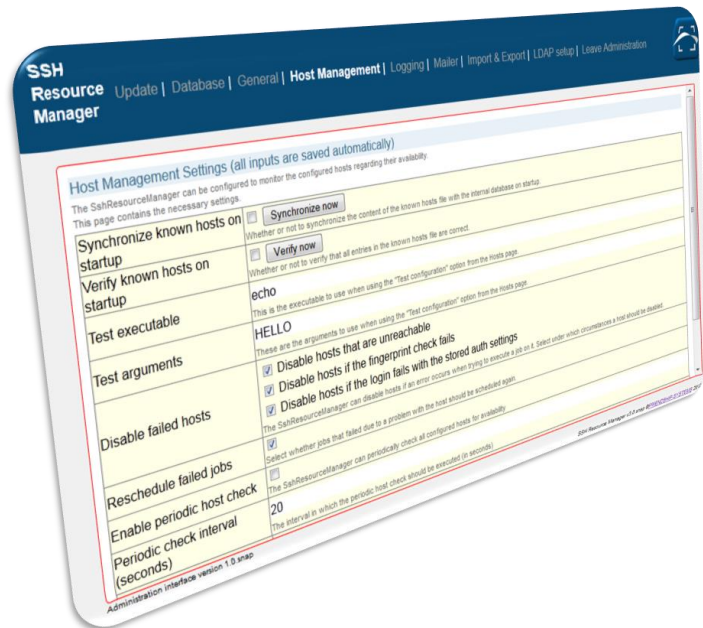
- Enable or disable the mailer
- SMTP server address
- SMTP server port
- Username and password required to authenticate with the SMTP server
- The "from" address for mails sent by the *SshResourceManager*
- A default "to" address for mails sent by the *SshResourceManager*
- Whether or not to send a test mail on start-up and the possibility to trigger a test mail right away.
- Enable or disable mailer debug output

All settings on this page are not stored until the *Submit* button was pressed.

B.5 Notes

Additional information can be found in the documentation shipped with CAESES. If you have further questions and/or problems that are not covered in this document or in the user guide, do not hesitate to contact FRIENDSHIP-SYSTEMS.

Part C | Administration Interface



C.1 Introduction

The administration interface allows to configure most settings that are explained in part A of this guide at runtime. Additionally, it allows updating and restarting the *SshResourceManager*.

Installation

Similar to the *SshConfigurationManager*, the administration interface is included in the *SshResourceManager* and needs no additional setup. However, since the username and password of the administration interface is stored inside the configuration file of the *SshResourceManager*, it is advised to change the default values for the settings **administratorUser** and **administratorPassword** inside the configuration file. The login can also be changed inside the administration interface on the [Update](#) page.

Login

Once the *SshResourceManager* was started, you can access the administration interface by opening a browser and navigating to the *SshConfigurationManager*. On the bottom of the login page there is a link titled *Administration*. Additionally, once logged in to the *SshConfigurationManager* SuperUsers will have an *Administration* link in the main navigation. Note that the login for the administration interface is not related to the users that were set up for the *SshResourceManager*.

C.2 Usage

Update

After login the initial page shows the current state and version of the *SshResourceManager*. Internally the *SshResourceManager* system consists of three parts: The *SshResourceManager* itself, which does the actual job and resource management, the Server which represents the lower layer that performs the network transportation and the administration interface.

Besides pure information this page also allows to update and restart the *SshResourceManager*. Restarting the *SshResourceManager* can be helpful after manually editing the configuration file.

In order to update the *SshResourceManager*, press the *Upload file* button or drag and drop a file from a file explorer to the button. The file is then uploaded to the server, verified and checked for the versions of the contained components of the *SshResourceManager*. If the file containing the new version is already located on the server, it is also possible to browse the server's file system by pressing the *Browse* button.

If new versions of the components are found, the buttons to start the update are enabled. If no new versions are found (e.g. when downgrading to a previous version) the *Force* option can be used to bypass the version check and perform the update nevertheless. If the *Dry run* option is enabled, the update will only be simulated.

At the bottom of this page, the login data for administration interface can be changed. Note that changing that data requires a restart of the *SshResourceManager* system. The restart is triggered automatically and the page will be reloaded once the system is up again. It is then necessary to log in using the new data.

Database

On this page the database that is used by the *SshResourceManager* can be configured. In order to be able to communicate with a database server, the *SshResourceManager* requires a JDBC (Java Database Connectivity) driver that matches the database management system used. JDBC drivers are available for all common databases. However, due to license restrictions they cannot be shipped with the *SshResourceManager*. Please download the appropriate JDBC driver for your database and put it into the *dbdriver* subdirectory of the *SshResourceManager* installation.

Note that all changes on this page are not stored until the *Submit* button on the bottom is pressed.

JDBC Driver

The *SshResourceManager* needs to know the component (class) of the JDBC driver. The name should be contained in the documentation of the downloaded driver. Please enter that name into the text-field.

Hibernate Dialect

As different database systems understand slightly different dialects of the common query language, the *SshResourceManager* needs to know the dialect understood by your system. Please select the appropriate entry from the list. Please see the following link for additional information:
<http://docs.jboss.org/hibernate/orm/3.5/api/org/hibernate/dialect/package-summary.html>

Database URL

This setting holds the address (Uniform Resource Locator) of your database in JDBC syntax. The basic syntax is as follows:

```
jdbc:<database_system>://<server_name>/<database_name>
```

Example:

```
jdbc:mysql://localhost/myDatabase
```

Timeout

For performance reasons, database management systems cache connections that are no longer used. This property should reflect the setting of your database system after which such connections are considered to be no longer valid.

User name

This is the user name that is used when connecting to the database.

Password

This is the password that is used when connecting to the database.

General

This page contains general settings that configure the behavior of the *SshResourceManager*.

Job Data Dir

In this directory temporary data of the *SshResourceManager* is stored (e.g. input/output files of the jobs). If this is relative path, it is resolved using the *SshResourceManager*'s installation directory as the root.

Input File Server Port

Additional to the main (control) port of the *SshResourceManager*, it listens on two more ports for transferring the in- and output files. This configures the port to use for receiving input files that are send by CAESES. A setting of 0 means that the *SshResourceManager* will use a random free port.

Result File Server Port

Additional to the main (control) port of the *SshResourceManager*, it listens on two more ports for transferring the in- and output files. This configures the port to use for sending result files to CAESES. A setting of 0 means that the *SshResourceManager* will use a random free port.

File Transfer Timeout

This is the timeout (in ms) after which file transfers will timeout (i.e. fail) if no data arrives.

Output Buffer Size

This is the internal buffer size (in MB) that is allocated by the *SshResourceManager* when sending result files. Larger values can result in higher speeds, but may cause false timeouts.

Maximum number of jobs

This is the number of jobs that the *SshResourceManager* can execute simultaneously.

Delete temporary remote files

If this is unchecked all temporary data of a finished job will be left in the temporary directory of the remote computers after a job was completed. This may fill up the hard drives of the remote computers very quickly.

Allow commands in arguments

By default the *SshResourceManager* does not allow to execute jobs that have commandline arguments which would cause multiple commands to be executed (i.e. &, || and ;). Enable this checkbox if you want to allow this. Note that it might be a security risk to enable this option as it would, for example, allow to pass `"|| rm -rf /"` as an argument.

Time Zone

This is the time zone to use when the *SshResourceManager* prints time codes (e.g. in logfiles).

Host Management

The *SshResourceManager* can be configured to monitor the configured hosts regarding their availability and react on problems. This page contains the necessary settings.

Synchronize known hosts on startup

When the *SshResourceManager* starts, it can synchronize the file storing the known hosts with its internal database. This is useful when using the system's global known hosts file. Note that the file to store the known hosts in cannot be changed at runtime but only by editing the configuration file manually.

Verify known hosts on startup

During start up, The *SshResourceManager* can verify whether the fingerprints stored in the known hosts file are matching the fingerprints of the hosts. If that is not the case the host will be marked as having being unknown.

Test executable

This is the executable to use when using the *Test configuration* option on the *SshConfigurationManager*'s Hosts page.

Test arguments

This is the argument string to use when using the *Test configuration* option on *SshConfigurationManager*'s Hosts page.

Disable failed hosts

Select under which circumstances the *SshResourceManager* will automatically disable a host.

Reschedule failed jobs

Select whether to jobs that failed due to connection problems should be re-entered into the waiting jobs queue.

Enable periodic host check.

The *SshResourceManager* can periodically check all (active) configured hosts whether they are available and disable them if one of the problems configured in the *Disable failed hosts* setting apply.

Periodic check interval

This is the interval in seconds in which the periodic host check (if enabled) is executed.

Check fingerprints

This determines whether the periodic host check (if enabled) should also verify the fingerprints. Note that the fingerprint check may take some time, especially for Linux hosts.

Notification email address

If the mailer is configured, an email will be sent to this address if a problem was identified that caused a host to be automatically disabled. There will also be a notification email if there are no hosts left to execute a certain jobs due to automatic disabling of hosts.

Logging

The *SshResourceManager* produces console log output as well as HTML logfiles. This page allows to configure the verbosity of the log output as well as the file handling of the logfiles. The logfiles can also be accessed by clicking the *Show logs* link at the top of the page.

Enable Console logging

If this is disabled, the *SshResourceManager* will not produce any output on the console.

Console logging level

This is the minimum severity of a message that is printed to the console. The logging levels are:

- DEBUG – highest verbosity
- INFO – medium verbosity
- WARN – print warnings only
- ERROR – show errors only

File logging level

This is the minimum severity level of a message that is written into the log files. The available levels are the same as for the console logging.

Maximum logfile backups

The *SshResourceManager* keeps backups of previous logfiles. This property holds the number of backups to be stored, before old files are overwritten.

Maximum logfile size

This is the maximum file size in KB of individual logfiles.

Mailer

The *SshResourceManager* can be configured to send information emails for certain events. This includes notification mails to users that their job has finished as well as notifications to the administrator if a host was automatically disabled. In order to activate that feature, provide the information on this page. Note that all settings are not stored until the submit button is pressed.

Enable Mailer

This property determines whether or not to enable the mailer.

SMTP Server Address

This is the address of the SMTP server to send emails from.

SMTP Server Port

This is the port of the SMTP server.

SMTP User

If the SMTP server requires authentication, the username can be configured with this setting.

SMTP Password

If the SMTP server requires authentication, this is the password used to log in.

From address

This is the email address that will appear in the *from* field of emails sent by the *SshResourceManager*

Default to address

This is the default address to send emails to, if, for example, a user has no email address configured.

Send test mail

If this option is enabled, the *SshResourceManager* will send a test email on startup to verify that the mailer was set up correctly.

SMTP Debug

If problems occur with the SMTP settings, this allows to enable additional console output to identify the problem.

Import & Export

The configuration of the *SshResourceManager* can be exported to XML. Press the *Export* button to save the current configuration to disk. An exported configuration can be imported again. Press the *Upload file* button or drag and drop a file to it from a file explorer. It is uploaded to the *SshResourceManager* and validated. If it is valid, the *Import* button is enabled.

The exported configuration also includes the database settings. When, for example, the database should be switched, it is advisable not to import those settings. However, note that when importing the database (either the one currently configured or the one imported from the XML file) will be cleared before importing.

LDAP setup

The *SshResourceManager* allows to bind to an existing LDAP server for its user management. Any LDAP server that supports the LDAPv3 protocol as described in [RFC2251](#) and updated in [RFC4511](#) is supported.

To be able to login to the *SshResourceManager* with this configuration, users must either be created manually on the Users page of the *SshConfigurationManager* or the *Copy user on first login* option needs to be enabled. If that option is used, new users will automatically be copied into the *SshResourceManager*'s internal database on first login and can be further modified there, but modifications will not reflect to the LDAP server. In order to automatically assign access roles to users that have been created using the *Copy user on first login* option, it is also possible to create mappings between LDAP groups and the user roles known by the *SshResourceManager*.

Enable LDAP Authentication

Enable and disable LDAP authentication. Note that it is not possible to disable LDAP authentication if users exist that have the *LDAP only* option set.

Name

This is the hostname of the server running LDAP.

Port

This is the port at which the LDAP server is listening.

Username

If the LDAP server requires authentication, this is the username used to connect to the LDAP server.

Password

If the LDAP server requires authentication, this is the password used to connect to the LDAP server.

Login cache time

Successful logins using LDAP are cached for this time (in seconds) in order to reduce the load on the LDAP server.

Copy user on first login

If this option is enabled, an LDAP user will be automatically copied to the *SshResourceManager* internal database on first login if it does not exist.

Default user level

If the *Copy user on first login* option is enabled, the new user will have this user level by default, if none of the group mappings apply or if no group mappings are configured.

Synchronize group membership

If this setting is enabled, the user level of a user will be synchronized with its primary LDAP group (if group mappings are configured).

Connection mode

Choose whether to connect to the LDAP server using a secure SSL/StartTLS connection. Note that no certificate verification is performed.

Base DN

This is the LDAP root node from which to search for users and groups, for example: `cn=users,dc=company,dc=com`

User name Attribute

This is the attribute that holds a user's name, for example `uid`.

Additional User DN

This is prepended to the base DN to limit the scope when searching for users, for example `ou=people`.

User object filter

This is the complete filter (including parenthesis and operators) to use when searching for users, for example `(objectclass=posixAccount)`.

Primary group id attribute

This is the LDAP attribute that holds the user's primary group. This is optional, but required to set up group mappings.

User Email attribute

The LDAP attribute that contains the user's email address.

Group name attribute

This is the attribute to use for the group name. It is required if setting up group mappings.

Additional Group DN

This is prepended to the base DN to limit the scope when searching for groups

Group object filter

This is the complete filter (including parenthesis and operators) to use when searching for groups, for example `(objectclass=posixGroup)`.

Group id attribute

This is the attribute to use for the group id.

Locked user group

If *Copy user on first login* is enabled, users with this group set as their primary group are created as *Locked Users*.

Normal user group

If *Copy user on first login* is enabled, users with this group set as their primary group are created as *Users*.

SuperUser group

If *Copy user on first login* is enabled, users with this group set as their primary group are created as *SuperUsers*.