KNIME Server Administration Guide

KNIME AG, Zurich, Switzerland
Version 4.18 (last updated on)

Table of Contents

Introduction

This guide covers in detail the configuration options for KNIME Server.

If you are looking to install KNIME Server, you should first consult the KNIME Server Installation Guide.

For administration options to configure KNIME WebPortal please refer to the KNIME WebPortal Administration Guide.

For guides on connecting to KNIME Server from KNIME Analytics Platform, or using KNIME WebPortal please refer to the following guides:

- KNIME Server User Guide

- KNIME WebPortal User Guide

An additional resource is also the KNIME Server Advanced Setup Guide.

Release Notes

KNIME Server 4.18 is a feature release of the 4.x release line and is recommended for use in production environments.

This version of KNIME Server works **only** with **executors** of version 5.4.

All local KNIME Analytics Platform **clients** that have worked with KNIME Server 4.17 will continue to work with KNIME Server 4.18 without restrictions.

> **i** To find out which version of KNIME Server you are currently running, you can check the Administration pages on the WebPortal.

Possible Security Policy Updates

- If you have set your CSP header to a custom value, you might need to adjust the CSP header for the included fonts to load correctly. Specifically `data`-URLs need to be allowed for fonts. If you have no `font-src` defined, simply adding `font-src 'self' data:;` to the end of your rules will suffice, otherwise adjust `font-src` accordingly.

Server architecture

KNIME Server is a Java Enterprise Application, and the KNIME WebPortal a standard Java Web Application, both installed on a Tomcat application server, the blue box in the middle of the figure below. Users can log in to the server and the server will authenticate against any authentication source provided by Tomcat.
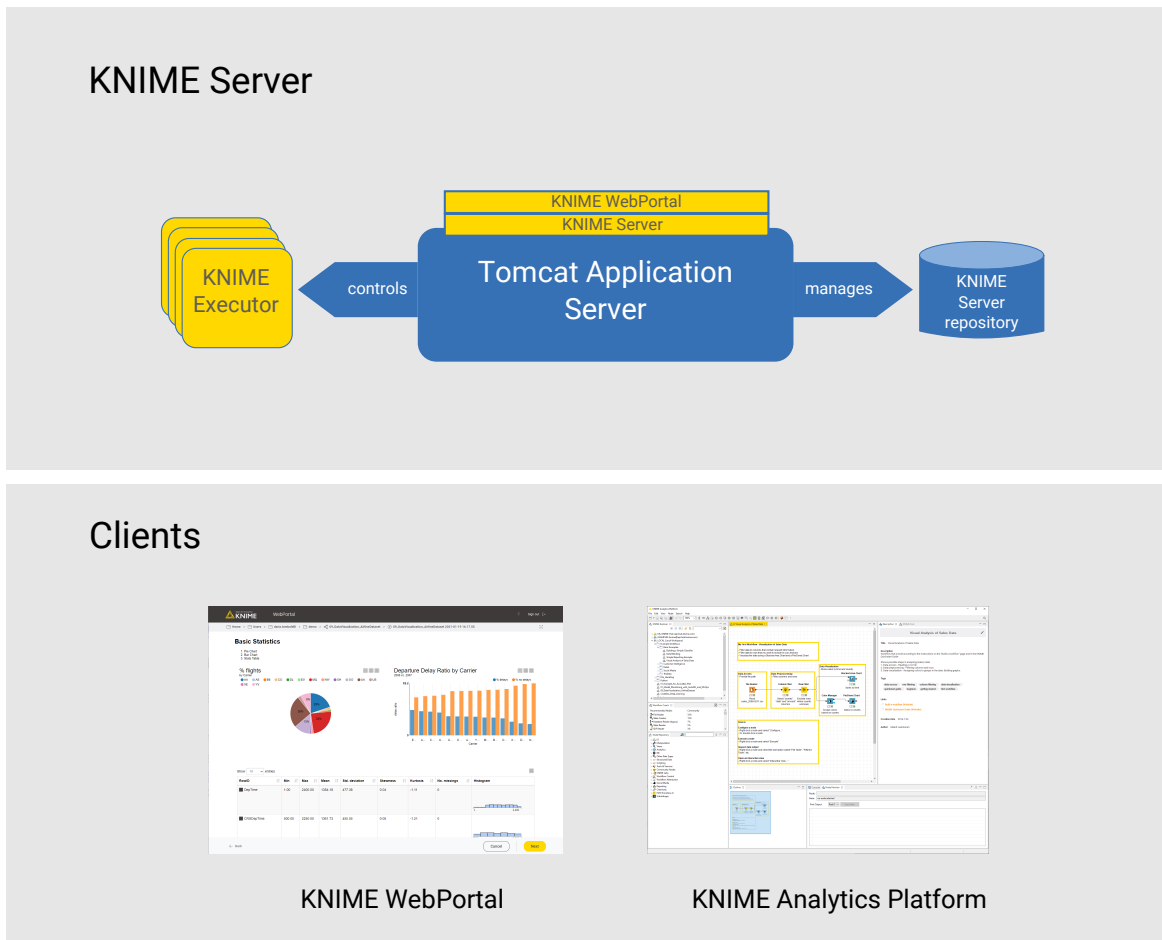
One of the main tasks of KNIME Server is to manage and control the server's repository. Workflows uploaded to the server go through the server application and are stored in the repository which is just a folder on the server's file system (the blue cylinder on the right in the diagram). Access to the stored workflows is controlled in KNIME Server and access rights for the workflows can be manipulated from KNIME Explorer once the client side server extensions are installed.

Workflow execution on the server is carried out by a KNIME Executor. The KNIME Executor is a persistent headless instance of a normal KNIME Analytics Platform application (leftmost element in the diagram above).

It is important to note that workflows can only be successfully loaded and executed on the server, if the executor has the required features installed and is of the same version (or

newer) than the KNIME Analytics Platform version that was used to create the workflow.

Monitoring and administration portal

With KNIME Server version 4.12 new monitoring and administration portals have been added. They are reachable when signing in to the KNIME WebPortal as an administrator.

Through the monitoring portal you can have an overview on which jobs are running and their current state, as well as which schedules are currently active, and gain information on how executors are being used.

The administration portal, instead, gives you an overview on the status of your KNIME Server, allows for an easy upload of a new license file, and also to configure your KNIME Server options through a browser-based user interface.
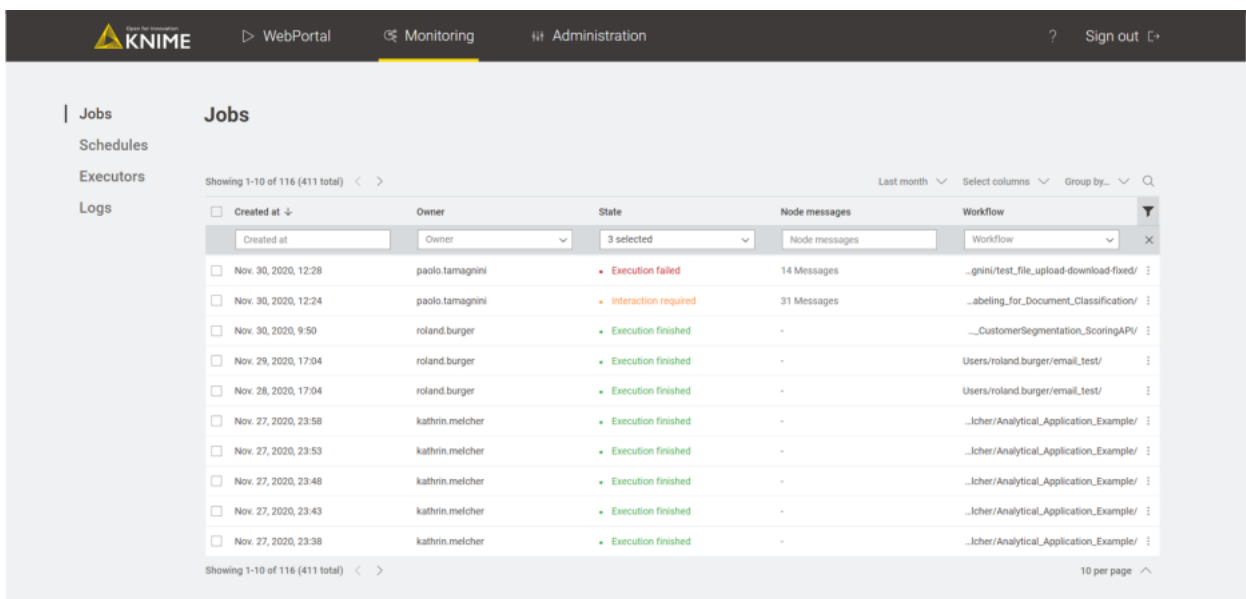
> ℹ️ With KNIME Server version 4.12, the monitoring portal is available also to users without administrator privileges but it will be showing **only** *Jobs* and *Schedules* owned by the specific user. Consumers, instead, do not have access to any of these portals.

Monitoring portal

The monitoring portal presents four sections:

- *Jobs*: An overview of all the jobs that currently exist on KNIME Server, both from the KNIME WebPortal or from the KNIME Analytics Platform on KNIME Server.

Here you can:

- Refine the table: You can show only jobs that ran in a specific time range, you can select the columns you want to show in the table, regroup the jobs in the table by different criteria, e.g. by *Owner*, *State*, or *Workflow*, or search through the list of jobs by keyword

- Filter the table: Click the filter icon on the right side of the table to filter the jobs listed in the table by keyword per column

- Delete job(s): You can select jobs via the checkbox on the left, and delete them. You can discard jobs one by one clicking the three dots at the end of a job's row and choose *Discard*.

- Show node messages: Click the message cell corresponding to the job whose messages you want to see to show them.

- *Schedules*: An overview of all the scheduled jobs on KNIME Server.

Also here you can use the filtering options described above. It is also possible to activate or deactivate a specific schedule by the toggle button on the corresponding schedule row or a group of schedules by selecting them with the checkbox on the left.

- *Access tokens*: This section is an overview of the access tokens created by users to access WebPortal workflows without needing to login.

- *Executors*: An overview of the status of your KNIME Executor(s), grouped by executor groups. Here all the jobs associated with the executor are shown as well as information about the usage of the Executors.
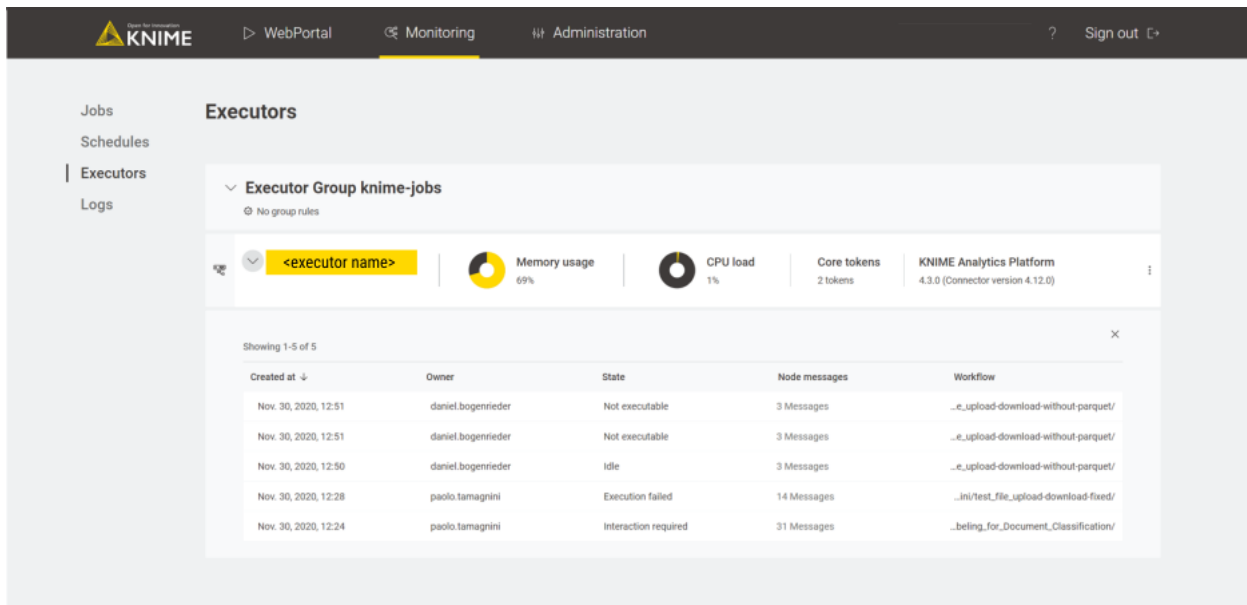
Here you can also set Executors to a draining state. To do so click the three vertical dots icon at the end of the Executor line, and select *Set to draining state* from the context menu.

- *Logs*: In this section you can download all logs of your KNIME Server.
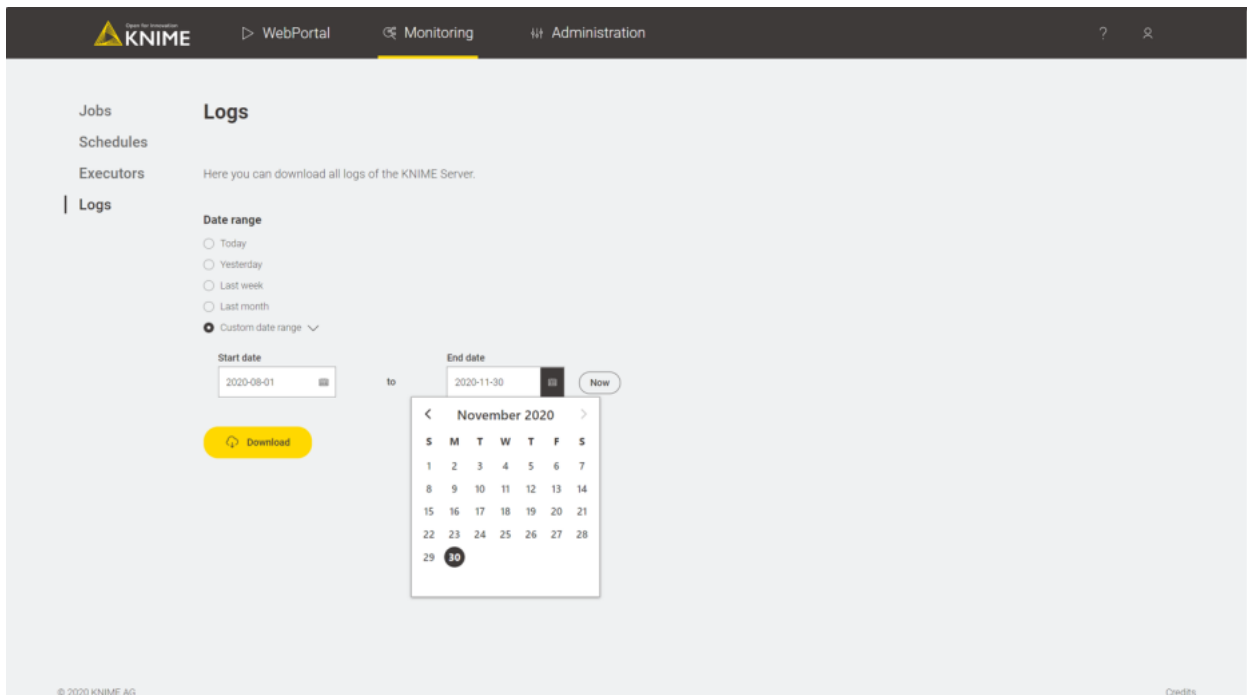
Administration portal

The administration portal presents four sections:

- *License*: An overview of the KNIME Server status with the following information:
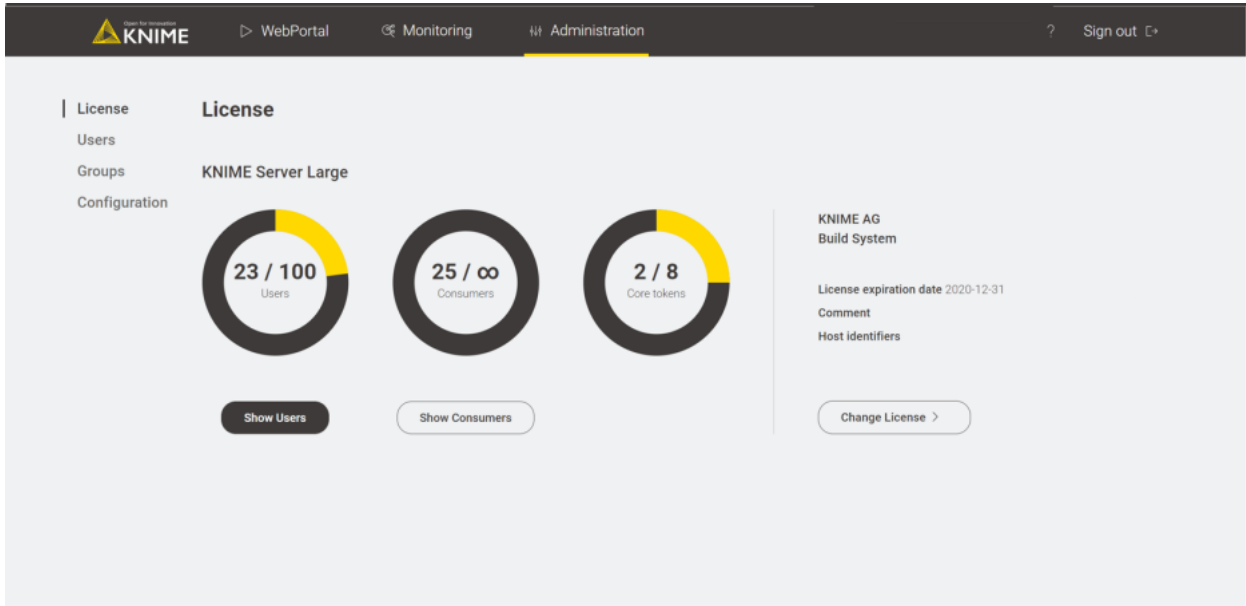
License Type	Type of the used license.
Number of Users	The number of active and available users. Click *Show Users* button to show a list of currently active users.
Number of Consumers	The number of active consumers. Click *Show Consumers* button to show a list of currently active consumers.
Number of Core tokens	The number of used and available Core tokens.
Company	The company the license has been issued for.
License expiration date	The expiration date of the current license in the format YYYY-MM-DD.
Comment	An optional comment regarding the license.
Host identifiers	Host information used to check the license against. This can be, for instance, MAC- or IP-addresses.

Click the button *Change License* to upload a new license file.

• *Users/Groups*: An overview of the users and the groups they have been added to.

> ℹ The users and groups management is only available if the database-based authentication is chosen (see section Database-based authentication). If the LDAP is configured, the users and groups management will not be available.
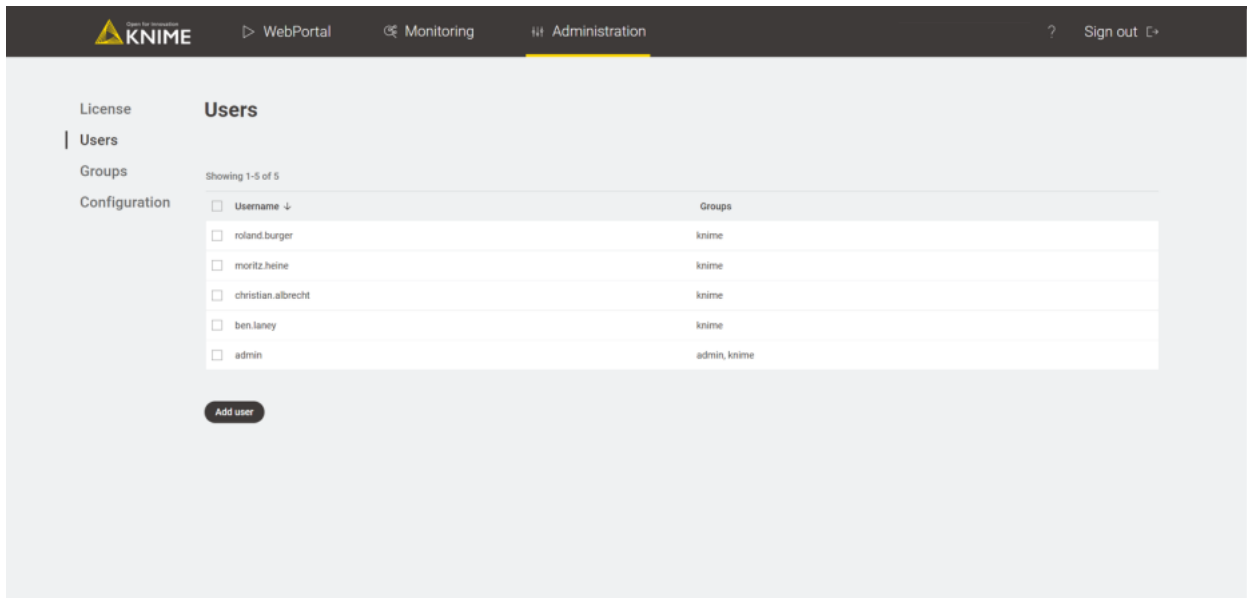
In the users section you can:

- Add a user by clicking the *Add user* button. A panel on the right will open where you can type the *User name*, the *Password*, and assign the user to the available groups, choosing the groups from a drop-down list menu.
- Delete user(s) by checking the corresponding check-box and clicking *Delete*.

Through the Groups section you can:

- Add a group by clicking the *Add group* button. A panel on the right will open where you can type the *Group name*.
- Delete group(s) by checking the corresponding check-box and clicking *Delete*.

- *Configuration*: This section is an overview of the current settings of the KNIME Server. You can set up here all the configurations that are available for the Server instance. For an more detailed explanation of the possible configuration options, please refer to the Server configuration files and options section.

Job

License

Users

Groups

| Configuration

General

Action

Authentication

Executor

Job

noVNC

Repository

WebPortal

com.knime.server.job.max_execution_time

Allows to set a maximum execution time for jobs.

```
unlimited
```

com.knime.server.job.default_cpu_requirement

Specifies the default CPU requirement in number of cores of jobs without a specific requirement set.

```
0
```

com.knime.server.job.max_lifetime

Specifies the time of inactivity, before a job gets discarded.

```
7d
```

com.knime.server.job.default_report_timeout

Specifies how long to wait for a report to be created by an Executor.

```
1m
```

com.knime.server.job.default_load_timeout

Specifies how long to wait for a job to get loaded by an executor.

```
3m
```

Scroll down
to see all configuration
options

Users, consumers, and cores

User: a user is an individual person with a unique ID and may only be an employee, contractor, advisor, or agent of the customer who is authorized by the customer to use KNIME Software. Users have access to the full functionality of KNIME Software and are active for as long as they own items on KNIME Server such as workflows, data, folders, or scheduled jobs, etc. The license defines the permitted number of users.

Consumer: a consumer is either an individual person (internal or external) or a machine with a unique ID, who is authorized by the customer to use KNIME Software. Their only permitted interactivity is to execute workflows either via the provided API or via KNIME WebPortal. The license defines the permitted number of consumers. KNIME Server Large has an unlimited number of consumers.

Core: a core refers to a physical CPU core in the case of physical hardware and a "virtual CPU" core in the case of infrastructure as a service. In cases where the infrastructure as a service defines a relationship between physical CPUs and virtual CPUs, the physical CPU core count shall prevail. The number of licensed cores for execution refers to the maximum number of CPU cores that can be allocated to running the software. The license defines how many CPU cores can be used by KNIME Software.

Server configuration files and options

KNIME Server can be configured:

- Via the administration portal accessible through the KNIME WebPortal

- By manually changing the options on the `knime-server.config` file

When changing the Server configurations from the administration portal the `knime-server.config` file is automatically overwritten.

All the options that are listed in the next section are configurable via the administration portal.

KNIME Server configuration file

When manually configuring KNIME Server you need access to the `knime-server.config` file. The file can be found in `<knime-server-repository>/config/knime-server.config`. Most of the parameters defined in this file can be changed at runtime and will take effect as soon as possible. Default values will be used for empty or missing configuration options.

The section KNIME Server configuration file options contains a comprehensive list of all configuration options and explanations. For a list of all configuration options and explanations valid for KNIME WebPortal please refer to the configuration file options section of the KNIME WebPortal Administration Guide.

KNIME Server configuration file options

Below you will find a table with all supported configuration options (in alphabetical order). Some of them are described in more detail in later sections. The options can be set in the file `<knime-server-repository>/config/knime-server.config`.

For Windows users: For paths in the server configuration file either use forward slashes ("/") or double backslashes ("\\"). A single backslash is used to escape characters.

The following annotations to the table, provide some additional information about which Executor type is affected, and whether changes take effect at runtime, or require a server restart.

[ST]   changes take effect after a restart of KNIME Server

[RT]   changes can take effect at runtime

Some options can be set as property in the `knime-server.config` file as well as by defining an environment variable (Env). The environment variable changes will only take effect after a restart of KNIME Server. If the environment variable for an option is set, the property in the configuration file will be ignored.

`com.knime.server.admin_email=<email>,<email>,···` **[RT]**

> A comma separated list of email addresses that will get notified when an action is required regarding the license, e.g. the license is about to expire or the maximum number of users has been reached etc.

`com.knime.server.canonical-address=<URL to server>` **[RT]**

> The communication between Executor and server is performed through the server's REST interface. In case auto-detection of the server's address doesn't work correctly, you have to specify the canonical address here, e.g. http://knime-server:8080/. This option is not required if server and Executor are running on the same computer. See also section enabling workflow execution below for more details.

Env: `KNIME_SERVER_CANONICAL_ADDRESS=<URL to server>`

`com.knime.server.client.default_version=<Version, e.g. 4.11.0>` **[RT]**

> The default version of the KNIME Server REST API that is supported by clients. This version of the REST API is assumed to be supported by clients in case no `KNIME-API-Version`-header is set. The default value is `-1.0.0` indicating that the most current KNIME Server REST API is supported.

`com.knime.server.config.watch=<true|false>` **[ST]**

> If set to true changes to the configuration file are applied immediately without a server restart. Default is `false`, i.e. all changes will require a server restart.

`com.knime.server.csp-report-only=<true|false>` **[RT]**

> Tells the browser to still serve content that violates the Content-Security-Policy and instead display a warning, by setting the Content-Security-Policy-Report-Only header rather than the Content-Security-Policy header (defaults to false). For more information about Content-Security-Policy-Report-Only, please refer to this resource.

`com.knime.server.default_mount_id=<mount ID>` **[RT]**

Specifies the name of the default mount ID. This is fetched, when clients set up their mount point to the server. Defaults to the server's hostname.

Env: `KNIME_SERVER_DEFAULT_MOUNT_ID=<mount ID>`

`com.knime.enterprise.executor.embedded-broker=<true|false>` **[ST]**

Enables the use of the embedded message queue (Apache Qpid) instead of a separate RabbitMQ installation. This allows you to run distributed KNIME Executors on the same system as the KNIME Server. By default this is disabled.

`com.knime.enterprise.executor.embedded-broker.port=<value>` **[ST]**

Allows to configure the port for the embedded message queue (see option above). The default is 5672 and you should only change it if the port is already in use by another service. You also need to adjust the message broker address in the Executor's *knime.ini* in this case.

`com.knime.enterprise.executor.msgq=amqp://<user>:<password>@<rabbitmq-host>/<vhost>` **[ST]**

URL to the RabbitMQ virtual host. In case RabbitMQ High Available Queues are used, simply add additional `<rabbitmq-host>:<port>` separated by commas to the initial `amqp` address:
`com.knime.enterprise.executor.msgq=amqp://<username>:<password>@rabbitmq-host/knime-server,amqp://<rabbitmq-host2>:<port2>,amqp://<rabbitmq-host3>:<port3>` The client will try all provided addresses in order and will take the first one that works.

Env: `KNIME_EXECUTOR_MSGQ=amqp://<user>:<password>@<rabbitmq-host>/<vhost>`

`com.knime.enterprise.executor.msgq.connection_retries=<value>` **[ST]**

Defines the maximum number of connection retries for the message queue, that should be performed during server startup. The delay between retries is 10 seconds. The default is `-1`. For values less than 0 the server will try to reconnect indefinitely.

Env: `KNIME_MSGQ_CONNECTION_RETRIES=<value>`

`com.knime.enterprise.executor.msgq.tls_version=<value, e.g. TLSv1.2>`

> Defines which TLS version should be used when connecting to RabbitMQ. The supported TLS version depends on the used Java Virtual Machine and the TLS support of RabbitMQ. By default TLS 1.2 will be used.

Env: `KNIME_EXECUTOR_MSGQ_TLS_VERSION=<value>`

`com.knime.enterprise.executor.msgq.enable_hostname_verifier=<true|false>`

> Defines if the hostname of the RabbitMQ server should be verified when connecting via TLS. By default the hostname won't be verified.

Env: `KNIME_EXECUTOR_MSGQ_ENABLE_HOSTNAME_VERIFIER=<true|false>`

`com.knime.enterprise.executor.msgq.truststore_path=<value>`

> Defines the path to the truststore that contains the server certificate used by RabbitMQ. By default all certificates are accepted.

Env: `KNIME_EXECUTOR_MSGQ_TRUSTSTORE_PATH=<value>`

`com.knime.enterprise.executor.msgq.truststore_algorithm=<value>`

> Defines the truststore algorithm for the specified `com.knime.enterprise.executor.msgq.truststore_path`. By default the system's default algorithm will be used. In case `com.knime.enterprise.executor.msgq.truststore_path` is not set this option is ignored.

Env: `KNIME_EXECUTOR_MSGQ_TRUSTSTORE_ALG=<value>`

`com.knime.enterprise.executor.msgq.truststore_passphrase=<value>`

> Defines the truststore passphrase for the specified `com.knime.enterprise.executor.msgq.truststore_path`. By default an empty passphrase is used. In case `com.knime.enterprise.executor.msgq.truststore_path` is not set this option is ignored.

Env: `KNIME_EXECUTOR_MSGQ_TRUSTSTORE_PASSPHRASE=<value>`

`com.knime.enterprise.executor.msgq.truststore_type=<value>`

> Defines the truststore type for the specified
> `com.knime.enterprise.executor.msgq.truststore_path`. By default the system's
> default type will be used. In case
> `com.knime.enterprise.executor.msgq.truststore_path` is not set this option is
> ignored.

Env: `KNIME_EXECUTOR_MSGQ_TRUSTSTORE_TYPE=<value>`

`com.knime.enterprise.executor.msgq.keystore_path=<value>` **[ST]**

> Defines the path to the keystore that contains the client certificate provided to
> RabbitMQ when connecting via TLS. By default no keystore is set.

Env: `KNIME_EXECUTOR_MSGQ_KEYSTORE_PATH=<value>`

`com.knime.enterprise.executor.msgq.keystore_algorithm=<value>` **[ST]**

> Defines the keystore algorithm for the specified
> `com.knime.enterprise.executor.msgq.keystore_path`. By default the system's
> default algorithm will be used. In case
> `com.knime.enterprise.executor.msgq.keystore_path` is not set this option is
> ignored.

Env: `KNIME_EXECUTOR_MSGQ_KEYSTORE_ALG=<value>`

`com.knime.enterprise.executor.msgq.keystore_passphrase=<value>` **[ST]**

> Defines the keystore passphrase for the specified
> `com.knime.enterprise.executor.msgq.keystore_path`. By default an empty
> passphrase is used. In case `com.knime.enterprise.executor.msgq.keystore_path` is
> not set this option is ignored.

Env: `KNIME_EXECUTOR_MSGQ_KEYSTORE_PASSPHRASE=<value>`

`com.knime.enterprise.executor.msgq.keystore_type=<value>` **[ST]**

> Defines the keystore type for the specified
> `com.knime.enterprise.executor.msgq.keystore_path`. By default the system's
> default type will be used. In case
> `com.knime.enterprise.executor.msgq.keystore_path` is not set this option is
> ignored.

Env: `KNIME_EXECUTOR_MSGQ_KEYSTORE_TYPE=<value>`

`com.knime.enterprise.executor.msgq.names=<value>,<value>,···` **[RT]**

> Defines the names of Executor Groups. The number of names must match the
> number of rules defined with `com.knime.enterprise.executor.msgq.rules`. See
> executor groups for more information.

`com.knime.enterprise.executor.msgq.rules=<value>,<value>,···` **[RT]**

> Defines the exclusivity rules of the Executor Groups. The number of rules must match
> the number of names defined with `com.knime.enterprise.executor.msgq.names`. See
> executor groups for more information.

`com.knime.enterprise.executor.msgq.tokens=<value>,<value>,···` **[RT]**

> Defines the maximum number of tokens usable by the individual Executor Groups.
> While it is possible to define more tokens than provided by the license the minimum
> of the available tokens is used. The number of values must match the number of
> names defined with `com.knime.enterprise.executor.msgq.names`.

`com.knime.server.executor.blacklisted_nodes=<node>,<node>,···` **[RT]**

> Specifies nodes that are blacklisted by the server, i.e. which aren't allowed to be
> executed. For blacklisting a node you have to provide its factory name. Wildcards (*)
> are supported. For more information see here.

`com.knime.server.executor.max_log_file_timeout=<duration with unit, e.g. 30s>`
[RT]

> Specifies the maximum time the servers wait for executors to send their logs files when they are requested by an admin. The default is `10s` and is sufficient in most cases. However if you have executors with large log files or the connection to the executor is slow then you may have to increase the timeout in case some executors' log file are missing from the downloaded archive.

`com.knime.server.executor.watchdog.interval=<duration with unit, e.g. 20m>` **[RT]**

> Specifies the maximum time an executor can be absent, i.e. not responding to status requests, before its jobs are marked as vanished (see Executor Watchdog). In case the value is set to an equivalent of `0s` the watchdog is shutdown and a restart of the server is required to start the watchdog again. It is recommended to set this option to a reasonable high value as otherwise it might generate additional load on the server and executors. The value is capped at `60m`, i.e. higher values will have the exact same behavior. The default is `5m`.

`com.knime.server.executor.reject_future_workflows=<true|false>` **[RT]**

> Specifies whether the Executor should reject loading workflows that have been create with future versions. For new installations the value is set to true. If no value is specified the Executor will always try to load and execute any workflow by default.

`com.knime.server.executor.update_metanodelinks_on_load=<true|false>` **[RT]**

> Specifies whether component links in workflows should be updated right after the workflow has been loaded in the KNIME Executor. Default is not to update component links.

`com.knime.server.gateway.timeout=<duration with unit, e.g. 30s, 1m>` **[RT]**

> Specifies the timeout used internally for gateway requests coming from the KNIME Analytics Platform Remote Job View or from KNIME WebPortal. Default value is `1m`.

`com.knime.server.job.async_load_reconnect_timeout=<duration with unit, e.g. 60m, 36h, or 2d>` **[RT]**

> Specifies the default connection timeout of asynchronously loaded jobs in case of a server restart. If a server restart occurs the server tries to reconnect to jobs that have been loaded asynchronously, as they might be still in the message queue or discarded due to an error. For this the maximum of the remaining load timeout or the `async_load_reconnect_timeout` is used to wait for status updates. If the time elapses without a status update loading will be canceled and the job state will be set to `LOAD_ERROR`.

`com.knime.server.job.execution_retry_waiting_time=<duration with unit, e.g. 30s, 2m, 1h>` **[RT]**

> Specifies the waiting time between execution retries of failed jobs (see the KNIME Server User Guide).
> The default is `0s`.

`com.knime.server.job.default_cpu_requirement=<number of cores as decimal, e.g. 0, 0.1, 1, 2, 4>` **[RT]**

> Specifies the default CPU requirement in number of cores of jobs without a specific requirement set. See CPU and RAM requirements for more information. The default is `0`.

`com.knime.server.job.default_load_timeout=<duration with unit, e.g. 60m, 36h, or 2d>` **[RT]**

> Specifies how long to wait for a job to get loaded by an Executor. If the job does not get loaded within the timeout, the operation is canceled. The default is `3m`. This timeout is only applied if no explicit timeout has been passed with the call.

`com.knime.server.job.default_ram_requirement=<memory with unit, e.g. 512MB, 4GB>` **[RT]**

> Specifies the default RAM requirement of jobs without a specific requirement set. See CPU and RAM requirements for more information. In case no unit is provided it is automatically assumed to be provided in megabytes. The default is `0MB`.

`com.knime.server.job.default_report_timeout=<duration with unit, e.g. 60m, 36h, or 2d>` **[RT]**

>Specifies how long to wait for a report to be created by an Executor. If the report is not created within the timeout, the operation is canceled. The default is `1m`. This timeout is only applied if no explicit timeout has been passed with the call.

`com.knime.server.job.default_swap_timeout=<duration with unit, e.g. 60m, 36h, or 2d>` **[RT]**

>Specifies how long to wait for a job to be swapped to disk. If the job is not swapped within the timeout, the operation is canceled. The default is `1m`. This timeout is only applied if no explicit timeout has been passed with the call (e.g. during server shutdown).

`com.knime.server.job.disable_use_knime_server_login=<true|false>` **[RT]**

>Specifies whether the option 'User KNIME Server Login' of the credential nodes should be disabled (`true`). In case it is disabled the server won't extract the credentials from the request, otherwise it will try to extract the credentials and store it encrypted with the job information. The default value is `false`.

`com.knime.server.job.discard_after_timeout=<true|false>` **[RT]**

>Specifies whether jobs that exceeded the maximum execution time should be canceled and discarded (`true`) or only canceled (`false`). May be used in conjunction with `com.knime.server.job.max_execution_time` option. The default (`true`) is to discard those jobs.

`com.knime.server.job.exclude_data_on_save=<true|false>` **[RT]**

>Specifies whether node outputs of jobs that are saved as workflows shall be excluded. If this is set to `true` the resulting workflows will be reset, i.e. no output data are available at the nodes. The default value is `false`.

`com.knime.server.job.max_execution_time=<duration with unit, e.g. 60m, 36h, or 2d>` **[RT]**

> Allows to set a maximum execution time for jobs. If a job is executing longer than this value it will be canceled and eventually discarded (see `com.knime.server.job.discard_after_timeout` option). The default is unlimited job execution time. Note that for this setting to work, `com.knime.server.job.swap_check_interval` needs to be set a value **lower** than `com.knime.server.job.max_execution_time`.

`com.knime.server.job.max_lifetime=<duration with unit, e.g. 60m, 36h, or 2d>` **[RT]**

> Specifies the time of inactivity, before a job gets discarded (defaults to `7d`). Negative numbers disable forced auto-discard.

`com.knime.server.job.webportal.max_lifetime=<duration with unit, e.g. 60m, 36h, or 2d>` **[RT]**

> Specifies the time of inactivity, before a job started from the WebPortal gets discarded (defaults to the same value as `com.knime.server.job.max_lifetime`), negative numbers disable forced auto-discard. Only change this value in case you are having issues with a single setting for all job types.

`com.knime.server.job.max_schedule_failures=<number>` **[RT]**

> Specifies the maximum number of consecutive failures to start a scheduled job before the schedule gets disabled. The default value is three consecutive failures. If a negative value is provided (e.g. -1) scheduled jobs will never get disabled due to failures.

`com.knime.server.job.max_time_in_memory=<duration with unit, e.g. 60m, 36h, or 2d>` **[RT]**

> Specifies the time of inactivity before a job gets swapped out from the Executor (defaults to `60m`).
> Negative numbers disable swapping.

`com.knime.server.job.webportal.max_time_in_memory=<duration with unit, e.g. 60m, 36h, or 2d>` **[RT]**

> Specifies the time of inactivity before a job started from the WebPortal gets swapped out from the Executor (defaults to the same value as `com.knime.server.job.max_time_in_memory`), negative numbers disable swapping. Only change this value in case you are having issues with a single setting for all job types.

`com.knime.server.job.save_workflow_summary=<true|false>` **[RT]**

> Specifies if the workflow summary should be stored with the job upon swapping. This should be only disabled in case there are problems during job swapping. Default value is `true`.

`com.knime.server.job.start_execution_timeout=<duration with unit, e.g. 30s, 1m, or 1h>` **[RT]**

> Specifies the time the server will wait for an executor response when triggering execution of a loaded job. If the executor does not respond within the provided time it will cancel the execution and throw an error. Default is `1m`.

`com.knime.server.job.swap_check_interval=<duration with unit, e.g. 30s, 1m, or 1h>` **[RT]**

> Specifies the interval at which the server will check for inactive jobs that can be swapped to disk. Default is every `1m`.

`com.knime.server.login.allowed_groups =<group>,<group>,···` **[RT]**

> Defines the groups that are allowed to log in to the server.
> Default value allows users from all groups.

Env: `KNIME_LOGIN_ALLOWED_GROUPS=<group>,<group>,···`

`com.knime.server.login.consumer.allowed_accounts =<account>,<account>,···` **[RT]**

> Defines account names that are allowed to log in to the server as *consumer*.
> Default value allows login as consumer for all users.

Env: `KNIME_CONSUMER_ALLOWED_ACCOUNTS=<account>,<account>,···`

`com.knime.server.login.consumer.allowed_groups =<group>,<group>,···` **[RT]**

> Defines the groups that are allowed to log in to the server as *consumer*.
> Default value allows login as consumer from all groups.

Env: `KNIME_CONSUMER_ALLOWED_GROUPS=<group>,<group>,···`

`com.knime.server.login.jwt-lifetime=<duration with unit, e.g. 12h or 30d>` **[RT]**

> Defines the maximum lifetime of JSON Web Tokens issued by the server. The default
> value is `30d`. A negative value allows unrestricted tokens (use this value with care
> because there is no way to revoke issued tokens).

`com.knime.server.login.user.allowed_accounts =<account>,<account>,···` **[RT]**

> Defines account names that are allowed to log in to the server as *user*. Default value
> allows login as user for all users.

`com.knime.server.login.user.allowed_groups =<group>,<group>,···` **[RT]**

> Defines the groups that are allowed to log in to the server as a *user*.
> Default value allows login as user from all groups.

`com.knime.server.report_formats=<formats>` **[RT]**

> Defines the different formats available for report generation as a comma separated
> list of values. Possible values are `html`, `pdf`, `doc`, `docx`, `xls`, `xlsx`, `ppt`, `pptx`, `ps`, `odp`, `odt`
> and `ods`. If this value is empty or not set the default list of formats is `html`, `pdf`, `docx`,
> `xlsx` and `pptx`.

`com.knime.server.repository.hide_unreadable_groups=<true|false>` **[RT]**

> When set to `true`, if a user does not have:
>
> - read-permission for workflow groups, e.g. `<workflow_group1>`,
>   `<workflow_group2>`
> - and, write-permission for the parent workflow group, e.g.
>   `<parent_workflow_group>` in path `<parent_workflow_group>/<wokflow_group1>`

then the not readable workflow groups are hidden to the user.
When set to `false`, which is the default value for this option, all workflow groups are shown
as soon as the user has read permissions for the parent workflow group.

`com.knime.server.repository.throttling.concurrent_fetches=<value, e.g. 5, 10>` **[ST]**

> Defines the maximum number of concurrent fetches for the repository. In case the number of the currently fetching clients reaches the number of allowed fetches further clients will be kept waiting by a throttle. Note that only clients are handled by the throttle that request at least the number of specified levels (see `com.knime.server.repository.throttling.min_level`). The default value is the number of available cores.

`com.knime.server.repository.throttling.enable=<true|false>` **[RT]**

> In case it is set to `true` throttling for repository requests is enabled.
> The default value is `false`, so this option is not enabled by default.
>
> ℹ️ This option is experimental and should only be activated in case there are problems where KNIME Server might get unresponsive after too many clients are connected at the same time.

`com.knime.server.repository.throttling.keep_alive_interval=<duration with uni, e.g. 30s, 1m>` **[RT]**

> The interval in which the repository throttle sends a whitespace (' ') to waiting clients to ensure that they won't run into a read timeout.

`com.knime.server.repository.throttling.min_level=<value, e.g. 4, 8>` **[RT]**

> The minimum number of repository levels (query parameter `level=X` or `deep=true`) that have to be requested for a client to go through the throttle. Clients in the throttle may have to wait until their request is handled until free resources are available. Requests that fetch fewer levels are handled immediately.

`com.knime.server.repository.update_recommendations_at=<time>` **[RT]**

> Defines a time during the day (in ISO format, i.e. 24h notation, e.g. `21:15`) at which the node recommendations for the workflow coach are updated based on the current workflow repository contents. Default is undefined which means that no node recommendations will be computed and provided by the server.

`com.knime.server.server_admin_groups=<group>,<group>,···` **[RT]**

> Specifies the admin group(s). Users belonging to at least one of these groups are considered KNIME Server admins (not Tomcat server admins). Default is no admin groups.

> Env: `KNIME_SERVER_ADMIN_GROUPS=<group>,<group>,···`

`com.knime.server.server_admin_users=<user1>,<user2>,···` **[RT]**

> Specifies the user(s) that are KNIME Server admins (not Tomcat admins). Default is no users.

`com.knime.server.user_directories.directory_location=<location>` **[ST]**

> Specifies the base directory in which user directories shall be created on first login. When the base directory is created its `<owner>` is set to the one defined with `com.knime.server.user_directories.parent_directory_owner`. Also all non existing directories under `<location>` will be created and their owner set to `<owner>`. The permissions of the created directories are: `owner: rwx, world: r--`. If left empty no user directories will be created and all `com.knime.server.user_directories` options will be ignored. Note that only logins via the KNIME Analytics Platform will cause a user directory to be created.

`com.knime.server.user_directories.parent_directory_owner=<owner>` **[ST]**

> Specifies the owner of the base directory created at `<location>` (see `com.knime.server.user_directories.directory_location`). If left empty the default value `knimeadmin` will be used.

`com.knime.server.user_directories.owner_permissions=<permission>` **[ST]**

> Specifies the permissions of the owners (users themselves) for their created user directories. The defined permissions have to be in a block of 3 characters (r,w,x,-), e.g. `rwx` or `r-x`. If left empty the default value `rwx` is used.

`com.knime.server.user_directories.inherit_permissions=<true|false>` **[ST]**

> Specifies if the permissions of the created user directories shall be inherited from their parent directory.
> If left empty the default value `false` is used.

`com.knime.server.user_directories.groups=<group1>:<permission1>,<group2>:<permission2>,…` **[ST]**

> Specifies the permissions of groups for the created user directories. The defined permissions have to be in a block of 3 characters (r,w,x,-), e.g. `rwx` or `r-x`. If left empty no group permissions are set.

`com.knime.server.user_directories.users=<user1>:<permission1>,<user2>:<permission2>,…` **[ST]**

> Specifies the permissions of users for the created user directories. The defined permissions have to be in a block of 3 characters (r,w,x,-), e.g. `rwx` or `r-x`. If left empty no user permissions are set.

`com.knime.server.user_directories.world_permissions=<permission>` **[ST]**

> Specifies the permissions of others for the created user directories. The defined permissions have to be in a block of 3 characters (r,w,x,-), e.g. `rwx` or `r-x`. If left empty the default value `r--` is used.

`com.knime.server.action.callworkflow.enable_discard_checkboxes=<true|false>` **[RT]**

> Specifies if the options `Discard Workflow Job after successful Execution` and `Discard Workflow Job after failed Execution` for call workflow actions are enabled in KNIME Analytics Platform. This option only works for KNIME Analytics Platform 4.3 or higher. The default value is `true`.

`com.knime.server.action.callworkflow.force_discard_on_failure=<true|false>` **[RT]**

> Specifies the default value for `Discard Workflow Job after failed Execution` for call workflow action in KNIME Analytics Platform. This option only works for KNIME Analytics Platform 4.3 or higher. The default value is `true`.

`com.knime.server.action.callworkflow.force_discard_on_success=<true|false>` **[RT]**

> Specifies the default value for `Discard Workflow Job after successful Execution` for call workflow action in KNIME Analytics Platform. This option only works for KNIME Analytics Platform 4.3 or higher. The default value is `true`.

`com.knime.server.action.job.enable_discard_checkboxes=<true|false>` **[RT]**

Specifies if the options `Discard Workflow Job after successful Execution` and `Discard Workflow Job after failed Execution` for job execution are enabled in KNIME Analytics Platform. This option only works for KNIME Analytics Platform 4.3 or higher. The default value is `true`.

`com.knime.server.action.job.force_discard_on_failure=<true|false>` **[RT]**

Specifies the default value for `Discard Workflow Job after failed Execution` for job execution in KNIME Analytics Platform. This option only works for KNIME Analytics Platform 4.3 or higher. The default value is `false`.

`com.knime.server.action.job.force_discard_on_success=<true|false>` **[RT]**

Specifies the default value for `Discard Workflow Job after successful Execution` for job execution in KNIME Analytics Platform. This option only works for KNIME Analytics Platform 4.3 or higher. The default value is `false`.

`com.knime.server.action.upload.force_reset=<true|false>` **[RT]**

Specifies if all workflows shall be reset before being uploaded. This only works for workflows that are uploaded in the KNIME Analytics Platform 4.2 or higher. If left empty the default value `false` is used. The user can only change the reset behavior manually if `/instance/org.knime.workbench.explorer.view/action.upload.enable_reset_checkbox` is set to `true`, otherwise the behavior cannot be changed by the user.

`com.knime.server.action.upload.enable_reset_checkbox=<true|false>` **[RT]**

If set to `true` together with `com.knime.server.action.upload.force_reset` the user has the option to change the reset behavior in the `Upload to Server or Hub` dialog. This only works for workflows that are uploaded in the KNIME Analytics Platform 4.2 or higher. If left empty the default value `false` is used.

`com.knime.server.action.snapshot.force_creation=<true|false>` **[RT]**

Specifies if a snapshot shall always be created when overwriting a workflow or file. This only works when overwriting workflows or files in the KNIME Analytics Platform 4.2 or higher. If left empty the default value `false` is used.

---

com.knime.server.action.schedule.force_skip_execution=<true|false> [RT]

Specifies the default value for `Skip execution if previous job is still running` for scheduled job execution in KNIME Analytics Platform. This only works for KNIME Analytics Platform 4.5 or higher. The default value is `false`.

---

com.knime.server.action.schedule.enable_skip_execution_checkbox=<true|false> [RT]

Specifies if the option `Skip execution if previous job is still running` for scheduled job execution is enabled in KNIME Analytics Platform. This only works for KNIME Analytics Platform 4.5 or higher. The default value is `true`.

---

com.knime.server.workflow_authentication_groups=<group1>,<group2>,⋯ [RT]

Defines the groups that are allowed to create workflow access tokens.

Env: `KNIME_SERVER_WORKFLOW_AUTHENTICATION_GROUPS=<group>,<group>,⋯`

---

com.knime.server.workflow_authentication_users=<user1>,<user2>,⋯ [RT]

Defines the users that are allowed to create workflow access tokens.

Env: `KNIME_SERVER_WORKFLOW_AUTHENTICATION_USERS=<group>,<group>,⋯`

> **i** Users who should be allowed to create access tokens have to be explicitly added to `com.knime.server.workflow_authentication_users`. Note that this is also true for admin users. I.e., even admin users will **not** be able to create authentication tokens unless they are added to that list. By default, this list is empty, i.e., no one can create tokens.

---

com.knime.server.auth.min.password.length=<value> [RT]

Specifies the minimum length that is required for user passwords. The `default value is `-1`, which means no minimum password length. This configuration option is only relevant when using the internal user database.

---

com.knime.server.auth.complex.passwords=<true|false> [RT]

Specifies if complex passwords should be enforced. A complex password contains at least one lower-case character, one upper-case character, one number, and a non-alphanumeric character. The default value is `false`. This configuration option is only relevant when using the internal user database.

In KNIME Analytics Platform, these options are supported by KNIME Server: add them to the `knime.ini` file, after the `-vmargs` line, each in a separate line.

```
-Dcom.knime.server.server_address=<KNIME server>
```

Sets the `<KNIME server>` as the default Workflow Server in the client view.

```
com.knime.licensedir=<directory>
```

Allows to change the location of the license directory.
Default is the installation directory.

Default mount ID

KNIME supports mount point relative URLs using the `knime` protocol (see the KNIME Explorer section in the KNIME Workbench Guide for more details). Using this feature with KNIME Server requires both the workflow author and their collaborator to use the shared Mount IDs. With this in mind, you can now set a common name (Mount ID) for the Server to all users.

The default name for your server can be specified in the configuration file:

```
com.knime.server.default_mount_id=<server name>
```

> ℹ️ Please note that a valid Mount ID contains only characters a-Z, A-Z, '.' or '-'. It must start with a character and not end with a dot nor hyphen. Additionally, Mount IDs starting with `knime.` are reserved for internal use.

Blacklisting nodes

You might want to prevent the usage of certain nodes on the Executor of KNIME Server. While you can decide, which extensions you install for the Executor there might be nodes in the basic installation of KNIME Analytics Platform or in a required extension that shouldn't be used.

The configuration option

```
com.knime.server.executor.blacklisted_nodes=<node>,<node>,...
```

allows you to define a list of nodes that should be blocked by the Executor. This list also supports wildcards (*). If a workflow contains a blacklisted node the Executor will throw an error and abort loading the workflow.

To blacklist a node you have to provide the full name of the node factory. The easiest way to determine the factory names of the nodes you want to block is to create a workflow with all nodes that should be blacklisted. After saving the workflow you are able to access the `settings.xml` of each node under `<knime-workspace>/<workflow>/<node>/settings.xml`. The factory name can be found in the entry with key `"factory"`.

The following shows an example on how to block the Java Snippet nodes. The factory information for the Java Snippet node is

```
<entry key="factory" type="xstring"
value="org.knime.base.node.jsnippet.JavaSnippetNodeFactory"/>
```

To block the Java Snippet node we simply provide the value (without the quotes)

```
com.knime.server.executor.blacklisted_nodes=org.knime.base.node.jsnippet.JavaSnippetNode
Factory
```

The factory names for Java Snippet (simple), Java Snippet Row Splitter, and Java Snippet Row Filter are

```
org.knime.ext.sun.nodes.script.JavaScriptingNodeFactory
org.knime.ext.sun.nodes.script.node.rowsplitter.JavaRowSplitterNodeFactory
org.knime.ext.sun.nodes.script.node.rowfilter.JavaRowFilterNodeFactory
```

Since they all share the same prefix, we append n factory name making use of wildcards:

```
com.knime.server.executor.blacklisted_nodes=org.knime.base.node.jsnippet.JavaSnippetNode
Factory,org.knime.ext.sun.nodes.script.*Java*
```

While users are still able to upload workflows containing these nodes, the Executor won't load a workflow containing any of them.

KNIME Executor job handling

Job swapping

Jobs that are inactive for a period of time may be swapped to disc and removed from the Executor to free memory or Executor instances. A job is inactive if it is either fully executed or waiting for user input (on the KNIME WebPortal). If needed, it will be retrieved from disk automatically.

The configuration option

```
com.knime.server.job.max_time_in_memory=<duration with unit, e.g. 60m, 36h, or 2d>
```

controls the period of inactivity allowed before a job will be swapped to disk (default = `60m`). If you specify a negative number this feature is disabled and inactive jobs stay in memory until they are discarded.

> ℹ️ There are certain flows that will not be restored in the exact same state that it was in, before it got swapped out. For example, if a flow gets swapped with a loop partially executed, this loop iteration will be reset and the loop execution is restarted.

Job auto-discard

There is an additional threshold for inactivity of a job after which it may be discarded automatically. A discarded job due to inactivity cannot be recovered. The time threshold for a job to be automatically discarded is controlled by setting

```
com.knime.server.job.max_lifetime=<duration with unit, e.g. 60m, 36h, or 2d>
```

The default value (if the option is not set) is `7d`.

Managing User and Consumer Access

It is possible to restrict which groups (or which individual users) are eligible to log in as either users or consumers. In this context, a user is someone who logs in from a KNIME Analytics Platform client to e.g. upload workflows, set schedules, or adjust permissions. On the other hand, a consumer is someone who can only execute workflows from either the KNIME WebPortal or via the KNIME Server REST API.

In order to control who is allowed to log in as either user or consumer, the following settings need to be adjusted via the administration portal or `knime-server.config`:

`com.knime.server.login.allowed_groups`: This setting has to include **all** groups that should be allowed to login to KNIME Server, regardless of whether they are users or consumers.

`com.knime.server.login.consumer.allowed_groups`: List of groups that should be allowed to use the WebPortal or REST API to execute workflows.

`com.knime.server.login.user.allowed_groups`: List of groups that should be allowed to connect to KNIME Server from a KNIME Analytics Platform client.

Usage Example

`com.knime.server.login.allowed_groups=marketing,research,analysts`

`com.knime.server.login.consumer.allowed_groups=marketing,research,analysts`

`com.knime.server.login.user.allowed_groups=research`

In the above example, we first restrict general access to KNIME Server to individuals in the groups `marketing`, `research`, and `analysts`. All individuals who are not in any of these groups won't be able to access KNIME Server at all. Next, we allow all three groups to login as consumers via WebPortal or REST API. Finally, we define that only individuals in the group `research` should be able to log in as users from a KNIME Analytics Platform client.

> **i** By default, these settings are left empty, meaning that as long as users are generally able to login to your KNIME Server (e.g. because they are in the allowed AD groups within your organization), they can log in as either users or consumers. Since the number of available user licenses is typically lower than the number of consumers, it is recommended to restrict user access following the above example.

Executor Preferences

If the KNIME Executor requires certain preferences (e.g. database drivers or path to Python environment), you need to provide a preference files that the Executor(s) can retrieve from the application server.

To get a template of the preferences:

1. Start KNIME (with an arbitrary workspace).

2. Set all preferences via "File" → "Preferences" and export the preferences via "File" → "Export Preferences". This step can also be performed on a client computer but make

sure that any paths you set in the preferences are also valid on the server.

Open the exported preferences and insert the relevant lines into:

`<knime-server-repository>/config/client-profiles/executor/executor.epf`

Note: Make sure to specify the paths of all database drivers in the new preference page, in order to be able to execute workflows with database nodes. The page is available in the *KNIME → Database Drivers* category of the preferences.

> **i** It is recommended to only copy over the settings you will actually use on the Executor, like database drivers or Python preferences. The full preferences export is likely to contain e.g. host-specific paths that are not valid on the target system.

We have bundled a file called `executor.epf` into the `<knime-server-repository>/config/client-profiles/executor` folder. In order for those preferences to be used, you must either adjust the service of the Executor or the `knime.ini` file.

Setting the preferences in the service (preferred way)

Linux

1. Install the Executor service as described in the KNIME Server Installation Guide.
2. After installing the service run

   ```
   systemctl edit knime-executor.service
   ```

3. Adjust the environment variable `KNIME_EXECUTOR_PROFILES`

   ```
   Environment='KNIME_EXECUTOR_PROFILES=-profileLocation
   http://127.0.0.1:8080/<WebPortal Context ROOT, most likely
   "knime">/rest/v4/profiles/contents -profileList executor'
   ```

   Note, that the command has to be surrounded by single quotes.

4. Save and exit the editor
5. Start the executor service

In case you upgraded the Executor from a version older than 4.13.0 copy the folder

```
<knime_executor>/systemd/usr
```

to the root of your file system, overwriting the previously copied files and run

```
systemctl daemon-reload
```

Windows

1. Edit the `install-executor-as-service.bat` and change

   ```
   REM SET "KNIME_EXECUTOR_PROFILES=-profileLocation http://127.0.0.1:8080/<WebPortal
   Context ROOT, most likely "knime">/rest/v4/profiles/contents -profileList
   executor"
   ```

   to

   ```
   SET "KNIME_EXECUTOR_PROFILES=-profileLocation http://127.0.0.1:8080/<WebPortal
   Context ROOT, most likely "knime">/rest/v4/profiles/contents -profileList
   executor"
   ```

2. Save the file

3. Execute the `install-executor-as-service.bat`.

Setting the preferences in the `knime.ini` file

Open the `knime.ini` file of the executor and insert

```
-profileLocation
http://127.0.0.1:8080/<WebPortal Context ROOT, most likely
"knime">/rest/v4/profiles/contents
-profileList
executor
```

before the line containing `-vmargs`. Note, that these are KNIME specific settings and may cause problems when trying to install extensions via command line.

Adding Executor preferences for headless Executors

In order to be able to execute workflows that contain database nodes that use custom or proprietary JDBC driver files on KNIME Server, the `executor.epf` file must contain the path to

the JDBC jar file, or the folder containing the JDBC driver. This may be specified in the KNIME Analytics Platform (Executor) GUI and the `executor.epf` file exported as described in the above section. This is the recommended route for systems that have graphical access to the KNIME Analytics Platform (Executor).

Some systems do not have graphical access to the KNIME Analytics Platform (Executor) GUI. In that case the `executor.epf` can be manually created, or created on an external machine and copied into location on the server. The relevant lines that must be contained in the `executor.epf` file are:

```
file_export_version=3.0
\!/=
/instance/org.knime.workbench.core/database_drivers=/path/to/driver.jar;/path/to/driver-
folder
/instance/org.knime.workbench.core/database_timeout=60
```

Note that `driver.jar` may also reference a folder in some cases (e.g. MS SQL Server and Simba Hive drivers).

> **i** If you are using distributed KNIME Executors, please see the Server-managed Customization Profiles section of the KNIME Database Extension Guide for how to distribute JDBC drivers.

Local file system access by KNIME workflows

KNIME nodes have been revised to use a new shared framework for file access.

When executing on KNIME Server, a preference controls whether those nodes may access the local file system of the KNIME Server Executor or not. With the KNIME Server release 4.12, local file system access is **disallowed** by default (previously it was allowed).

Please note that the following configuration is **not recommended**.

To **allow** local file system access, you can add the following line to the customization profile used by your KNIME Server Executor(s):

```
/instance/org.knime.filehandling.core/allow_local_fs_access_on_server=true
```

This preference affects all nodes that are part of the revised file handling framework. Old nodes that have not yet been ported to the new framework are not affected by this setting.

knime.ini file

You might want to tweak certain settings of this KNIME instance, e.g. the amount of available memory or set system properties that are required by some extensions. This can be changed directly in the `knime.ini` in the KNIME Executor installation folder.

KNIME Server will read the `knime.ini` file next to the KNIME executable and create a custom ini file for every Executor that is started. However, if you use a shell script that prepares an environment the server may not be able to find the ini file if this start script is in a different folder. In this case the `knime.ini` file must be copied to `<knime-server-repository>/config/knime.ini`. If this file exists, the server will read it instead of searching for a `knime.ini` next to the executable or start script.

Log files

There are several log files that could be inspected in case of unexpected behavior:

Tomcat server log

Location: `<apache-tomcat>/logs/catalina.yyyy-mm-dd.log`
This file contains all general Tomcat server messages, such as startup and shutdown. If Tomcat does not start or the KNIME Server application cannot be deployed, you should first look into this file.

Location: `<apache-tomcat>/logs/localhost.yyyy-mm-dd.log`
This file contains all messages related to the KNIME Server operation. It does <u>not</u> include messages from the KNIME Executor!

For new installations these files are kept for 90 days before being removed. The default behavior can be changed by editing the `<apache-tomcat>/conf/logging.properties` file and amending any entries with:

```
1catalina.org.apache.juli.FileHandler.maxDays = 90
```

Job tracing

Since KNIME Server 4.12 it's possible to enable a job trace log which records important operations on any job (loading, execution, discarding). The job trace log is disabled by default and can be enabled via `<apache-tomcat>/conf/logging.properties`. Fresh installations of KNIME Server 4.12 already contain the necessary configuration in that file, you only have to

uncomment the respective lines. For existing installations, make the following modifications to `logging.properties`:

1. Append `, 6jobtracer.org.apache.juli.FileHandler` to the line starting with `handlers` at the top of the file

2. Add the following section at the bottom of the file:

```
com.knime.enterprise.server.jobs.Tracer.handlers =
6jobtracer.org.apache.juli.FileHandler
6jobtracer.org.apache.juli.FileHandler.level = FINE
6jobtracer.org.apache.juli.FileHandler.directory = ${catalina.base}/logs
6jobtracer.org.apache.juli.FileHandler.prefix = jobs.
6jobtracer.org.apache.juli.FileHandler.format=[%1$tF %1$tT] [%4$-7s] %5$s %n
com.knime.enterprise.server.jobs.Tracer.level = FINE
```

After modifying the file you have to restart KNIME Server. The job traces can then be found in log files starting with `jobs.` in the usual Tomcat log directory. Every line contains one event for a job as a JSON object which can be postprocessed by e.g. a KNIME workflow. The format of the JSON object is self-explanatory.

KNIME executor log

Location: `<executor-workspace>/.metadata/knime/knime.log`
The *executor-workspace* is usually in the home directory of the operating system user that runs the executor process and is called `knime-workspace`. If you provided a custom workspace using the `-data` argument when starting the executor you can find it there.

If you are still using deprecated RMI executors, the *executor-workspace* is `<knime-server-repository>/runtime/runtime_knime-rmi_<suffix>`.

This file contains messages from the KNIME Executor that is used to execute workflows on the server (for manually triggered execution, scheduled jobs, and also for generated reports, if KNIME Report Server is installed)

The executor's log file rotates every 10MB by default. If you want to increase the log file size (to 100MB for example), you have to append the following line at the end of the executor's `knime.ini`:

```
-Dknime.logfile.maxsize=100m
```

Also useful in some cases is the Eclipse log file `<executor-workspace>/.metadata/.log`

KNIME Analytics Platform (client) log

Location: `<local workspace>/.metadata/knime/knime.log`
This file contains messages of the client KNIME application. Messages occurring during server communications are logged there. The Eclipse log of this application is in `<local workspace>/.metadata/.log`

Embedded Data Apps administration

It is possible to generate access tokens that allow users to run Data Apps, i.e. KNIME WebPortal applications, without the need to log in explicitly to the KNIME WebPortal. Authentication then happens via the token, which is used as query parameter in the WebPortal URL. This allows the user to embed Data Apps in other websites, reaching a wide audience of consumers without the need to create individual users for all of them. Since this feature assumes that consumers are not counted individually, it is only available for KNIME Server Large licenses, where consumers are unlimited.

Allow user to create access tokens

Users who should be allowed to create access tokens have to be explicitly added to the list of authorized users in the in `knime-server.config` by using the configuration option `com.knime.server.workflow_authentication_users`. Note that this is also true for admin users. I.e., even admin users will **not** be able to create access tokens unless they are added to that list. By default, this list is empty, i.e., no one can create tokens.

Embedded Data Apps in iframe

In order to create an Embedded Data App in an iframe, the Server's cookie policy must be changed to work with the latest browsers. The server cookie's `SameSite` attribute must be set to `none` and the Server must be accessed using https.

To set the Server cookie's `SameSite` attribute to `none`, the `CookieProcessor` has to be defined in `<apache-tomcat>/conf/Catalina/localhost/knime.xml` by adding the following to the defined context:
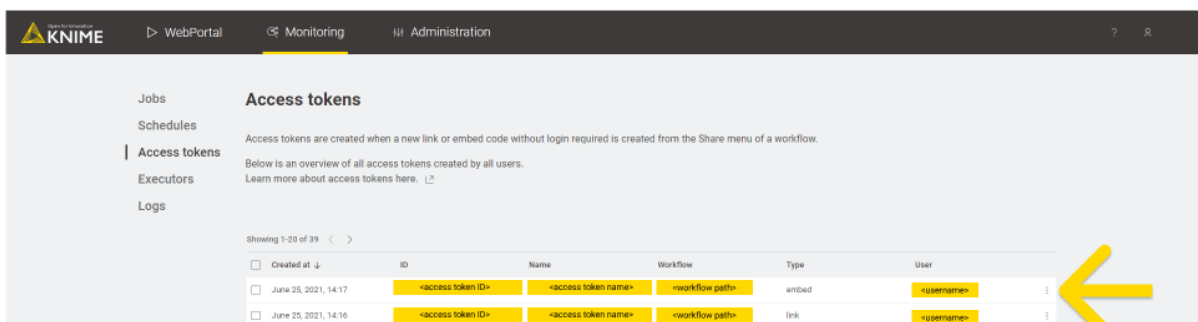
```
<Context>
    ...
    <CookieProcessor className="com.knime.enterprise.tomcat.cookie.KnimeCookieProcessor"
sameSiteCookies="none" />
    ...
</Context>
```

This ensures that the session cookie is sent along when the WebPortal is accessed through an iframe.

Access tokens list

Via the Monitoring portal you can list all the access tokens that have been created by users on KNIME Server, under *Monitoring → Access tokens*. You can delete access tokens by selecting the corresponding three dots and clicking *Delete*.

Executor Watchdog

KNIME Server is able to detect vanished executors in case they have loaded jobs using the option `com.knime.server.executor.watchdog.interval` (see here). To detect a vanished executor the Executor Watchdog will request status updates on a regular basis. In case the latest response of the executor is older than the `max_absence_time` it is considered to be vanished and all its loaded jobs are marked as vanished, i.e. their state is set to `VANISHED`. A vanished job cannot be loaded into another executor. Note that the job state could still change in the unlikely event that a new job status update is sent to the server after the job has been marked as vanished.

Email notification

KNIME Server allows users to be notified by email when a workflow finishes executing. The emails are sent from a single email address which can be configured as part of the web application's mail configuration. If you don't want to enable the email notification feature, no email account is required. You can always change the configuration and enter the account details later.

Setting up the server's email resource

The email configuration is defined in the `<knime-server-repository>/config/knime-server.config` which you can change via the administration portal. The installer has already created this file. In order to change the email configuration, you have to modify or add configuration properties. The table below shows the list of supported parameters (see also the JavaMail API documentation).

Name	Value
`mail.smtp.from`	Address from which all mails are sent
`mail.smtp.host`	SMTP server, required
`mail.smtp.port`	SMTP port, default `25`
`mail.smtp.auth`	Set to `true` if the mail server requires authentication; optional
`mail.smtp.user`	Username for SMTP authentication; optional
`mail.password`	Password for SMTP authentication; optional
`mail.smtp.starttls.enable`	If `true`, enables the use of the STARTTLS command (if supported by the server) to switch the connection to a TLS-protected connection before issuing any login commands. Defaults to false.
`mail.smtp.ssl.enable`	If set to true, use SSL to connect and use the SSL port by default. Defaults to `false`.

> **i** With KNIME Server 4.12 the email configuration moved from the `knime.xml` to
> the `<knime-server-repository>/config/knime-server.config`, so any existing
> custom configuration must be moved if you are updating from a KNIME Server
> version 4.11 and older to version 4.12 and newer. The email configuration
> settings in the `knime.xml` file will be ignored.

If you do not intend to use the email notification service (available in the KNIME WebPortal
for finished workflow jobs), you can skip this step.

Note that the mail configuration file contains the password in plain text. Therefore, you
should make sure that the file has restrictive permissions.

User authentication

As described briefly in the Server architecture section it is possible to use any of the authentication methods available to Tomcat in order to manage user authentication. By default the KNIME Server installer configures a database (H2) based authentication method. Using this method it is possible for admin users to add/remove users/groups via the administration portal. Other users may change their password using this technique.

For enterprise applications, use of LDAP authentication is recommended, and user/group management is handled in Active Directory/LDAP itself.

In all cases the relevant configuration information is contained in the

```
`<Realm className="org.apache.catalina.realm.LockOutRealm">`
```

tag in `<apache-tomcat>/conf/server.xml`.

The default configuration uses a `CombinedRealm` which allows multiple authentication methods to be used together. Examples for each of database, file and LDAP authentication are contained within the default installation. Configuration of all three authentication methods are described briefly in the following sections. In all cases the Tomcat documentation should be considered the authoritative information source.

LDAP authentication

LDAP authentication is the recommended authentication in any case where an LDAP server is available. If you are familiar with your LDAP configuration you can add the details during installation time, or edit the `server.xml` file post installation. If you are unfamiliar with your LDAP settings, you may need to contact your LDAP administrator, or use the configuration details for any other Tomcat based system in your organization. Please refer to the KNIME Server Advanced Setup Guide for details on setting up LDAP.

Connecting to an SSL secured LDAP server

In case you are using encrypted LDAP authentication and your LDAP server is using a self-signed certificate, Tomcat will refuse it. In this case you need to add the LDAP server's certificate to the global Java keystore, which is located in `<jre-folder>/lib/security/cacerts`:

```
keytool -import -v -noprompt -trustcacerts -file \
      <server certificate> -keystore <jre>/lib/security/cacerts \
      -storepass changeit
```

Alternatively you can copy the `cacerts` file, add your server certificate, and add the following two system properties to `<apache-tomcat>/conf/catalina.properties`:

```
javax.net.ssl.trustStore=<copied keystore>
javax.net.ssl.keyStorePassword=changeit
```

Single-sign-on with LDAP and Kerberos

It is possible to use Kerberos in combination with LDAP for Single-Sign-On for authentication with KNIME Server.

This is an advanced topic and is covered in the KNIME Server Advanced Setup Guide.

Token-based authentication

KNIME Server also allows authentication by JWT (JSON Web Tokens) that have previously been issued by the server. The REST endpoint `/rest/v4/auth/jwt` can be used to acquire such a JWT for the currently logged in user. Subsequent requests need to carry the token in the `Authorization` header as follows:

```
Authorization: Bearer xxx.yyy.zzz
```

where `xxx.yyy.zzz` is the JWT. Token-based authentication is enabled by default and cannot be disabled. However, you can restrict the maximum lifetime of JWTs issued by the server via the server configuration option `com.knime.server.login.jwt-lifetime`, see section KNIME Server configuration file options.

The OpenAPI documentation for the REST API which can be found at: `https://<hostname>/knime/rest/doc/index.html#/Session` should be considered the definitive documentation for this feature.

Large number of users in a group

Since the JWT includes the group membership for the user, this can get very large in some cases. JWTs with more than 30 groups and that are larger than 2kB are now compressed. If

they are still larger than 7kB a warning is logged with hints how to resolve potential problems.

One solution is to increase the maximum HTTP header size in Tomcat by adding the attribute maxHttpHeaderSize="32768" to all defined Connectors in the server.xml (the default is 8kB). In case Tomcat is running behind a proxy, the limit may need to be increased there, too. In case of Apache it's the global setting LimitRequestFieldSize 32768.

Database-based authentication

Database-based authentication is recommended to be used by small workgroups who do not have access to an LDAP system, or larger organizations in the process of trialing KNIME Server. If using the previously described H2 database it is possible to use the administration portal to manage users and groups. It is possible to use other SQL databases e.g. PostgreSQL to store user/group information, although in this case it is not possible to use the administration portal to manage users/groups, management must be done in the database directly.

For default installations this authentication method is enabled within the `server.xml` file. No configuration changes are required. In order to add/remove users, or create/remove groups the administration pages of the WebPortal can be used. The administration pages can be located by logging into the WebPortal as the admin user, see section administration portal for more details.

Batch insert/update of usernames and roles is possible using the admin functionality of the KNIME Server REST API. This is described in more detail in the section RESTful webservice interface. A KNIME Workflow is available in the distributed KNIME Server installation package that can perform this functionality.

File-based authentication

For KNIME Server versions 4.3 or older the default configuration used a file-based authentication which we describe for legacy purposes. It is now recommended to use either database-based or LDAP authentication. The advantages of each are described in the corresponding sections above and below.

The XML file `<apache-tomcat>/conf/tomcat-users.xml` contains examples on how to define users and groups (roles). Edit this file and follow the descriptions. By default this user configuration file contains the passwords in plain text. Encrypted storage of passwords is described in the Tomcat documentation.

Configuring a license server

Since version 4.3 KNIME Server can distribute licenses for extensions to the KNIME Analytics Platform to clients. In order to use the license server functionality, you require a master license.

The master license file(s) should be copied into the `licenses` folder of the server repository (next to the server's license). The server will automatically pick up the license and offer them to clients. For configuring the client, see the section about "Retrieving client licenses" in the KNIME Explorer User Guide.

Client licenses distributed by the server are stored locally on the client and are tied to the user's operating system name (not the server login!) and its KNIME Analytics Platform installation and/or the computer. They are valid for five days by default which means that the respective extensions can be used for a limited time even if the user doesn't have access to the license server.

If the user limit for a license has been reached, no further licenses will be issued to clients until at least one of the issued licenses expires. The administrator will also get a notification email in this case (if their email notification is configured, see previous section Email notification).

License renewal

If the server is not behaving as expected due to license issues, please contact KNIME by sending an email to support@knime.com or to your dedicated KNIME support specialist.

If the license file is missing or is invalid a message is logged to the server's log file during server start up. KNIME clients are not able to connect to the server without a valid server license. Login fails with a message "No license for server found".

If the KNIME Server license has expired connecting clients fail with the message "License for enterprise server has expired on …". Please contact KNIME to renew your license.

If more users than are licensed attempt to login to the WebPortal, some users will see the message: "Maximum number of WebPortal users exceeded. The current server license allow at most <number of licensed users> WebPortal users.". In this case you will need to email KNIME at support@knime.com to discuss options to increase the number of licensed users.

After you receive a new license file, you can upload it via the administration portal from the KNIME WebPortal under the License section. Click *Change License* button and a widget will open where you can upload or drag and drop your license file.

You can also upload a new license file manually following these steps:

1. Remove the old expired license from the `<knime-server-repository>/licenses` folder. In case there are multiple license files in this folder, find the one containing a line with

```
"name" = "KNIME Server"
```

and the `"expiration date"` set to a date in the past. The license file is a plain text file
and can be read in any text editor.

2. Store the new license file in the license folder with the same owner and the same
   permissions as the old file. The new license is applied immediately; a server restart is
   not necessary.

Backup and recovery

The following files and/or directories need to be backed up:

- The full server repository folder, except the `temp` folder

- The full Tomcat folder

- In case you installed your own molecule sketcher for the KNIME WebPortal (see above), also backup this folder.

A backup can be performed while the server is running but it's not guaranteed that a consistent state will be copied as jobs and the workflow repository may change while you are copying files.

In order to restore a backup copy the files and directories back to their original places and restart the server. You may also restore to different location but make sure to adjust the paths in the start script, the repository location in the context configuration file, and paths in the server configuration.

KNIME Executor administration

As specified in the KNIME Server Installation Guide for single node Server installations to a target machine with access to the internet you can perform the KNIME Executor installation alongside the KNIME Server installation process.

If this is not the case you can perform the installation of KNIME Executor manually. To do so please follow the steps in the KNIME Server Installation Guide.

Installing additional extensions

The easiest way to install additional extensions into the Executor (e.g. Community Extensions or commercial 3rd party extensions) is to start the Executor in GUI mode and install the extensions as usual. In case you don't have graphical access to the server you can also install additional extensions without a GUI. The standard `knime` executable can be started with a different application that allows changing the installation itself:

```
./knime -application org.eclipse.equinox.p2.director -nosplash
  -consolelog -r _<list-of-update-sites>_ -i _<list-of-features>_ -d _<knime-
installation-folder>_
```

Adjust the following parameters to your needs:

- `<list-of-update-sites>`: a comma-separated list of remote or local update sites to use. ZIP files require a special syntax (note the single quotes around the argument). Example:

```
-r 'https://update.knime.org/analytics-
platform/5.4,jar:file:/tmp/org.knime.update.analytics-platform_5.4.0.zip!/'
```

> **i** Some extensions, particularly from community update sites, have dependencies to other update sites. In those cases, it is necessary to list **all** relevant update sites in the installation command.

  - Adding the following four update sites should cover the vast majority of cases:
    - https://update.knime.com/analytics-platform/5.4
    - https://update.knime.com/community-contributions/5.4
    - https://update.knime.com/community-contributions/trusted/5.4
    - https://update.knime.com/partner/5.4

If you have limited internet access you can install extensions from a local ZIP file.

- You can download KNIME update sites as ZIP file at the following links:
    - KNIME Analytics Platform Update Site
    - KNIME Community Extensions
    - KNIME Partner Extensions

- `<list-of-features>`: a comma-separated list (spaces after commas are not supported) of features/extensions that should be installed. You can get the necessary identifiers by looking at *Help → About KNIME → Installation Details → Installed Software* in a KNIME instance that has the desired features installed. Take the identifiers from the "Id" column and make sure you don't omit the `.feature.group` at the end (see also screenshot on the next page). Example:

```
-i org.knime.product.desktop,org.knime.features.r.feature.group
```

You can get a list of all installed features with:

```
./knime -application org.eclipse.equinox.p2.director -nosplash \
  -consolelog -lir -d _<knime-installation-folder_
```

- `<knime-installation-folder>`: the folder into which KNIME Analytics Platform should be installed (or where it is already installed). Example:

```
-d /opt/knime/knime_5.4
```

Uninstalling additional extensions

It is also possible to uninstall extensions via command line.

You can do that by using `-u` in the command instead of `-i`.

```
./knime -application org.eclipse.equinox.p2.director -nosplash
  -consolelog -r _<list-of-update-sites>_ -u _<list-of-features>_ -d _<knime-
installation-folder>_
```

Enabling workflow execution

Sometimes workflow jobs running in the Executor want to access files on the server, e.g. via workflow-relative URLs or by a URL using the server's mount point ID. Since the Executor cannot authenticate itself to the server with the user's password (because it's generally not known by either the server or the Executor) a token is generated by the server when the workflow is started or scheduled. This token represents the user including its group membership *at the time it is created*. If group membership changes while the workflow job is still running or there are further scheduled executions, these changes will not be reflected in the workflow execution. Also if access has been revoked from the user completely, existing (scheduled) jobs can still access the server repository.

If the Executor is running on a different machine than the server, please pay attention to the following: the communication between server and Executor is partially performed via the REST interface, e.g. when a workflow requests files from the server repository. Therefore the Executor must know the server's address. The server tries to auto-detect its address and sends it to the Executor. However, if the server is running behind a proxy (e.g. Apache) or has a different external IP address than internally, auto-detection will give a wrong address and the Executor will not be able to reach the server. In this case you have to set the configuration option `com.knime.server.canonical-address` to the server's canonical address, e.g. `http://knime-server.behind.proxy/` (you do not need to provide the path to the server application). This address must be usable by the Executor.

KNIME Executors in distributed systems

Distributed KNIME Executors: Introduction

As part of a highly available architecture, KNIME Server 4.18 allows you to distribute execution of workflows over several Executors that can sit on separate hardware resources. This allows KNIME Server to scale workflow execution with increasing load because it is no longer bound to a single computer.

If you're planning to use the distributed KNIME Executors in production environments please get in touch with us directly for more information.

Installation, configuration, and operation is very similar to the single Executor setup. The server communicates with the Executors via a message queueing system (and HTTP(S)). We use RabbitMQ for this purpose, and it's recommended, although not required, to install that on a separate machine as part of a highly available architecture.
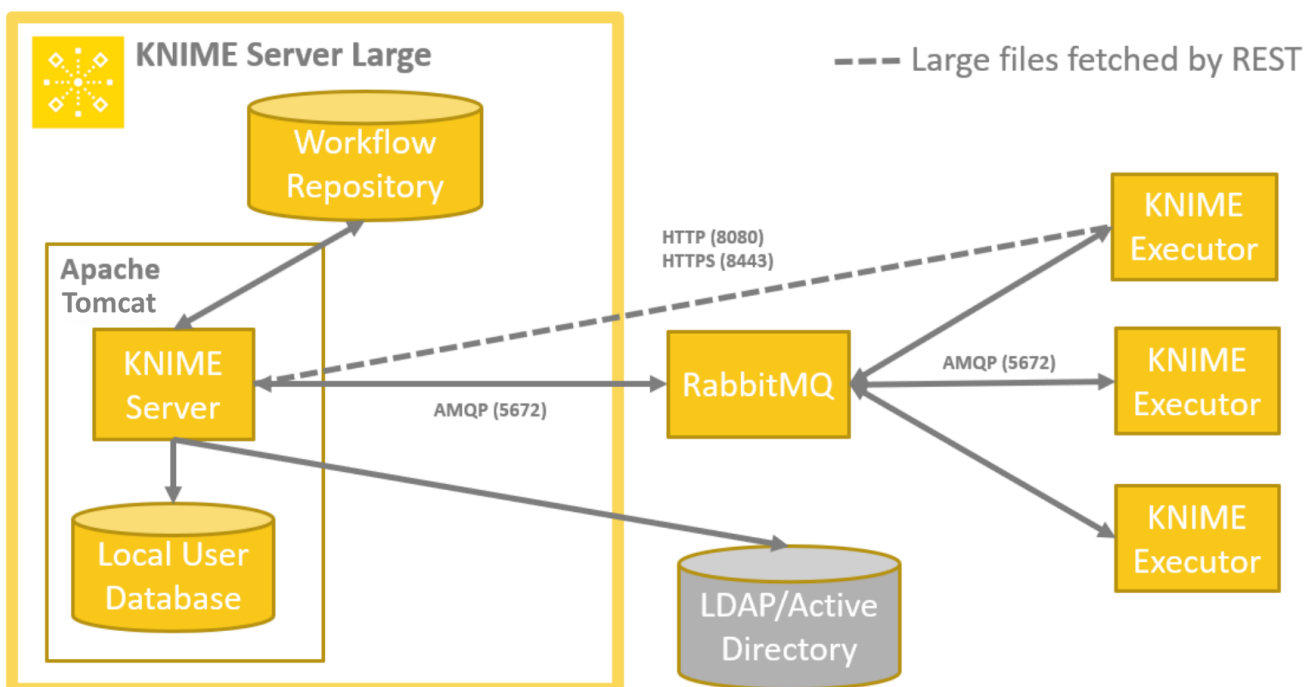
To install distributed KNIME Executors please follow the instructions provided in the Distributed KNIME Executors: Installation instructions section of the KNIME Server Installation Guide.

Distributed KNIME Executors: Administration and settings

Load throttling

If too many jobs are sent to KNIME Executors this may overload them and all jobs running on that Executor will suffer and potentially even fail if there aren't sufficient resources available any more (most notably memory). Therefore an Executor can reject new jobs based on its current load. By default an Executor will not accept new jobs any more if its memory usage is above 90% (Java heap memory) or the average system load is above 90% (averaged over 1-minute). These options can be set as property in the `knime.ini` file as well as by defining an environment variable (Env). The environment variable changes will only take effect after a restart of the KNIME Executor. If the environment variable for an option is set, the property in the 'knime.ini' file will be ignored.

```
-Dcom.knime.enterprise.executor.heapUsagePercentLimit=<value-in-percent e.g. 90>
```
The average Heap space usage of the executor JVM over one minute. Default 90 percent

Env: `KNIME_EXECUTOR_HEAP_USAGE_PERCENT_LIMIT=<value-in-percent e.g. 90>`

```
-Dcom.knime.enterprise.executor.cpuUsagePercentLimit=<value-in-percent e.g. 90>
```
The average CPU usage of the executor JVM over one minute. Default 90 percent.

Env: `KNIME_EXECUTOR_CPU_USAGE_PERCENT_LIMIT=<value-in-percent e.g. 90>`

Under certain circumstances the heap usage can be above the threshold (i.e. the executor will reject new jobs) but the memory is not actually being used any more and Java's garbage collector does not reclaim it. As a workaround we added a mechanism in KNIME Server 4.13 that explicitly invokes the garbage collector if the heap usage is above the threshold and the last garbage collection was more than a certain amount of time ago. This duration can be configured in the `knime.ini` or by an enviroment variable:

```
-Dcom.knime.enterprise.executor.delay_between_full_gcs=<duration with unit, e.g.
30s, 1m>
```
The time since the last full garbage collection that must elapse before the executor explicitly invokes another full garbage collection. The default value is *5m*.

Env: `KNIME_EXECUTOR_DELAY_BETWEEN_FULL_GCS=<duration with unit, e.g. 30s, 1m>`

You should not set this value too low because during garbage collection the executor may be completely unresponsive. Especially for very large heaps garbage collection may take quite

some time (up to several minutes).

If you want to disable this feature, set the value to a very large value such as *100d*.

Resource throttling

It is possible to restrict the number of cores/threads used by the Executor. In normal operation, you do not need to set this preference. Typically, the JVM will determine how many cores are available in the system (including identifying hyper-threaded cores as a 'core'), and the Executor will then set `knime.maxThreads=2*num_cores`.

In some cases, though, you may wish to restrict how many cores/threads the Executor can use. Examples of when this may be desired include when additional KNIME Executor cores on the machine must be reserved for another task, or in a local Docker setup where containers detect all cores available on a machine. Both of these configurations are typically not recommended, as it can be difficult to guarantee good resource sharing. Generally, it is better to run workloads on individual machines or in isolated pods using Kubernetes.

However, should you need to do so, you would use the following setting:

```
/instance/org.knime.workbench.core/knime.maxThreads=<maximum number of threads
 to use>
```
> This setting controls the number of threads that the KNIME Executor will use to process workflows, and must be added to one of the preferences (`.epf`) files used by the Executor. (For more information on Executor preferences, see Executor Preferences.)

Automated Scaling

Currently we allow automated scaling by monitoring Executor heap space and CPU usage. It is also possible to blend these metrics using custom logic to invent custom scaling metrics. In some cases it may also be desirable to allow jobs to stack up on the queue and use the 'queue depth' as a fourth metric type. In order to do so, it is necessary to edit the knime.ini of the Executors.

```
-Dcom.knime.enterprise.executor.allowNoExecutors=<true|false>
```

> <Experimental Setting> Specifies whether the last Executor accepting jobs is allowed to reject jobs. That will result in the behaviour that it is possible for jobs to pile-up on RabbitMQ. It may be necessary to increase the `com.knime.server.job.default_load_timeout` and the `com.knime.explorer.job.load_timeout` in the Analytics Platform to ensure sensible behaviour. The default is `false`, which emulates the behaviour before the setting was added.

When using an automatic scaling setup, jobs that are waiting for an Executor to start, might run into timeouts. The default wait time for a job to be loaded by an Executor can be increased by setting the `com.knime.server.job.default_load_timeout` option in the server configuration as described in section Server configuration files and options.

When starting jobs interactively using the Analytics Platform, the connection might also time out. The timeout can be increased by adding the following option to the knime.ini file of the KNIME Analytics Platform.

```
-Dcom.knime.explorer.job.load_timeout=<duration with unit, e.g. 60m, 36h, or 2d>
```

> Specifies the timeout to wait for the job to be loaded.
> The default duration is `5m`.

Generally, the timeout in the Analytics Platform should be higher than the timeout set in the KNIME server. This prevents the interactive session from running into read timeouts.

Reconnecting to message queue

In case the connection to the message queue gets lost (e.g. by restarting RabbitMQ), starting with KNIME Server 4.11 the Executor will try to reconnect to the message queue. The following option can be adjusted in the `knime.ini` file of the Executor:

```
-Dcom.knime.enterprise.executor.connection_retries=<number of retries>
```

> Specifies the number of retries that should be attempted to reconnect to the message queue. Between each attempt the Executor waits 10 seconds. For KNIME Server 4.12.3 and older the default value is set to `9` i.e. the Executor tries reconnecting for 90 seconds. For newer versions the default is to try reconnecting indefinitely. Note that this option can be also set via the environment variable `KNIME_EXECUTOR_CONNECTION_RETRIES`, which takes precedence over the system property set in the `knime.ini` file. For `number of retries` less than 0 the number of retries is infinite.

Setting Executor to draining state

It is possible to switch off an Executor without interfering with running jobs. In fact, you can set the Executor to a `draining` state, e.g. via the monitoring portal, or via REST API `PATCH` call (`PATCH https://<knime-server>/knime/rest/v4/admin/executors/:uuid`) with body `{"isDraining" : true}`.

This means that the Executor will not accept any new jobs, but any existing job will still be finished.

The Executor will stay set to `draining` state unless:

- The Executor is shut down. This can be done via a REST API `DELETE` call (`DELETE https://<knime-server>/knime/rest/v4/admin/executors/:uuid`).

  > **i** Please be aware that if you shut down a `draining` Executor before all the existing jobs are finished you will lose all progress of the jobs on the Executor. They will not get swapped to the Server before shutting down. You can check the status of the Executor via REST API.

- The Executor is set to `active` again via the monitoring portal, or via a REST API `PATCH` call
  (`PATCH https://<knime-server>/knime/rest/v4/admin/executors/:uuid`) with body `{"isDraining" : false}`.

Loading multiple jobs in parallel

Starting with KNIME Server 4.12.1 a single executor is able to load multiple jobs in parallel whereas in previous versions jobs were loaded one after the other. This allows a higher

throughput in case loading a job takes a while (e.g. due to large workflows).

```
-Dcom.knime.enterprise.executor.parallel-job-loads=<number of parallel loading
jobs>
```
> Specifies the maximum number of jobs that can be loaded in parallel by an executor. The default value is set to `3`.

Health Check

Starting with KNIME Server 4.13.0 the Executor contains a health check endpoint conforming with the Eclipse Microprofile Health. This endpoint can be accessed via GET requests and returns the current health status of the Executor. To activate the endpoint the following options have to be provided:

```
-Dcom.knime.enterprise.executor.healthcheck.port=<port number>
```
> Specifies the port number of the health check endpoint. If no port is defined then no endpoint will be available. Note that this option can be also set via the environment variable `KNIME_EXECUTOR_HEALTH_CHECK_PORT`, which takes precedence over the system property set in the `knime.ini` file.

```
-Dcom.knime.enterprise.executor.healthcheck.path=<path>
```
> Specifies the path of the health check endpoint. If no path is defined the default `/health` will be used. Note that this option can be also set via the environment variable `KNIME_EXECUTOR_HEALTH_CHECK_PATH`, which takes precedence over the system property set in the `knime.ini` file.

Once the Executor has started successfully the endpoint is reachable on the Executor's machine via the following endpoints:

```
http://localhost:<port>/<path>/live
```
> Returns the live state of the Executor.
> The Executor is live if it received any core tokens.

```
http://localhost:<port>/<path>/ready
```

> Returns the ready state of the Executor.
> The Executor is ready if it is accepting new jobs.

```
http://localhost:<port>/<path>
```

> Returns the health state of the Executor. The Executor is healthy if it received any
> core tokens. This endpoint combines the behavior of the two endpoints `/<path>/live`
> and `/<path>/ready` and is a fallback in case the other two endpoints cannot be used.

For each of the endpoints the following HTTP status codes are defined:

- `200`: if the status of the endpoint is `UP`
- `500`: if an error occurred while determining the state of the Executor
- `503`: if the status of the endpoint is `DOWN`

In addition to the status codes, a JSON object will be returned with the following schema:

```
{
  "type": "object",
  "properties": {
    "status": {
      "type": "string",
      "example" : "UP"
    },
    "checks": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "name": {
            "type": "string",
            "example" : "consumingMessages"
          },
          "status": {
            "type": "string",
            "example" : "UP"
          },
          "data": {
            "type": "object",
            "properties": {
              "info": {
                "type": "string",
                "example" : "Executor ready to accept new jobs"
              }
            }
          }
        }
      }
    }
  },
  "required": [
    "status",
    "checks"
  ],
  "additionalProperties": false
}
```

Swapping upon shutdown

Starting with KNIME Server 4.12.2 the Executor contains an option to define the time the Executor waits until a job is swapped during shutdown.

```
-Dcom.knime.enterprise.executor.swap-wait=<duration, e.g. 30s, 1m, 1h, 1d>
```

> Specifies the time the Executor will wait for a single job to be swapped. In case the job has not been swapped within the timeout the Executor will continue with the next job in the list. In case no other job has to be swapped the Executor will continue with its shutdown routine. The default value is `1m`. Note that this option can also be set via the environment variable `SHUTDOWN_SWAP_WAIT`, which takes precedence over the system property set in the `knime.ini` file.

Note: The total time the Executor will wait for all jobs to be swapped depends on the number of jobs currently loaded. Please ensure that the Executor is not forcefully shutdown in case the shutdown routine takes too long as it may result in uncleaned temporary files and lost jobs.

TLS connection to RabbitMQ

Starting with KNIME Server 4.13.0 we extended the support for TLS connections to RabbitMQ. While KNIME Server and Executor were able to establish a TLS connection in previous versions the number of options was limited, and all certificates provided by RabbitMQ were accepted by default. To enable TLS in RabbitMQ please follow the official documentation.

To connect KNIME Server and Executor to RabbitMQ via TLS you have to adjust the broker address settings `com.knime.enterprise.executor.msgq` and `-Dcom.knime.enterprise.executor.msgq` to:

```
com.knime.enterprise.executor.msgq=amqps://<username>:<password>@rabbitmq-host:<tls-port>/<virtual-host>
```

Note the appended `s` of the `amqps` protocol. By adjusting this address TLS connection is already possible, however, KNIME Server and Executor will accept any certificate provided by RabbitMQ. To only trust a specific certificate you have to export it into a truststore, e.g. by using the `keytool`. Once you created a truststore you can provide its information to KNIME Server via its `knime-server.config` file or via environment variables.

com.knime.enterprise.executor.msgq.tls_version=<value, e.g. TLSv1.2>

> Defines which TLS version should be used when connecting to RabbitMQ. The supported TLS version depends on the used Java Virtual Machine and the TLS support of RabbitMQ. By default TLS 1.2 will be used.

Env: `KNIME_EXECUTOR_MSGQ_TLS_VERSION=<value>`

com.knime.enterprise.executor.msgq.enable_hostname_verifier=<true|false>

> Defines if the hostname of the RabbitMQ server should be verified when connecting via TLS.
> By default the hostname won't be verified.

Env: `KNIME_EXECUTOR_MSGQ_ENABLE_HOSTNAME_VERIFIER=<true|false>`

com.knime.enterprise.executor.msgq.truststore_path=<value>

> Defines the path to the truststore that contains the server certificate used by RabbitMQ.
> By default all certificates are accepted.

Env: `KNIME_EXECUTOR_MSGQ_TRUSTSTORE_PATH=<value>`

com.knime.enterprise.executor.msgq.truststore_algorithm=<value>

> Defines the truststore algorithm for the specified `com.knime.enterprise.executor.msgq.truststore_path`. By default the system's default algorithm will be used. In case `com.knime.enterprise.executor.msgq.truststore_path` is not set this option is ignored.

Env: `KNIME_EXECUTOR_MSGQ_TRUSTSTORE_ALG=<value>`

com.knime.enterprise.executor.msgq.truststore_passphrase=<value>

> Defines the truststore passphrase for the specified `com.knime.enterprise.executor.msgq.truststore_path`. By default an empty passphrase is used. In case `com.knime.enterprise.executor.msgq.truststore_path` is not set this option is ignored.

Env: `KNIME_EXECUTOR_MSGQ_TRUSTSTORE_PASSPHRASE=<value>`

`com.knime.enterprise.executor.msgq.truststore_type=<value>`

> Defines the truststore type for the specified
> `com.knime.enterprise.executor.msgq.truststore_path`. By default the system's
> default type will be used. In case
> `com.knime.enterprise.executor.msgq.truststore_path` is not set this option is
> ignored.

Env: `KNIME_EXECUTOR_MSGQ_TRUSTSTORE_TYPE=<value>`

ℹ For the Executor you can use the same environment variables. In case you want to set the settings via the `knime.ini` file you simply can use its identifier prefixed by a `-D`, e.g. `-Dcom.knime.enterprise.executor.msgq.truststore_path=<value>`. Each modification of any of these settings requires a restart.

In addition to this a two way TLS connection to RabbitMQ can be established where KNIME Server and Executor provide a client certificate for RabbitMQ. To enable two way TLS in RabbitMQ follow the peer verification documentation.

`com.knime.enterprise.executor.msgq.keystore_path=<value>` **[ST]**

> Defines the path to the keystore that contains the client certificate provided to
> RabbitMQ when connecting via TLS.
> By default no keystore is set.

Env: `KNIME_EXECUTOR_MSGQ_KEYSTORE_PATH=<value>`

`com.knime.enterprise.executor.msgq.keystore_algorithm=<value>` **[ST]**

> Defines the keystore algorithm for the specified
> `com.knime.enterprise.executor.msgq.keystore_path`. By default the system's
> default algorithm will be used. In case
> `com.knime.enterprise.executor.msgq.keystore_path` is not set this option is
> ignored.

Env: `KNIME_EXECUTOR_MSGQ_KEYSTORE_ALG=<value>`

---

com.knime.enterprise.executor.msgq.keystore_passphrase=<value> [ST]

Defines the keystore passphrase for the specified
com.knime.enterprise.executor.msgq.keystore_path. By default an empty
passphrase is used. In case com.knime.enterprise.executor.msgq.keystore_path is
not set this option is ignored.

Env: KNIME_EXECUTOR_MSGQ_KEYSTORE_PASSPHRASE=<value>

---

com.knime.enterprise.executor.msgq.keystore_type=<value> [ST]

Defines the keystore type for the specified
com.knime.enterprise.executor.msgq.keystore_path. By default the system's
default type will be used. In case
com.knime.enterprise.executor.msgq.keystore_path is not set this option is
ignored.

Env: KNIME_EXECUTOR_MSGQ_KEYSTORE_TYPE=<value>

ℹ️ For the Executor you can use the same environment variables. In case you want
to set the settings via the knime.ini file you simply can use its identifier
prefixed by a -D, e.g.
-Dcom.knime.enterprise.executor.msgq.keystore_path=<value>. Each
modification of any of these settings requires a restart.

Loading new jobs

When a new job is loaded a so called new job message is sent to the message broker. As
soon as the Executor receives such a message it checks if the job can be accepted. In case
that the current Executor cannot handle the message it pushes it back to the message
broker. In previous versions the message was discarded as soon as an Executor has seen it a
second time and no other Executor could accept it. In order to ensure that all Executors have
the opportunity to see this message and accept it even in complex setups the following
setting is introduced:

`com.knime.enterprise.executor.job.rotation.limit=<value>` **[ST]**

> Defines the maximum number of times a new job message can be rotated. After an Executor sees the message for the specified number of times and no Executor marked it as executable the message is moved to the dead letter queue. Default value is 5.
>
> Env: `KNIME_EXECUTOR_JOB_ROTATION_LIMIT=<value>`

Job Pools

For workflows that are frequently executed it's now possible (starting with KNIME Server 4.8.1) to keep a certain number of jobs from that workflow in memory. This eliminates the overhead of loading the workflow in an Executor after the first use of that job. This should be particularly beneficial in cases where job loading time is large compared to job execution time.

Enabling job pools

In order to enable a job pool, a property has to be set on the workflow that should be pooled. Setting workflow properties can be done in the KNIME Explorer (starting with KNIME Server 4.9.0) by right-clicking on a workflow and selecting 'Properties…'. A dialog will open that lets the user view and edit the properties of the workflow.

Otherwise, workflow properties can also be set via a REST call, e.g. using `curl`:

```
curl -X PUT -u <user>:<password> http://<server-
address>/knime/rest/v4/repository/<workflow>:properties?com.knime.enterprise.server.jobp
ool.size=<pool size>
```

This will enable a pool with at most *pool-size* jobs for the workflow *workflow*.

It is only possible for single-call executions that do loading, execution, and discard in one call (i.e. the current `:execution` resource). Jobs that clients execute with multiple REST calls (load, execute, re-execute, discard) cannot be pooled.

Disabling job pools

Job pools can be disabled by setting the job pool size to 0, either in the KNIME Explorer or via a REST call:

```
curl -X PUT -u <user>:<password> http://<server-
address>/knime/rest/v4/repository/<workflow>:properties?com.knime.enterprise.server.jobp
ool.size=0
```

Using job pools

In order to make use of the pooled jobs, a special REST resource has to be called for executing a job. Instead of calling out to `:execution` you have to call to `:job-pool`. Apart from that both calls are identical concerning semantics and allowed parameters.

Executing a pooled job might look as follows:

```
curl -u <user>:<password> http://<server-
address>/knime/rest/v4/repository/<workflow>:job-pool?p1=v1&p2=v2
```

This will call *workflow* passing *v1* for input parameter *p1* and *v2* for input parameter *p2*. Calls using `POST` will work in a similar way using the `:job-pool` resource.

Behaviour of job pools

Job pools exhibit a certain behaviour which is slightly different from executing a non-pooled job. Clients should be aware of those differences.

- If the pool is empty (either intially or if all pooled jobs are currently in use) the job will be loaded from the workflow and thus the call will take longer.

- A used job will be put back into the pool right after the result has been returned if the pool isn't already full. Otherwise the job will be discarded.

- Pooled jobs are tied to the user that triggered initial loading of the job. A pooled job will never be shared among different users.

- If there is no job in the pool for the current user, the oldest job in the pool from a different user will be removed. This can lead to contention if there are more distinct users calling out to the pool than the pool size.

- Pooled jobs will be removed if they are unused for more than the configured job swap timeout (see the server configuration options).

- A pooled job **without** any input nodes will be reset before every invocation, even the first one! This is different from executing a non-pooled job but is required for consistent behaviour across multiple invocations. Otherwise the first and subsequent operations may behave differently if the workflow is saved with some executed nodes.

- In a pooled job **with** input nodes all of them will receive input values before execution: either the value that has been passed in the call, or if no explicit value has been provided its default value. This means that **all** input nodes will be reset prior to execution and not just the nodes explicitly set in the call. Again, this is different from executing a non-pooled job where only input nodes with explicitly provided values will be reset but required for consistency. Otherwise the results of a call may depend on the parameters passed in the previous call.

Workflow Pinning

Workflow Pinning can be used to let workflows only be executed by a specified subset of the available KNIME Executors when distributed KNIME Executors are enabled.
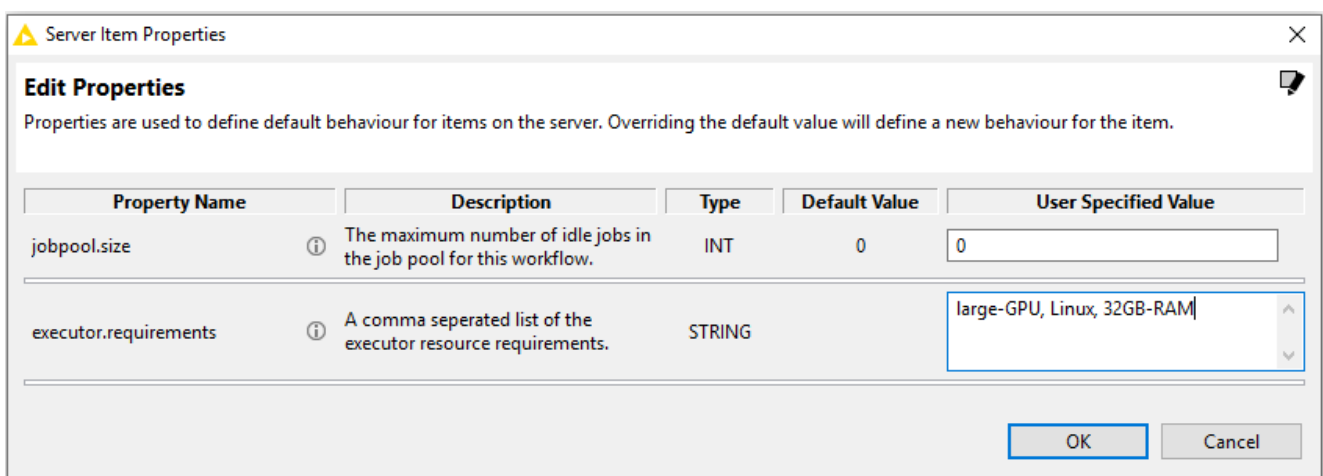
For workflows that need certain system requirements (e.g. specific hardware, like GPUs, or system environments, like Linux) it's now possible (starting with KNIME Server 4.9.0) to define such Executor requirements per workflow. Only KNIME Executors that fulfill the Executor requirements will accept and execute the workflow job. To achieve this behavior, a property has to be set for the workflows. Additionally, the system admin of the KNIME Executors has to specify a property for each Executor separately. The properties consist of values that define the Executor requirements, set for a workflow, and Executor resources, set for an Executor, respectively.

Prerequisites for workflow pinning

In order to use workflow pinning, the KNIME Server Distributed Executors must be enabled and RabbitMQ must be installed. Otherwise, the set Executor requirements are ignored.

Setting executor.requirements property for a workflow

Executor requirements for a workflow can be defined by setting a property on the workflow. The Executor requirements are a simple comma-seperated list of user-defined values. Setting workflow properties can be done in the KNIME Explorer by right-clicking on a workflow and selecting 'Properties…'. A dialog will open that lets the user view and edit the properties of a workflow.

Alternatively, workflow properties can also be set via a REST call, e.g. using `curl`:

```
curl -X PUT -u <user>:<password> http://<server-
address>/knime/rest/v4/repository/<workflow>:properties?com.knime.enterprise.server.exec
utor.requirements=<executor requirements>
```

This will set the executor requirements *executor-requirements* for the workflow *workflow*.

Setting executor.resources property for an executor

To define which resources an Executor can provide, a property has to be set for the Executors. This can be done in two ways:

1. Setting an environment variable on the system of an Executor. The name of the variable has to be 'KNIME_EXECUTOR_RESOURCES' and the value must be a comma-separated list of user-defined values.

```
KNIME_EXECUTOR_RESOURCES=value1, value2, value3
```

2. Setting a system property in the *knime.ini* file, which is located in the installation folder of the Executor. The file contains the configuration settings of the Executor, i.e. options used by the Java Virtual Machine. The name of the property has to be 'com.knime.enterprise.executor.resources' and the value must be a comma-seperated list of user-defined values.

```
-Dcom.knime.enterprise.executor.resources=value1, value2, value3
```

> **i** The environment variable has priority over the system property if both are specified.

Removing executor.requirements property for a workflow

Executor requirements can be removed by setting the property to an empty field. This can be done either in the KNIME Explorer or via a REST call:

```
curl -X PUT -u <user>:<password> http://<server-
address>/knime/rest/v4/repository/<workflow>:properties?com.knime.enterprise.server.exec
utor.requirements=
```

Removing executor.resources property for an Executor

The property can be removed either by completely removing the environment variable or by completely removing the property in the *knime.ini* file depending on the way the property was set. Alternatively, the property can also be removed by leaving the value of the environment variable or the value of the property in the *knime.ini* file empty.

> **i** A restart of the Executor is required to apply the changes.

Behaviour of Executor requirements

An Executor only accepts a job if it can fulfill all the Executor requirements that were defined for the workflow. Otherwise, it will just ignore the job.

- Jobs with no Executor requirements will be accepted by all available Executors.

- The executor.requirements property values only need to be a subset of the Executor's defined executor.resources property values in order for the workflow to be accepted by the Executor for execution.

- If no Executor can fulfill the Executor requirements, the queued job will be discarded.

- If the appropriate Executors cannot accept new jobs because their load is too high, the new queued job will run in a timeout (normally after 60 seconds) and discard itself, see Load throttling.

```
Example:
Workflow1 executor.requirements: medium_RAM, two GPU, Linux
Workflow2 executor.requirements: small-RAM, Linux
Workflow3 executor.requirements:
Executor1 executor.resources: small-RAM, Linux, two GPU
Executor2 executor.resources: medium_RAM, Windows, two GPU
Workflow1 will be ignored by both Executors and will be discarded.
Workflow2 will be ignored by Executor2 and accepted by Executor1.
Workflow3 will be accepted by any of the available Executors.
```

CPU and RAM requirements

Starting with KNIME Server 4.11 it is possible to define CPU and RAM requirements for a workflow. By default these requirements are ignored and disabled, unless at least one of the default values `com.knime.server.job.default_cpu_requirement` or `com.knime.server.job.default_ram_requirement` of the KNIME Server configuration file

options is set.

Setting CPU and RAM requirements property for a workflow

CPU and RAM requirements can be set in the same way as Executor requirements and is described in Setting executor.requirements property for a workflow. To set the CPU and RAM requirements the following keywords have been introduced:

```
cpu=<number of cores as decimal, e.g. 0, 0.1, 1, 2, 4>
```

The number of cores needed to execute the workflow. Note, that this value also allows decimals with one decimal place (further decimal places are ignored) in case workflows are small and don't need a whole core. The default is 0.

```
ram=<memory with unit, e.g. 512MB, 4GB>
```

An integer describing the size of memory needed for execution. The following units are allowed: GB (Gigabyte) and MB (Megabyte). In case no unit is provided it is automatically assumed to be provided in megabytes. The default is 0MB

In case no CPU or RAM requirement has been set for the workflow the default values `com.knime.server.job.default_cpu_requirement` and `com.knime.server.job.default_ram_requirement` defined in the KNIME Server configuration file are used. If both default values are either not set at all or set to 0 the CPU and RAM requirements of workflows are ignored.

Setting CPU and RAM properties for a KNIME Executor

The Executor detects the available number of cores and the maximum assignable memory automatically at startup.

Behaviour of CPU and RAM requirements

An Executor only accepts a job if it can fulfill the CPU and RAM requirements that were defined for the workflow. Otherwise, it will ignore the job. If a job gets accepted by an Executor its required CPU and RAM will be subtracted from the available resources until it gets either discarded/deleted or swapped back to KNIME Server. The time a job is kept on the Executor can be changed via the option `com.knime.server.job.max_time_in_memory` defined in the KNIME Server configuration file.

```
Example:
Workflow1 executor.requirements: cpu=1, ram=16gb
Workflow2 executor.requirements: cpu=1, ram=8gb
Workflow3 executor.requirements: cpu=0.1, ram=512mb
Executor: number of cores: 4, available RAM: 32GB
Workflow1 can be executed 2 times in parallel, since RAM is limiting
Workflow2 can be executed 4 times in parallel, since CPU and RAM is limiting
Workflow3 can be executed 40 times in parallel, since CPU is limiting
```

Executor Reservation

With the release of KNIME Server 4.11, we introduce the possibility to reserve KNIME Executors for exclusive use. This goes beyond the already existing workflow pinning since KNIME Executors can now refuse to accept jobs unless certain requirements are met.

There are two main use cases where this can be helpful:

1. Executor reservation based on workflow requirements: This allows you to ensure that Executors with certain properties (e.g. large memory, GPU) only accept jobs which are flagged as requiring these properties.

2. Executor reservation based on availability requirements for individual users or groups of users: This allows you to guarantee availability of execution resources to individuals or groups. E.g., you can reserve a KNIME Executor to only accept jobs if they are issued by users from a certain group.

Prerequisites for Executor Reservation

In order to use Executor reservation the same prerequisites as for workflow pinning are needed. The distributed KNIME Executors must be enabled and RabbitMQ must be installed. Reservations are ignored in single-executor deployments.

Setting executor.reservation property for a KNIME Executor

To define which requirements a job has to fulfill in order to get accepted by an Executor a property has to be set (in addition to defining the resources the Executor provides for workflow pinning) for this Executor. This can be done in two ways:

1. Setting an environment variable on the system of an Executor. The name of the variable has to be `KNIME_EXECUTOR_RESERVATION` and the value must be a valid boolean expression of Executor resources.

   ```
   KNIME_EXECUTOR_RESERVATION=resource1 && resource2 || resource3
   ```

2. Setting a system property in the *knime.ini* file, which is located in the installation folder of the Executor. The file contains the configuration settings of the Executor, i.e. options used by the Java Virtual Machine. The name of the property has to be `com.knime.enterprise.executor.reservation` and the value must be a valid boolean expression of Executor resources.

```
-Dcom.knime.enterprise.executor.reservation=resource1 && resource2 || resource3
```

> **i** The environment variable has priority over the system property if both are specified.

Removing executor.reservation property for a KNIME Executor

The property can be disabled by either removing the environment variable or by removing the property in the *knime.ini* file, depending on how the property was set. Alternatively, the environment variable or the value of the property in *knime.ini* can be set to an empty string.

> **i** A restart of the Executor is required to apply the changes.

Setting Executor reservation properties for a workflow

Setting the Executor reservation rules for individual workflows uses the same procedure as for workflow pinning. I.e., execution reservation is accessed by right-clicking a workflow in the KNIME Explorer and opening the 'Properties…' dialog.

Syntax and behaviour of Executor Reservation

The rule for Executor reservation is defined by a boolean expression and supports the following operations:

`resource: value`

A resource evaluates to `true` if and only if the job requirements contain the specified resource.
See workflow pinning.

`&&: r1 && r2`

Logical `AND` evaluates to `true` if and only if `r1` and `r2` evaluate to `true`, otherwise evaluates to `false`.

```
||: r1 || r2
```

> Logical `OR` evaluates to `true` if either `r1` or `r2` or both evaluate to `true`, otherwise evaluates to `false`.

```
!: !r
```

> Logical negation evaluates to `true` if and only if `r` evaluates to `false`, otherwise evaluates to `false`.

```
user: (user = <user>)
```

> Evaluates to `true` if and only if the user loading the job is `<user>`. Note that the parentheses are mandatory.

```
group: (group = <group>)
```

> Evaluates to `true` if and only if the user loading the job is in the specified group. Note that the parentheses are mandatory.

Note: the usual operator precedence of logical operators applies, i.e. `!` has a high precedence, `&&` has a medium precedence and `||` has a low precedence. Additionally, you can use parentheses, to overcome this precedence, e.g.:

```
A && B || A && C = A && (B || C)
```

A KNIME Executor only accepts a job if

- the Executor can fulfill **all** requirements that the job has, **and**
- if the job's resources requirements match the Executor's reservation rule.

Otherwise, the job will be rejected by the Executor. This also means that jobs with no resource requirements will be rejected if at least one resource is defined in the Executor's reservation rule.

The resources used in the reservation rule should be a subset of resources provided by the Executor, otherwise all jobs may get rejected as the Executor won't be able to fulfill the requirements.

If a job is not accepted by any Executor it will be discarded. If there are Executors that would accept a job but cannot do so right now because their load is too high, the new job will run into a timeout (normally after 60 seconds) and discard itself, see Load Throttling.

Most of special characters with exception of `'` are allowed to be part of users, groups, or resources. In this case the user names, group names, and resources values have to be put between `'`, e.g.:

```
(user = 'knime@knime.com') || (group = '@knime.com') && 'Python+Windows'
```

Example:

- Resources required by workflows:

    ◦ *w1* requires `large_RAM, Linux`

    ◦ *w2* requires `large_RAM, GPU`

    ◦ *w3* requires `Linux`

    ◦ *w4* requires `Windows`

    ◦ *w5* requires nothing

- Resources provided by Executors and reservation rules:

    ◦ *e1* provides `large_RAM, Linux, GPU` and is reserved for `large_RAM && (GPU || Linux)`

    ◦ *e2* provides `GPU, Windows` and is reserved for `!Linux`

- Possible job executions

    ◦ *w1* will be rejected by *e2* (because *e2* is reserved for `!Linux`) and will be accepted by *e1*.

    ◦ *w2* will be rejected by *e2* (because *e2* does not provide `large_RAM`) and will be accepted by *e1*.

    ◦ *w3* will be rejected by both KNIME Executors (because *e1* is reserved for `large_RAM` and *e2* is reserved for `!Linux`) and will be discarded.

    ◦ *w4* will be rejected by *e1* (because *e1* does not provide `Windows`) and accepted by *e2*.

    ◦ *w5* will be rejected by *e1* (because *e1* is reserved for `large_RAM`) and accepted by *e2* (because the empty requirement matches `!Linux`).

Executor Groups

With the release of KNIME Server 4.11, we introduce the possibility to group KNIME Executors for exclusive use. This extends the executor reservation since jobs are assigned to the specified KNIME Executor Group matching their requirements.

The main use case where this can be helpful is to allow you to ensure that jobs with certain properties (e.g. large memory, GPU), or based on certain user and groups are only handled by a specific group of KNIME Executors. This decreases potential delay in picking up the job as only possibly matching KNIME Executors will see the message. Furthermore it allows you to divide KNIME Executors into logical groups for easy maintenance (e.g. concerning scaling).

Prerequisites for KNIME Executor Groups

In order to use KNIME Executor Groups the same prerequisites as for workflow pinning are needed. The distributed KNIME Executors must be enabled and RabbitMQ must be installed. Groups are ignored in single-Executor deployments.

Creating KNIME Executor Groups

To define KNIME Executor Groups the following options have to be set in the KNIME Server configuration file:

`com.knime.enterprise.executor.msgq.names=<value>,<value>,…`

Defines the names of the KNIME Executor Groups. The number of names must match the number of rules defined with `com.knime.enterprise.executor.msgq.rules`. Note, that names starting with `amqp.` are reserved for RabbitMQ.

`com.knime.enterprise.executor.msgq.rules=<value>,<value>,…`

Defines the exclusivity rules of the KNIME Executor Groups. The number of rules must match the number of rules defined with `com.knime.enterprise.executor.msgq.names`.

Assigning KNIME Executors to a group

There are the following two ways to assign an Executor to a group.

1. Setting an environment variable on the system of a KNIME Executor. The name of the variable has to be `KNIME_EXECUTOR_GROUP` and the value must be one of the names defined in `com.knime.enterprise.executor.msgq.names`.

```
KNIME_EXECUTOR_GROUP=DefaultGroup
```

2. Setting a system property in the *knime.ini* file, which is located in the installation folder of the Executor. The file contains the configuration settings of the Executor, i.e. options used by the Java Virtual Machine. The name of the property has to be `com.knime.enterprise.executor.group` and the value must be one of the names defined in `com.knime.enterprise.executor.msgq.names`.

```
-Dcom.knime.enterprise.executor.group=DefaultGroup
```

> ℹ The environment variable has priority over the system property if both are specified.

In addition, it is necessary to also specify the resources that are offered by an Executor. The process is the same as described for workflow pinning. The list needs to contain at least all elements that are needed to distinguish the Executors within their group (except for rules based on user and/or group membership).

Setting Executor group properties for a workflow

Setting the KNIME Executor Groups for individual workflows uses the same procedure as for workflow pinning. I.e., execution reservation is accessed by right-clicking a workflow in the KNIME Explorer and opening the 'Properties…' dialog.

Syntax and behaviour of KNIME Executor Groups

The rules for KNIME Executor Groups are defined the same way as for executor reservation with the exception that a group with an empty rule accepts **every** job. KNIME Server sets up new message queues in RabbitMQ according to the provided groups.

If a workflow is loaded its requirements are considered and matched with the first workflow group for which the job fulfills the rules. Hence, the **order of the groups** in `com.knime.enterprise.executor.msgq.rules` may have an impact on which group gets selected. In case no suitable group can be found an error is thrown. Once a job is loaded it is associated with a single selected KNIME Executor Group.

> ℹ️ While Executor reservations are not necessary, the KNIME Executors still have to fulfill the requirements according to workflow pinning.

Example:

- Resources required by workflows:

    - *w1* requires `Python, GPU, group=G1`

    - *w2* requires `Python, GPU`

    - *w3* requires `Python, Linux`

    - *w4* requires `Python, Windows`

    - *w5* requires nothing

    - *w6* requires `huge_RAM`

- Executors groups with the rules:

    - *eg1* is reserved for `user='U1') || (group='G1' && Python && GPU`

    - *eg2* is reserved for `Python || GPU`

    - *eg3* is reserved for `Python || Windows`

    - *eg4* is reserved for `!huge_RAM`

    - *eg5* isn't reserved

- Possible job executions

    - *w1* will be passed to executor group *eg1*.

    - *w2* will be passed to executor group *e2*.

    - *w3* will be passed to executor group *eg2* (because *eg2* accepts every job that either has requirement `Python` **or** `GPU`).

    - *w4* will be passed to executor group *eg2* (because *eg2* accepts every job that either has requirement `Python` **or** `GPU` and does occur before group *eg3*).

    - *w5* will be be passed to executor group *eg4* (because it doesn't require `huge_RAM`).

    - *w6* will be be passed to executor group *eg5* (because it doesn't match any of the previous groups and *eg5* accepts every job).

Execution lifecycle

During the course of executing (or running) a workflow, there are several things that happen. Most of the time you don't need to know about this, but sometimes in more complex deployments, or for detailed debugging it may be helpful to understand the lifecycle of a workflow that is executed.

Workflows, Jobs and Job states

Workflows

The workflow is the collection of nodes, setup to perform your data analysis task. A workflow will contain all of the relevant (default) settings to perform the analysis. In addition to the settings a workflow may contain some data, e.g. if the workflow has been partially executed locally and then uploaded to the KNIME Server.

Jobs

On the KNIME Server, a Job is created whenever a workflow is executed. A full copy of the workflow is made into a location where other workflow executions can't interfere with it. For full details see Remote execution of workflows section on the KNIME Server User Guide.

Job states

Jobs exist in a variety of different states, which are displayed in either the Explorer view of the KNIME Analytics Platform, or the Jobs tab on the monitoring portal. The job states are:

- **LOADING** - Jobs being loaded by an Executor or waiting to be accepted by an Executor.

- **EXECUTING** - Job is currently executing.

- **EXECUTION_FINISHED** - Job has been executed successfully (may still be in memory, see notes below)

- **EXECUTION_FAILED** - Job has been executed, but failed (may still be in memory, see notes below)

- **EXECUTION_FAILED_WITH_CONTENT** - Job has been executed, failed, but was able to create a report

- **EXECUTION_CANCELLED** - Job has been cancelled manually during execution

- **INTERACTION_REQUIRED** - Job is currently executing on the WebPortal and is awaiting user input

- **NOT_EXECUTABLE** - Job contains individual, unconnected nodes

- **DISCARDED** - Job has been executed and discarded (meaning Executor resources, and server disk space are freed up.)

- **UNDEFINED** - This is the first state of a job, and may be seen in the case where a KNIME Executor cannot communicate with the server due to network issues, or the Executor not having enough free CPU/RAM resources.

- **VANISHED** - Job was in memory on an Executor that has crashed, and is therefore lost.

> **i** To ensure backwards compatibility with clients that use our REST API, we've introduced a new header called `KNIME-API-Version`, which can be set by clients to ensure full compatibility with their respective supported REST API version. In case this header is not provided the latest REST API version is used

Note that in addition to the job states there is the `In Memory` flag. The flag tells us whether the workflow is residing in the Executor memory, or has been swapped back to disk in the KNIME Server Repository. The setting `com.knime.server.job.max_time_in_memory` documented in KNIME Server configuration file options defines how long a job will remain in memory before being swapped. Additionally, when an Executor is gracefully shutdown then all jobs currently in memory are swapped back to disk. Additionally it's possible to manually force a job to swap to disk by issuing a REST call via SwaggerUI for Workflows using the job UUID.

Remote Workflow Editor

Introduction

The KNIME Remote Workflow Editor enables users to investigate the status of jobs on the server. Whenever a workflow is executed on the KNIME Server, it is represented as a job on the server and this instance of the workflow will be executed on the KNIME Server.

What is the Remote Workflow Editor

The Remote Workflow Editor looks just like your local workflow editor, apart from the fact that it is labelled and the canvas has a watermark to help identify that the workflow is running on the KNIME Server.

Most of the edit functionality that you would expect from editing a workflow locally on your machine is possible. Notable cases where it's not yet supported are: copying nodes from a local workflow to a remote workflow (and vice-versa), browse dialog for file reader/writer nodes browses the local filesystem rather than the remote filesystem.

Installation

The Remote Workflow Editor is installed on the KNIME Server into each KNIME Executor. Detailed instructions are found below.

For more information on how to install the relative extension on the KNIME Analytics Platform please refer to the Analytics Platform setup section on the KNIME Server User Guide.

Server setup

If KNIME Server is installed on Windows Server, then you may use the GUI to install the "KNIME Executor connector" from the "KNIME Server Executor (server-side extension)" feature. For Linux servers it is normally easier to use the command line to install the feature. Change to the KNIME Executor installation directory, and use the command:

```
./knime -application org.eclipse.equinox.p2.director -nosplash \
  -consolelog -r https://update.knime.com/analytics-platform/5.4 -i \
  com.knime.features.gateway.executor.feature.group -d $PWD
```
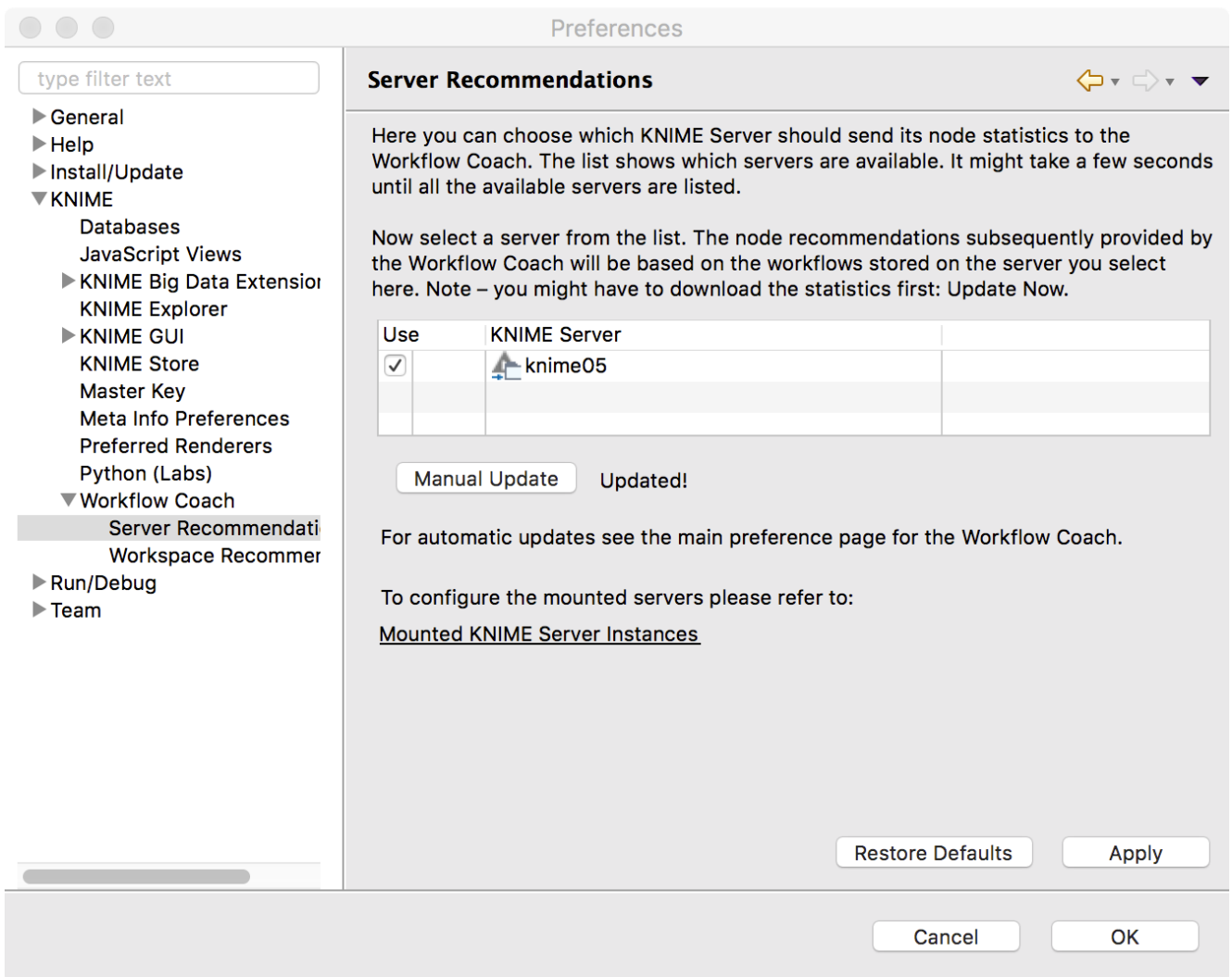
Custom Workflow Coach recommendations

KNIME Server is able to serve custom node recommendations to the workflow coach. In order to enable this functionality `com.knime.server.repository.update_recommendations_at=` must be set as described in the `knime-server.config` settings table.

The KNIME Analytics Platform preferences must be updated to enable the additional workflow coach recommendations:

Management Services for KNIME Analytics Platform: Customizations

Customizations allows to define centrally managed:

- Update sites
- Preference profiles (Database drivers, Python/R settings, etc.)

KNIME Server allows you to distribute customization profiles to connected KNIME Analytics Platform clients. A profile consists of a set of files that are fetched by the client during startup. The files are copied into the user's workspace. Files ending with `.epf` are treated as Eclipse preferences and can be used to override the *default* preferences which are usually defined by the extension vendors. Settings that an Analytics Platform user has already changed (i.e. which don't have the default value any more) are not affected. However, the user can choose to *"Restore ALL preferences to defaults"* via the preference page in the KNIME Analytics Platform. In this case the user is first prompted, then a backup of the preferences file is stored in the `<knime-workspace>/.metadata/knime/preferences-backup.epf`, finally, the server-managed settings will replace any preferences with the configured default values. The feature is available to all KNIME Server named-users and additionally to all registered consumers.

Analytics Platform Customization

The server installer will create a customization template profile in `config/client-profiles.template/customizations`. It consists of a preference file that contains all available configuration settings (including detailed descriptions) as well as some additional files that may be referenced in the preference file. Please see `customizations.epf` for details.

Server-side setup

In order to enable server-managed customization on the server side you have to create one or more subdirectories inside `<knime-server-repository>/config/client-profiles`. New server installations already come with an example profile that you can use as a starting point. You can have an arbitrary number of profiles. Which profiles are fetched by the client and in which order is defined by settings in the client (see below). If more than one profile defines a preference value, the *last* profile in the list requested by the client will determine the actual default value. Let's have a look at an example.

Suppose the `config/client-profiles` folder on the server has the following contents:

```
.../config/client-profiles/base/base.epf
        org.knime.workbench.explorer.view/knime.explorer.mountpoint=...
          org.knime.workbench.ui/knime.maxThreads=4
.../config/client-profiles/base/my-db-driver.jar
.../config/client-profiles/linux/linux.epf
        org.knime.workbench.ui/knime.maxThreads=8
        org.knime.python2/python2Path=/usr/bin/python2
        org.knime.python2/python3Path=/opt/anaconda3/bin/knime-python
.../config/client-profiles/windows/windows.epf
        org.knime.python2/python3Path=C:/Anaconda3/bin/knime-python.bat
.../config/client-profiles/windows/my-lib.dll
.../config/client-profiles/windows/my-db-driver.jar
```

If the client request the profiles "base,linux" (in this order), the default number of threads used by KNIME nodes will be 8. The python paths are set to the correct Linux paths. If another client requests "base,windows" the default number of threads will be 4 (from the base profile) and the Python 3 path will be set to a folder on the `C:\` drive. The pre-defined KNIME Explorer mount points will be identical for both clients because the value is only defined in the base profile.

A profile may contain several preferences files. They are all virtually combined into a single preference file for this profile in alphabetical order.

A profile may contain additional resources, for example JDBC driver files. The entire contents of the client-profiles folder including hidden files is sent to the client as a zip file and unpacked into a location in the client workspace. There is no conflict handling for any other files in the requested profiles (e.g. `my-db-driver.jar`) because they will end up in separate subdirectories on the client and not be processed further.

For further details and an example on how to distribute JDBC driver files go to the Server-managed Customization Profiles section of the KNIME Database Extension Guide.

If KNIME Server is running on Linux or macOS then the permissions of files inside profiles are transferred to the clients. This is useful for executable files on Linux or macOS clients, such as shell scripts. If you have such files in your profiles make sure to set the permissions accordingly on the server. KNIME Servers running on Windows don't support this feature because Windows file systems handle permissions differently.

Note that the profiles on the server are accessible without user authentication therefore they shouldn't contain any confidential data such as passwords.

In order to create preference files for clients, start a KNIME Analytics Platform with a *fresh workspace* on the desired environments (e.g. Linux, Windows). This ensures that all preferences are set to their vendor defaults. Then change the preferences to your needs and export them via *File → Export → KNIME Preferences*. Then copy the resulting `epf` file to the

profile folder on the server.

Variable replacement

It is possible to use variables inside the preferences files (only those files ending in `.epf`) which are replaced on the client right before they are applied. This makes the server-managed customizations even more powerful. These variables have the following format: `${prefix:variable-name}`. The following prefixes are available:

- `env`: the variable is replaced with the value of an environment value. For example, `${env:TEMP}` will be replaced with `/tmp` under most Linux systems.

- `sysprop`: the variable is replaced with a Java system property.
  For example, `${sysprop:user.name}` will be replaced with the current user's name. For a list of standard Java system properties see the JavaDoc. Additional system properties can be defined via `-vmargs` in the `knime.ini`.

- `profile`: the variable will be replaced with a property of the profile in which the current preference file is contained in. Currently "location" and "name" are supported as variable names. For example, `${profile:location}` will be replaced by the file system location of the profile on the client. This can be used to reference other files that are part of the profile, such as database drivers:
  `org.knime.workbench.core/database_drivers=${profile:location}/db-driver.jar`

- `origin`: the variable will be replaced with a HTTP response header sent by the server with the downloaded profiles. In addition to standard HTTP headers (which are probably not very useful), the following KNIME-specific origin variables are available:

  - `${origin:KNIME-Default-Mountpoint-ID}` — the server's configured default mount ID

  - `${origin:KNIME-EJB-Address}` — the address used by the KNIME Explorer; see the client profile templates in the repository created by the installer for an example

  - `${origin:KNIME-REST-Address}` — base address of the server's REST interface

  - `${origin:KNIME-WebPortal-Address}` — address of the server's WebPortal

  - `${origin:KNIME-Context-Root}` — base path on the server where all KNIME resources are available, usually `/knime`.

- `custom`: the variable will be replaced by the custom profile provider implementation that is also used to provide the profile location and list.

In case you want to have a literal in a preference value that looks like a variable, you have to use two dollar signs to prevent replacement. For example `$${env:HOME}` will be replaced with

the plain text `${env:HOME}`. If you want to have two dollars in plain text, you have to write three dollars (`$$${env:HOME}`) in the preference file.

Note that once you use variables in your preference files they are not standard Eclipse preference files anymore and cannot be imported as they are.

Client-side setup

The client has three possibilities to request profiles from a KNIME Server.

1. Two command line arguments which define the address and the (ordered) list of requested profiles (note that the command line argument and the variable must be separated onto two lines — as seen below):

```
-profileLocation
http://knime-server:8080/knime/rest/v4/profiles/contents
-profileList
base,linux
```

   Both arguments must be supplied either directly on the command line or in the `knime.ini` **before** the `-vmargs`.

2. Two preference settings in the "KNIME/Customization profiles" preference page. There the user can select a server and then define the ordered list of profiles that he/she wants to apply. Note that this setting cannot be controlled by the server-managed customization profiles. Changes will take effect after the next start.

3. A custom profile provider defined in a custom Eclipse plug-in. Since this involves writing Java code and is likely only of interest for large-scale installations we cover this approach in the KNIME Server Advanced Setup Guide.

The three possibilities are tried in exactly this order, i.e. if one of them provides a server address and a non-empty list of profiles it will be used and all following providers will be skipped.

It's also possible to provide a local file system folder as the `profileLocation` on the command line (or in your custom profile provider). The layout of this local folder must be the same as the profiles folder on the server.

Client customization

Besides the preferences that are exportable by KNIME Analytics Platform there are additional settings that can be added to the preference files to customize clients:

/instance/org.knime.workbench.explorer.view/defaultMountpoint/defaultMountpoints
=<mount id1>,<mount id2>,⋯

A comma separated list of default mount points that should be loaded,e.g. LOCAL,EXAMPLES,My-KNIME-Hub. Changes to this list only affects new workspaces, i.e. workspaces which already contain default mount points will still contain them even though only they haven't been defined here. If this option is absent and defaultMountpoint/enforceExclusion isn't set to true then all default mount points will be added. The current default mount points are LOCAL, EXAMPLES, and My-KNIME-Hub.

/instance/org.knime.workbench.explorer.view/defaultMountpoint/enforceExclusion=<
true|false>

If set to true then all default mount point not defined by /instance/org.knime.workbench.explorer.view/defaultMountpoint/defaultMountpoints will be removed on start up. Please note that if you want to use this option, the default mountpoints you want to include should **only** be listed in /instance/org.knime.workbench.explorer.view/defaultMountpoint/defaultMountpoints, and **not** in their full definition like when exporting the preferences.epf file from a KNIME Analytics Platform.

/instance/com.knime.customizations/helpContact.buttonText=<label, e.g. Contact
Support>

If set together with /instance/com.knime.customizations/helpContact.address a button with the provided label will occur under Help in KNIME Analytics Platform. Clicking on the button will, depending on the helpContact.address, either open the default mail client or the default browser with the provided address.

/instance/com.knime.customizations/helpContact.address=<uri, e.g.
mailto:support@company or https://company/support>

Sets the address of the support contact.
This option only takes effect in combination with /instance/com.knime.customizations/helpContact.buttonText.

/instance/com.knime.customizations/documentation.buttonText=<label, e.g. Documentation>

> Sets the label of the documentation button that can be found under `Help` in KNIME Analytics Platform. Clicking on the button will open the default browser and navigate to the documentation. If set to - the button will be hidden.

/instance/com.knime.customizations/documentation.address=<uri, e.g. https://company/documentation or file:///sharedSpace/documentation>

> Sets the address of the documentation.
> By default the documentation address points to the KNIME documentation.

/instance/com.knime.customizations/windowTitle.appendix=<appendix, e.g. sponsored by company>

> Adds the appendix to the window title of KNIME Analytics Platform.
> -

/instance/com.knime.customizations/updateSite.uris=<uri>,<uri>,···

> Adds the provided addresses to the update sites.
> -

/instance/com.knime.customizations/updateSite.names=<name>,<name>,···

> The names that are shown under `Available Software Sites` for the provided update sites of option. Note that the number of names must match the number of provided URIs.

/instance/com.knime.customizations/updateSite.default.disable=<true|false>

> Disables the default added update sites added by KNIME after a fresh installation or update. If a user enables these update sites again they will remain enabled.

/instance/com.knime.customizations/updateSite.default.forceDisable=<true|false>

> Disables the default added update sites added by KNIME after a fresh installation or update. If a user enables these update sites again they will be disabled with the restart of their client.

```
/instance/com.knime.explorer.server/oauth_login_success_page=<path, e.g.
${profile:location}/success.html>
```

> Replaces the redirect site that is shown by KNIME Analytics Platform after a
> successful login to KNIME Server via OAuth. Note, this option is only available
> starting with the extension KNIME ServerSpace 4.12.1.

```
/instance/com.knime.explorer.server/oauth_login_failed_page=<path, e.g.
${profile:location}/error.html>
```

> Replaces the redirect site that is shown by KNIME Analytics Platform after a failed
> login to KNIME Server via OAuth. Note, this option is only available starting with the
> extension KNIME ServerSpace 4.12.1.

Security considerations

The following section describe some general security considerations for running a KNIME Server. Some of them are active by default, some other require manual configuration based on your specific environment.

Protecting configuration files

The configuration files must be accessible by the system account running the KNIME Server. However, this account also runs the KNIME Executor which executes the workflows. This means that a malicious workflow can in principle access the server configuration files if the absolute file system paths are known. Therefore, for high security environments we recommend removing write permissions on the configurations files from the system account so that at least the workflow cannot modify them. This includes the following directories and their contained files:

- `<apache-tomcat>/conf`

- `<apache-tomcat>/bin`

- `<apache-tomcat>/endorsed`

- `<apache-tomcat>/lib`

- `<knime-server-repository>/config`

Encrypted communication

Communication between KNIME Analytics Platform and KNIME Server is performed via HTTP(S). By default, both unencrypted communication via HTTP and encrypted communication via HTTPS (SSL) is enabled.

All encryption is handled by Tomcat, see the Tomcat SSL Configuration How-to for full documentation.

Enabling HSTS (HTTP Strict Transport Security) is also described in the Tomcat documentation.

Server configuration

The KNIME Server installer will enable encryption using a generic server certificate that the client accepts. Note that most browsers will issue a certificate warning when you access the

KNIME WebPortal via https for the first time. For production it is recommended to add your own certificate as follows:

1. Obtain a certificate and create a new Java keystore file named `knime-server.jks` as described in Tomcat SSL Configuration How-to

2. Replace the `<apache-tomcat>/conf/knime-server.jks` with the keystore file created in the previous step (note: this will replace the generic server certificate)

3. Adjust the `certificateKeystorePassword` of the following "<Connector… />" definition found in `<apache-tomcat>/conf/server.xml` to match the password used in the first step:

```
<Connector SSLEnabled="true" compression="on" maxThreads="150"
    protocol="org.apache.coyote.http11.Http11Nio2Protocol"
    port="8443" scheme="https" secure="true" server="Apache Tomcat">
    <SSLHostConfig protocols="TLSv1, TLSv1.1, TLSv1.2">
        <Certificate
            certificateKeystoreFile="conf/knime-server.jks"
            certificateKeystorePassword=<your password>
            type="RSA"/>
    </SSLHostConfig>
</Connector>
```

You can also adjust the port number but you should not change any of the other value unless you understand the implications.

4. Restart Tomcat.

If you want to enforce using only encrypted communications (HTTPS), we suggest to completely disable the unencrypted HTTP connector on port 8080 (by default). To do this remove the line that defines the first HTTP Connector in the `server.xml` or embed it into an XML comment so that it is not processed on startup.

Client configuration

If you want encrypted connection from KNIME Analytics Platform to KNIME Server, you have to make sure that KNIME accepts the server certificate. If you have a "real" certificate that was signed by a well-known certification authority then you should be safe. If the signing CA is not known to Java you have to add the CA's certificate to the keystore used by KNIME:

1. Get the CA's certificate in PEM format.

2. Add the CA certificate to the JRE's keystore file in

(For Linux and Windows)

```
`<knime-folder>/jre/lib/security/cacerts`
```

(KNIME Analytics Platform 3.4.3 and older) or

```
`<knime-folder>/plugins/org.knime.binary.jre.<..>/jre/lib/security/cacerts`
```

(For macOS)

```
`/plugins/org.knime.binary.jre.macosx.x86_64_11.0.10.20210416/cacerts`
```

(KNIME Analytics Platform 3.5.0 and newer). This is performed with the keytool command that is part of any Java installation (e.g. `<knime-folder>/<jre-folder>/bin/keytool`):

```
keytool -import -trustcacerts -alias <ca-alias> \
    -file <CA.crt> -keystore jre/lib/security/cacerts
```

You can choose an arbitrary name for `<ca-alias>`. For `<CA.crt>` insert your CA's certificate file. The password for the keystore file is "changeit".

Disabling the Manager application

The default KNIME Server installation does not add any users with permissions to access the manager application. The Tomcat manager application is not required for the correct functioning of KNIME Server. You may wish to disable the functionality by deleting the `manager`, `host-manager` and `ROOT` directories from your installation. Note that you should not delete the `ROOT` directory if you chose to install KNIME Server using the context root of `ROOT`.

Tomcat shutdown port

The Tomcat shutdown port is accessible on port 8005, which should not be accessible from machines other than localhost. We have renamed the `SHUTDOWN` command to a random string that is generated at installation time.

You may choose to remove this option completely by finding the following configuration in the `server.xml`:

```
<Server port="8005" shutdown="<RANDOMSTRING>">
```

and changing it to: <Server port="-1" shutdown="<RANDOMSTRING>">

CSRF prevention

Cross-site request forgery (CSRF) is a type of malicious exploit of a website where unauthorized commands are transmitted from a user that the website trusts (see the Wikipedia entry for more technical details). In the context of KNIME Server this means that some other web page issues a (hidden) REST request to KNIME Server using the current user's active WebPortal session. The user usually doesn't notice anything but operations are performed with their account. Since version 4.3.4 KNIME Server contains a CSRF protection which prevents any modification requests (e.g. `POST`, `PUT`, or `DELETE`) to REST methods from hosts other than KNIME Server itself.

In case you have internal web pages on other hosts that deliberately perform valid requests you can disable CSRF protection by adding the following line to `<apache-tomcat>/conf/Catalina/localhost/knime.xml`:

```
<Parameter name="com.knime.server.rest.csrf-protection" value="false"
    override="false" />
```

Avoid clickjacking attacks

Clickjacking is also a malicious attempt to trick a user into clicking on something different than perceived, potentially revealing confidential information or taking control of the computer. (See the Wikipedia entry for more technical details). The best option to avoid clickjacking is setting the HTTP header `X-Frame-Options` to an appropriate value to prevent the WebPortal being embedded in a third party website. In KNIME Server this can be done with a configuration option `com.knime.server.webportal.restrict_x_frame_options`. The value can be one of `DENY`, `SAMEORIGIN` or `ALLOW-FROM any_origin_url`. See also this article from MDN about more details of the header and available options.

Please note that, if you want to embed the WebPortal on a different website and want this setting to be enabled, you will have to set the value to `ALLOW-FROM xxx` (where `xxx` has to be replaced with the URL of the embedding website).

Hiding server details

By default, Tomcat prints its name and version number on error pages (e.g. if a location entered in the browser does not exist) and in standard HTTP headers. This information can be used by an attacker to target potential security issues for this particular version. Therefore for high security environments it's recommended to at least hide the server's version. Fresh installations from 4.5 onwards already hide the version. If you are upgrading from an existing

installation, you can apply the following two small configuration changes:

- Add a file `<apache-tomcat>/lib/org/apache/catalina/util/ServerInfo.properties` with the following contents:

```
server.info=Apache Tomcat
server.number=8.5.11.0
server.built= Jan 10 2017 21:02:52 UTC
```

  Only the value of "server.info" is shown in error pages and by default includes the version number. The above example only exposes the server's name.

- Modify the `<Connector>` entries in `<apache-tomcat>/conf/server.xml` and add an attribute "server" with "Apache Tomcat" as value:

```
<Connector port="8080" *server="Apache Tomcat"* ... />
```

  This change hides the server version in HTTP headers.

You may also choose to set the following parameter in the knime-server.config file. For full details see the option `com.knime.server.webportal.hide_version` KNIME Server configuration file section in the KNIME WebPortal Administration Guide.

Advanced settings

There are a couple more actions you can take to make the server and the application even more secure which we don't discuss in detail here because they are only useful in special setups. Example are

- Cross-Origin Resource Sharing
- Strict Transport Security
- Content Security Policy (this policy cannot be implemented in Tomcat without writing custom code; see the section about Running behind frontend server for a possible solution)

Running behind frontend server

In some cases it makes sense to run KNIME Server (Tomcat) behind a frontend server. Examples are:

- Running several KNIME Servers under the same (public) hostname

- Adding custom HTTP headers (e.g. Content Security Policy, see above)

- Reusing existing HTTPS configurations

- Using standard ports (80, 443)

No configuration changes are required on the KNIME Server side, however, the frontend server must ensure that:

- The public hostname is passed to KNIME Server in all HTTP requests. See the example below for details

- The context root is passed to KNIME Server if it differs from the value configured in KNIME Server

- Information about the public protocol (HTTP or HTTPS) is passed onto the KNIME Server.

Otherwise links generated by KNIME Server may point to the internal address which is useless for outside clients and can even expose sensitive information. A sample configuration for Apache HTTPD looks as follows:

```
<VirtualHost *:443>
    ServerName public.knime.server

    # Make sure the public protocol is passed to the server;
    # not required if internal and external protocol are the same
    RequestHeader set X-Forwarded-Proto "https"

    # If a different context root than in KNIME Server is used
    # then the ProxyPass config should also be changed to reflect this
    # n.b. the leading slash is mandatory
    RequestHeader set KNIME-Context-Root-Rewrite "/apache-root"

    # Ensure that the public hostname is also used in forwarded requests
    ProxyPreserveHost On
    ProxyRequests Off

    ProxyPass /tomee/ejb http://internal:8080/tomee/ejb
          keepalive=On nocanon
    ProxyPass /knime http://internal:8080/knime

    # Optional
    ProxyPass /com.knime.enterprise.sketcher.ketcher
          http://internal:8080/com.knime.enterprise.sketcher.ketcher
</VirtualHost>
```

Please note that such advanced setups require detailed knowledge about Tomcat and Apache configuration (or whatever frontend server you are using) and we can only provide limited support.

Managing access to files/workflows/components

You can assign access permissions to each server item (workflows or workflow groups) to control the access of other users to your workflows and groups.

The owner

The server stores the owner of each server item, which is the user that created the item. When you upload a flow, copy a workflow, save a workflow job (an executed flow) or create a new workflow group you are assigned to the new item as owner. When a new server item is created, you can set the permissions how you want this item to be available to other users. Later on, only the owner can change permissions on an item.

User groups

When the KNIME Server administrator defines the users that have access to the KNIME Server, the users are assigned to groups. Groups can be defined as needed — for example one group per department, or per research group, etc. Each user must be in at least one group, and could be in many groups.
You can set a group to be an administrator group (with the configuration option "com.knime.server.server_admin_group=<group name>"). Users assigned to that group are considered server administrators.

Server administrator

Specific users can be set server administrator with a configuration option (`com.knime.server.server_admin_users=<user>,<user>,…,`) or by assigning them to the administrator group (see section User groups). Server administrators are not restricted by any access permissions. Administrators always have the right to perform any action usually controlled by user access rights. They can always change the owner of an item, change the permissions of an item, they see all workflow jobs (while regular users only see their own jobs) and they can delete all jobs and items.

Access rights

There are three different access rights that control access to a workflow and two for a workflow group:

Workflow group permissions

Read	Allows the user to see the content of the workflow group. All workflows and subgroups contained are shown in the repository view.
Write	If granted, the user can create new items in this workflow group. The user can create new subgroups and can store new workflows or Shared Components in the group. Also deletion of the group is permitted.

In order to add a workflow to a certain group, you only need permissions to write to that particular workflow group, not to any parent workflow group.

> ℹ It is also possible to hide workflow groups to users that do not have read-permission to those groups and do not have write-permission to the parent workflow group. To do so please refer to the option `com.knime.server.repository.hide_unreadable_groups` in the KNIME Server configuration file options section.

Workflow permissions

Execute	Allows the user to execute the flow, to create a workflow job from it. It does not include the right to download that job, or even store the job after it finishes (storing requires the right to download).
Write	If granted, the user can overwrite and delete the workflow.
Read	Allows the user to download the workflow (including all data stored in the flow) to its local desktop repository and inspect the flow freely.

Note: Executing or downloading/reading a flow does not require the right to read in the group that contains the flow. In fact, there is currently no right controlling the visibility of a single

flow (there is no "hidden" attribute).

Access to workflow jobs and scheduled jobs

There are no permissions to be set on a workflow job or a scheduled job. Only the owner — the user that created the job — can see the job in the repository view, and this user is the only one that can delete it (besides any server administrator).

In order to store a workflow job as new workflow in the server's repository, the user needs the right to download the original workflow (the flow the job was created from). (This is a requirement, because the newly created workflow is owned by the user that stores the job — and the owner can easily grant itself the right to download the flow. Thus, if the original flow didn't have the download right set, the user that is allowed to execute the flow could easily work around the missing download right.)

"Owner", "Group", and "Other" rights

As the owner of a server item (workflow, shared component or workflow group) you can grant access rights to other users. But you can only assign permissions on a group level, not for particular users.

Owner rights

The owners can assign permissions to themselves to protect a flow from accidental deletion. The owner can change its own permissions at any time.

Group rights

The owner of a server item can assign permissions to all users of a specific group. If an access right is granted to a group, all users that are assigned to this group have this right.

"Other" rights

Permissions can be set to all users that are not the owner and that are not in one of the groups.

Note: Access rights are adding up and can't be withdrawn — that means, if, for example, you grant the right to execute a flow to "other" users and you define permissions for a certain group of users not including the execute right, these users of that group are still able to

execute that flow, as they obtain that right through the "other" permissions.

Webservice interfaces

RESTful webservice interface

KNIME Server supports execution of workflows via a REST interface. The entry point for the REST interface is `http://server-address/knime/rest/`.

The interface is based on a hypermedia-aware JSON format called Mason. Details about the interface, its operations, endpoints and message formats are provided at the following locations (best opened in an internet browser):

- `http://<server-address>/knime/rest/_profile/knime-server-doc.xml` for the general interface and

- `http://<server-address>/knime/rest/v4/_profile/knime-server-doc-v4.xml` for the 4.x API

(see also the "Link" HTTP header in all responses returned by the server).

The usual starting point to query the repository and to execute operations is
`http://<server-address>/knime/rest/v4/repository/`
(note the trailing '/'). The returned document also contains links to further operations.

SwaggerUI for Workflows

The KNIME Server automatically generates SwaggerUI pages for all workflows that are present on the KNIME Server. From the KNIME Analytics Platform you can access that functionality using the `Show API definition` context menu item.
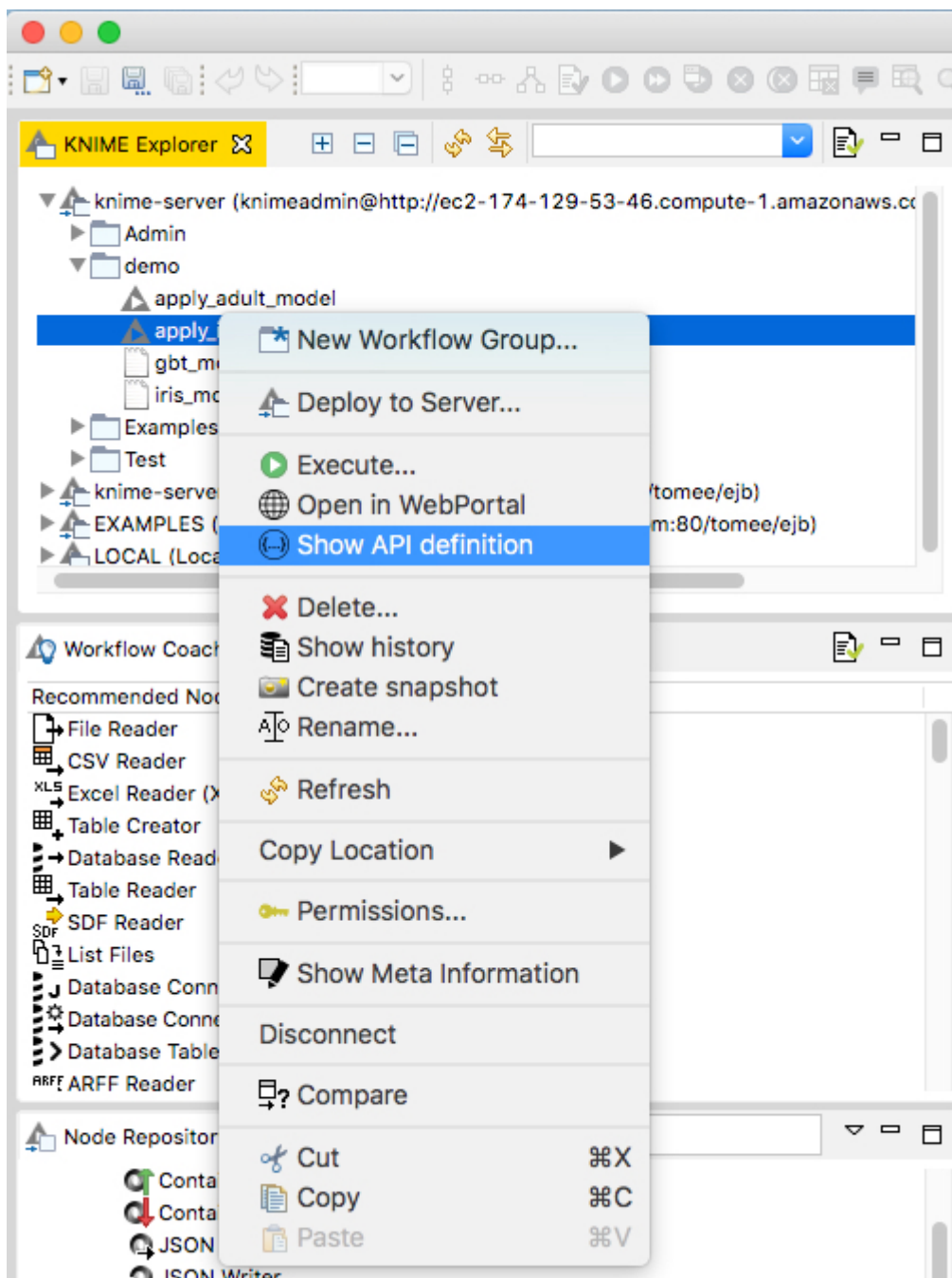
Clicking the menu item will open a SwaggerUI page for that workflow in your browser. It's also possible to browse to that page using the REST API as described in the above section.

POST /v4/repository/demo/apply_iris_model:execution Executes a job from this workflow 🔒

This call combines loading, executing, and deleting a job in one call. You can pass input parameter for quickform nodes defined in the workflow.
All input parameters are suffixed with their unique node ID in order to make the parameters unique themselves. If a parameter name is unique without the node ID suffix you can also omit the suffix when sending it to the server. For example, if the fully qualified parameter name is int-input-1 and there is no other input parameter that begins with int-input you can use int-input as the name in your request.

Parameters		Try it out

Name	Description
timeout integer (query)	Sets a timeout in milliseconds that the call should wait for the job being loaded. If the workflow doesn't load within the time a 504 error will be returned.
format string (query)	If the workflow creates a report you can specify the desired report format. If no report format is provided no report will be generated.
reset boolean (query)	True if the job should be reset before execution. If false (the default) job execution continues from its saved state.

Request body   application/json ⌄

Inline input parameters for the job

Example Value | Model

```
{
  "json-input-1": {
    "sepal length": 8.2,
    "sepal width": 5.3,
    "petal length": 0.3,
    "petal width": 0.4
  }
}
```

Common problems

Always reset with flow variables

If the values of flow variables are changed in the remote execution dialog, the flow must be reset in order for the new values to be propagated. In this case, don't remove the checkmark "Reset before Execution" in the execution dialog.

knime.ini file not found

If the KNIME instance that is used to execute flows on the server doesn't seem to have the settings specified in the `knime.ini` file, it is possible that the server didn't find the ini-file: The server takes the default ini-file from the same folder as the KNIME executable. If you specify a wrapper script as executable that is located outside the installation folder it doesn't find the default ini-file. In this case copy the ini-file from the installation folder into `<knime-server-repository>/config`.

Server startup takes a long time

In some cases it may take quite some time (up to several minutes) until the server responds to requests on Linux systems.

Insufficient entropy

This is usually caused by insufficient entropy for the random number generator used by Tomcat. You can work around this issue by specifying a different random number source, which will provide numbers faster but which are also less random:

1. Edit `<apache-tomcat>/conf/catalina.properties`.
2. Add a line `java.security.egd=file:/dev/./urandom` at the bottom of the file (note the "/./")
3. Restart Tomcat

Large number of jobs

In cases where the KNIME Server retains a large number of jobs then it may be necessary to increase the amount of memory that Tomcat can access. Simply edit the file setenv.bat

(Windows) or setenv.sh (Linux) to increase the value of -Xmx to double the current setting.

Changelog (KNIME Server 4.18)

KNIME Server 4.18.0

(released January 20, 2025)

Enhancement

- SRV-3863: Update Tomcat to 9.0.98

Third party software licenses

The KNIME Server software makes use of third-party software modules, that are each licensed under their own license. Some of the licenses require us to note the following:

The following libraries are used and licensed under the CDDL v1.1 and are owned by Oracle. The copyright belongs to the respective owners.

- `javax.json-api-1.1.4.jar`

- `javax.activation-api-1.2.0.jar`

- `javax.annotation-api-1.3.2.jar`

- `javax.json-1.1.4.jar`

- `jaxb-api-2.3.1.jar`

- `javax.xml.soap-api-1.4.0.jar`

- `jaxws-api-2.3.1.jar`

- `jstl-1.2.jar`

The following libraries are used and licensed under the Apache 2.0 license. The copyright belongs to the respective owners.

- `amqp-client-5.10.0.jar`

- `aws-java-sdk-core-1.11.1009.jar`

- `bcel-6.3.1.jar`

- `bson4jackson-2.12.0.jar`

- `commons-codec-1.15.jar`

- `commons-collections-3.2.2.jar`

- `commons-collections4-4.4.jar`

- `commons-compress-1.21.jar`

- `commons-fileupload-1.4.jar`

- `commons-io-2.8.jar`

- `commons-lang3-3.11.jar`

- `commons-logging-1.2.jar`

- `commons-text-1.9.jar`

- `cxf-core-3.4.5.jar`

- `cxf-rt-frontend-jaxrs-3.4.5.jar`

- `cxf-rt-rs-service-description-common-openapi-3.4.5.jar`

- `cxf-rt-rs-service-description-openapi-v3-3.4.5.jar`

- `cxf-rt-rs-service-description-swagger-ui-3.4.5.jar`

- `cxf-rt-security-3.4.5.jar`

- `cxf-rt-transports-http-3.4.5.jar`

- `error_prone_annotations-2.3.4.jar`

- `failureaccess-1.0.1.jar`

- `geronimo-jta_1.1_spec-1.1.1.jar`

- `guava-30.0-jre.jar`

- `httpclient-4.5.13.jar`

- `httpcore-4.4.13.jar`

- `ion-java-1.0.2.jar`

- `j2objc-annotations-1.3.jar`

- `jackson-annotations-2.13.2.jar`

- `jackson-core-2.13.2.jar`

- `jackson-databind-2.13.2.2.jar`

- `jackson-dataformat-cbor-2.13.2.jar`

- `jackson-dataformat-yaml-2.13.2.jar`

- `jackson-dataformat-xml-2.13.2.jar`

- `jackson-datatype-jdk8-2.13.2.jar`

- `jackson-datatype-jsr310-2.13.2.jar`

- `jackson-datatype-jsr353-2.13.2.jar`

- `jackson-jaxrs-base-2.13.2.jar`

- `jackson-jaxrs-json-provider-2.13.2.jar`

- `jackson-module-jaxb-annotations-2.13.2.jar`

- `jackson-module-mrbean-2.13.2.jar`

- `javassist-3.27.0-GA.jar`

- `je-7.4.5.jar`

- `jsr305-3.0.2.jar`

- `joda-time-2.8.1.jar`

- `listenablefuture-9999.0-empty-to-avoid-conflict-with-guava.jar`

- `ognl-3.0.8.jar`

- `openjpa-lib-3.1.2.jar`

- `org.osgi.compendium-4.3.1.jar`

- `org.osgi.core-4.3.1.jar`

- `qpid-bdbstore-8.0.2.jar`

- `qpid-broker-core-8.0.2.jar`

- `qpid-broker-plugins-amqp-0-8-protocol-8.0.2.jar`

- `snakeyaml-1.30.jar`

- `stax2-api-4.2.jar`

- `swagger-annotations-2.1.5.jar`

- `swagger-core-2.1.6.jar`

- `swagger-integration-2.1.6.jar`

- `swagger-jaxrs2-2.1.6.jar`

- `swagger-models-2.1.6.jar`

- `thymeleaf-2.1.4.RELEASE.jar`

- `txtmark-0.13.jar`

- `unbescape-1.1.0.RELEASE.jar`

- `woodstox-core-6.2.3.jar`

- `xmlbeans-3.1.0.jar`

- `xmlschema-core-2.2.5.jar`

The following libraries are used and licensed under the MIT license. The copyright belongs to the respective owners.

- `checker-qual-3.5.0.jar`

- `classgraph-4.8.65.jar`

- `slf4j-api-1.7.30.jar`

- jquery 2.2.4

- lodash 4.17.4

- react-15.6.2

- react-bootstrap 0.29.5

- react-bootstrap-table 3.3.4

- react-dom 15.6.2

- react-sidebar 2.1.1

The following libraries are used and licensed under the BSD 3-clause license. The copyright belongs to the respective owners.

- Node-forge 0.7.4 (Copyright (c) 2010, Digital Bazaar, Inc. All rights reserved.)

- serp-1.15.1.jar

- asm-9.0.jar

The following libraries are used and licensed under the BSD 2-clause license. The copyright belongs to the respective owners.

- parsington-2.0.0.jar

The following libraries are used and licensed under the Eclipse Distribution License v 1.0. The copyright belongs to the respective owners.

- jakarta.activation-1.2.1.jar

- jakarta.activation-api-1.2.2.jar

- jakarta.annotation-api-1.3.5.jar

- jakarta.jws-api-2.1.0.jar

- jakarta.validation-api-2.0.2.jar

- jakarta.xml.bind-api-2.3.3.jar

- jakarta.xml.soap-api-1.4.2.jar

- jakarta.xml.ws-api-2.3.3.jar

- jaxb-runtime-2.3.4.jar

- stax-ex-1.8.3.jar

- txw2-2.3.4.jar

- `istack-commons-runtime-3.0.12.jar`

- `saaj-impl-1.5.3.jar`

The following libraries are used and licensed under the Do what the fuck you want to public license. The copyright belongs to the respective owners.

- `reflections-0.9.12.jar`

The following libraries are used and licensed under the GPL2 w/ CPE. The copyright belongs to the respective owners.

- `jakarta.mail-1.6.5.jar`

CDDL v1.1

1. Definitions. 1.1. "Contributor" means each individual or entity that creates or contributes to the creation of Modifications. 1.2. "Contributor Version" means the combination of the Original Software, prior Modifications used by a Contributor (if any), and the Modifications made by that particular Contributor. 1.3. "Covered Software" means (a) the Original Software, or (b) Modifications, or (c) the combination of files containing Original Software with files containing Modifications, in each case including portions thereof. 1.4. "Executable" means the Covered Software in any form other than Source Code. 1.5. "Initial Developer" means the individual or entity that first makes Original Software available under this License. 1.6. "Larger Work" means a work which combines Covered Software or portions thereof with code not governed by the terms of this License. 1.7. "License" means this document. 1.8. "Licensable" means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein. 1.9. "Modifications" means the Source Code and Executable form of any of the following: A. Any file that results from an addition to, deletion from or modification of the contents of a file containing Original Software or previous Modifications; B. Any new file that contains any part of the Original Software or previous Modification; or C. Any new file that is contributed or otherwise made available under the terms of this License. 1.10. "Original Software" means the Source Code and Executable form of computer software code that is originally released under this License. 1.11. "Patent Claims" means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor. 1.12. "Source Code" means (a) the common form of computer software code in which modifications are made and (b) associated documentation included in or with such code. 1.13. "You" (or "Your") means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License. For legal entities, "You" includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the

direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity. 2. License Grants. 2.1. The Initial Developer Grant. Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, the Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license: (a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer, to use, reproduce, modify, display, perform, sublicense and distribute the Original Software (or portions thereof), with or without Modifications, and/or as part of a Larger Work; and (b) under Patent Claims infringed by the making, using or selling of Original Software, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Software (or portions thereof). (c) The licenses granted in Sections 2.1(a) and (b) are effective on the date Initial Developer first distributes or otherwise makes the Original Software available to a third party under the terms of this License. (d) Notwithstanding Section 2.1(b) above, no patent license is granted: (1) for code that You delete from the Original Software, or (2) for infringements caused by: (i) the modification of the Original Software, or (ii) the combination of the Original Software with other software or devices. 2.2. Contributor Grant. Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license: (a) under intellectual property rights (other than patent or trademark) Licensable by Contributor to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof), either on an unmodified basis, with other Modifications, as Covered Software and/or as part of a Larger Work; and (b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: (1) Modifications made by that Contributor (or portions thereof); and (2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination). (c) The licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first distributes or otherwise makes the Modifications available to a third party. (d) Notwithstanding Section 2.2(b) above, no patent license is granted: (1) for any code that Contributor has deleted from the Contributor Version; (2) for infringements caused by: (i) third party modifications of Contributor Version, or (ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or (3) under Patent Claims infringed by Covered Software in the absence of Modifications made by that Contributor. 3. Distribution Obligations. 3.1. Availability of Source Code. Any Covered Software that You distribute or otherwise make available in Executable form must also be made available in Source Code form and that Source Code form must be distributed only under the terms of this License. You must include a copy of this License with every copy of the Source Code form of the Covered Software You distribute or otherwise make available. You must inform recipients of any such Covered Software in Executable form as to how they can obtain such Covered

Software in Source Code form in a reasonable manner on or through a medium customarily used for software exchange. 3.2. Modifications. The Modifications that You create or to which You contribute are governed by the terms of this License. You represent that You believe Your Modifications are Your original creation(s) and/or You have sufficient rights to grant the rights conveyed by this License. 3.3. Required Notices. You must include a notice in each of Your Modifications that identifies You as the Contributor of the Modification. You may not remove or alter any copyright, patent or trademark notices contained within the Covered Software, or any notices of licensing or any descriptive text giving attribution to any Contributor or the Initial Developer. 3.4. Application of Additional Terms. You may not offer or impose any terms on any Covered Software in Source Code form that alters or restricts the applicable version of this License or the recipients' rights hereunder. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, you may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer. 3.5. Distribution of Executable Versions. You may distribute the Executable form of the Covered Software under the terms of this License or under the terms of a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable form does not attempt to limit or alter the recipient's rights in the Source Code form from the rights set forth in this License. If You distribute the Covered Software in Executable form under a different license, You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer. 3.6. Larger Works. You may create a Larger Work by combining Covered Software with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Software. 4. Versions of the License. 4.1. New Versions. Oracle is the initial license steward and may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Except as provided in Section 4.3, no one other than the license steward has the right to modify this License. 4.2. Effect of New Versions. You may always continue to use, distribute or otherwise make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. If the Initial Developer includes a notice in the Original Software prohibiting it from being distributed or otherwise made available under any subsequent version of the License, You must distribute and make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. Otherwise, You may also choose to use, distribute or otherwise make

the Covered Software available under the terms of any subsequent version of the License published by the license steward. 4.3. Modified Versions. When You are an Initial Developer and You want to create a new license for Your Original Software, You may create and use a modified version of this License if You: (a) rename the license and remove any references to the name of the license steward (except to note that the license differs from this License); and (b) otherwise make it clear that the license contains terms which differ from this License. 5. DISCLAIMER OF WARRANTY. COVERED SOFTWARE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED SOFTWARE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED SOFTWARE IS WITH YOU. SHOULD ANY COVERED SOFTWARE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER. 6. TERMINATION. 6.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive. 6.2. If You assert a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You assert such claim is referred to as "Participant") alleging that the Participant Software (meaning the Contributor Version where the Participant is a Contributor or the Original Software where the Participant is the Initial Developer) directly or indirectly infringes any patent, then any and all rights granted directly or indirectly to You by such Participant, the Initial Developer (if the Initial Developer is not the Participant) and all Contributors under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively and automatically at the expiration of such 60 day notice period, unless if within such 60 day period You withdraw Your claim with respect to the Participant Software against such Participant either unilaterally or pursuant to a written agreement with Participant. 6.3. If You assert a patent infringement claim against Participant alleging that the Participant Software directly or indirectly infringes any patent where such claim is resolved (such as by license or settlement) prior to the initiation of patent infringement litigation, then the reasonable value of the licenses granted by such Participant under Sections 2.1 or 2.2 shall be taken into account in determining the amount or value of any payment or license. 6.4. In the event of termination under Sections 6.1 or 6.2 above, all end user licenses that have been validly granted by You or any distributor hereunder prior to termination (excluding licenses granted to You by any distributor) shall survive termination. 7. LIMITATION OF LIABILITY. UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY

OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED SOFTWARE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU. 8. U.S. GOVERNMENT END USERS. The Covered Software is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" (as that term is defined at 48 C.F.R. § 252.227-7014(a)(1)) and "commercial computer software documentation" as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Software with only those rights set forth herein. This U.S. Government Rights clause is in lieu of, and supersedes, any other FAR, DFAR, or other clause or provision that addresses Government rights in computer software under this License. 9. MISCELLANEOUS. This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by the law of the jurisdiction specified in a notice contained within the Original Software (except to the extent applicable law, if any, provides otherwise), excluding such jurisdiction's conflict-of-law provisions. Any litigation relating to this License shall be subject to the jurisdiction of the courts located in the jurisdiction and venue specified in a notice contained within the Original Software, with the losing party responsible for costs, including, without limitation, court costs and reasonable attorneys' fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License. You agree that You alone are responsible for compliance with the United States export administration regulations (and the export control laws and regulation of any other countries) when You use, distribute or otherwise make available any Covered Software. 10. RESPONSIBILITY FOR CLAIMS. As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability. NOTICE PURSUANT TO SECTION 9 OF THE COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) The code released under the CDDL shall be governed by the laws of the State of California (excluding conflict-of-law provisions). Any litigation relating to this License

shall be subject to the jurisdiction of the Federal Courts of the Northern District of California and the state courts of the State of California, with venue lying in Santa Clara County, California. The GNU General Public License (GPL) Version 2, June 1991 Copyright (C) 1989, 1991 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor Boston, MA 02110-1335 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. Preamble The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too. When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things. To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it. For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights. We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software. Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations. Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all. The precise terms and conditions for copying, distribution and modification follow. TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION 0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you". Activities other than copying, distribution and modification are not

covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does. 1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program. You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee. 2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions: a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change. b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License. c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.) These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it. Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program. In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License. 3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following: a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or, b) Accompany it with a written offer, valid for at least three years, to

give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or, c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.) The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable. If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code. 4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program 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. 5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it. 6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License. 7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program. If any portion of this section is held invalid or

unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances. It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice. This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License. 8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License. 9. The Free Software Foundation may publish revised and/or new versions of the General Public 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. Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation. 10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally. NO WARRANTY 11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. 12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE

PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. END OF TERMS AND CONDITIONS How to Apply These Terms to Your New Programs If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms. To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found. One line to give the program's name and a brief idea of what it does. Copyright (C) <year> <name of author> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335 USA Also add information on how to contact you by electronic and paper mail. If the program is interactive, make it output a short notice like this when it starts in an interactive mode: Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details. The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program. You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names: Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker. signature of Ty Coon, 1 April 1989 Ty Coon, President of Vice This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License. Certain source files distributed by Oracle America, Inc. and/or its affiliates are subject to the following clarification and special exception to the GPLv2, based on the GNU Project exception for its Classpath libraries, known as the GNU Classpath Exception, but only where Oracle has expressly included in the particular source file's header the words "Oracle designates this particular file as subject to the "Classpath" exception as provided by Oracle in the LICENSE file that accompanied this code." You should also note that Oracle includes

multiple, independent programs in this software package. Some of those programs are provided under licenses deemed incompatible with the GPLv2 by the Free Software Foundation and others. For example, the package includes programs licensed under the Apache License, Version 2.0. Such programs are licensed to you under their original licenses. Oracle facilitates your further distribution of this package by adding the Classpath Exception to the necessary parts of its GPLv2 code, which permits you to use that code in combination with other independent modules not licensed under the GPLv2. However, note that this would not permit you to commingle code under an incompatible license with Oracle's GPLv2 licensed code by, for example, cutting and pasting such code into a file also containing Oracle's GPLv2 licensed code and then distributing the result. Additionally, if you were to remove the Classpath Exception from any of the files to which it applies and distribute the result, you would likely be required to license some or all of the other code in that distribution under the GPLv2 as well, and since the GPLv2 is incompatible with the license terms of some items included in the distribution by Oracle, removing the Classpath Exception could therefore effectively compromise your ability to further distribute the package. Proceed with caution and we recommend that you obtain the advice of a lawyer skilled in open source matters before removing the Classpath Exception or making modifications to this package which may subsequently be redistributed and/or involve the use of third party software. CLASSPATH EXCEPTION Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License version 2 cover the whole combination. As a special exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. An independent module is a module which is not derived from or based on this library. If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each

Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

You must give any other recipients of the Work or Derivative Works a copy of this License; and

You must cause any modified files to carry prominent notices stating that You changed the files; and

You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS+

MIT License

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE

New BSD License (3-clause)

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of <company name> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL DIGITAL BAZAAR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES

(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

BSD License (2-clause)

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

GPL2 w/ CPE

The GNU General Public License (GPL) Version 2, June 1991

Copyright © 1989, 1991 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor Boston, MA 02110-1335 USA

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

Preamble

The licenses for most software are designed to take away your freedom to share and change

it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with

modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions

for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

    a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

    b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

    c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program 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.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public 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.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

    NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF

THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
One line to give the program's name and a brief idea of what it does.
Copyright (C) <year> <name of author>
```

```
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
```

```
This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
General Public License for more details.
```

```
You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335 USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type
`show w'. This is free software, and you are welcome to redistribute
it under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the

General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the
program `Gnomovision' (which makes passes at compilers) written by
James Hacker.
```

```
signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

CLASSPATH EXCEPTION

Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License version 2 cover the whole combination.

As a special exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. An independent module is a module which is not derived from or based on this library. If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.

Eclipse Distribution License - v 1.0

Copyright (c) 2007, Eclipse Foundation, Inc. and its licensors.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of the Eclipse Foundation, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Open for Innovation

KNIME

KNIME AG
Talacker 50
8001 Zurich, Switzerland
www.knime.com
info@knime.com