Administrators guide

cmsWorks is a general purpose content management system completely written in the programming language Java without any other dependencies regarding the OS.

It has an application server integrated which provides base services such as handling http-requests or connecting to a database. Thus you can run (and even develop for) cmsWorks on any operating system providing a Java installation.

cmsWorks as application server stack

cmsWorks is able to run as a complete stand-alone server stack without the need of third-party-software like webservers or an application stack like XAMP. This is especially helpful when you are developing your pages using cmsWorks.

Anyway, in real world environments it is best to integrate it into a server environment using MySQL as a database and a Webserver similar to Apache or nginx as a flexible to configure and, under certain circumstances, resource preserving delivering system in front of it.

Who should read this documentation?

This documentation is for persons who will administrate the cmsWorks installation. In here you will find a description of the installation and the possibilities to adapt and maintain the system. More concrete, this documentation contains

- Introduction and overview of the structures of cmsWorks
- Installation
- Customization of the editors user interface
- Integration of ContentScripts
- Data management and the administration therefore
- Administration of categories, navigation and configuration tasks
- Service administration and error analysis

Table of Contents

1 Installation

In normal cases, installing cmsWorks is as simple as copying the cmsWorks-files to a server. Anyway, you need at least a "server" (may be a client machine) and a database. This page is to give more information about it.

Prerequisites: Operating system and database

A brief introduction which software is needed on the server / development system to install cmsWorks.

Installing the cmsWorks software bundle

The installation of cmsWorks on a server / development system.

Preparing and connecting to MySQL

Describes the preparation for the MySQL database server by defining the right (UTF-8) encoding, creating the database and user, importing an empty standard database and connecting cmsWorks to MySQL.

Starting / stopping cmsWorks

Showing how to start and stop the cmsWorks server software.

File system structure of cmsWorks

A brief introduction to the file system structure cmsWorks resides in.

1.1 Prerequisites: OS and database

cmsWorks runs on every operating system that supports Java and JDBC-compliant databases, be it Microsoft Windows, Microsoft Windows Server or Linux /*ix etc. . The community license accepts JDBC-connections to MariaDB / MySQL and the built-in in-memory-database of cmsWorks.

The prerequisites to run cmsWorks are:

- An operating system that supports the OpenJDK 17 or higher downloaded on the cmsWorks development / server machine
- The internal cmsWorks in-memory-database or a JDBC compliant database, reachable at least via network (for the community license this would be MariaDB / MySQL)
- If MariaDB is used, it has at least to be version 10 or higher, if MySQL is used, it has at least to be version 8 or higher

In most cases, the use of a webserver in front of the cmsWorks application stack is recommended on productive systems (i.e. to handle SSL/TLS). See chapter Using a web server for static content and further configurations for more information about this.

1.2 Installing the bundle

Having the prerequisites, the next step is to install the cmsWorks software. Because cmsWorks has at first practically no dependencies to the system it is installed on, in most cases deflating the installation bundle of cmsWorks will suffice.

Installation

Using Linux or similar operating system, a dedicated user account should be created which then will be used to run the cmsWorks software stack.

In any case, the archive containing the software has to be deflated into a folder (from now on called <cmsWorks-installdir>). If you created a user for it be sure to login as this user to start or modify cmsWorks and its files.

Configuring the start script

Now the start-script <cmsWorks-installdir>/run/cmsworks.sh (Linux / *ix) or <cmsWorks-installdir>/run/cmsworks.bat (Windows) has to be customized. Alter the first line to the installation destination of

Java on the machine (i.e. SET JAVAHOME=C:\Program Files\Java\jdk_17 on Windows or JAVAHOME=/usr/java/jdk17) on *ix machines like Linux.

Now you are ready to start cmsWorks as described in chapter starting/stopping cmsWorks.

1.3 Preparing and connecting to MariaDB / MySQL

Although cmsWorks comes with a fully configured and ready-to-use built-in in-memory database, one of the first things you probably want to do is to change this built in database to a "standalone" database. The in-memory database is useful for first steps and testing cases, but when storing bigger amounts of content it is recommended that a MariaDB / MySQL database is used, the configuration of these two should be pretty similar.

Preparing MariaDB / MySQL

cmsWorks takes heavy use of the UTF8 encoding to ensure the correct output in international languages. MySQL should therefore be prepared for this encoding - feel free to use other encodings like utf8mb4.

Depending on your database configuration, you should insert the following lines into your my.cnf / my.ini file on the database server into the [mysqld] section (or similar configurations for MariaDB depending on it's version):

```
skip-character-set-client-handshake=1
collation-server=utf8_general_ci
character-set-server=utf8
```

Additionally, MySQL has to be prepared for larger packet sizes to store bigger BLOB entries. Add the following line into the [mysqld] section, too (or similar configuration for MariaDB depending on it's version):

```
max_allowed_packet=32M
```

In case you want to store larger BLOB entries into the database, feel free to set a higher value. After the changes were done, the MySQL-server has to be restarted.

Creating a database (MySQL example)

Connect to your database (i.e. via mysql -u root if the database server is on the same machine you are working on) and execute the following lines to create the database and database user:

```
CREATE DATABASE cmsworksdb DEFAULT CHARACTER SET = UTF8;
CREATE USER 'cmsworks'@'%' IDENTIFIED BY 'cmspass';
GRANT ALL ON cmsworksdb.* TO 'cmsworks'@'%';
FLUSH PRIVILEGES;
```

In case you use MariaDB adapt that to your needs. It is essential that you use your own database name, user an password, adjust them in the database.properties file accordingly.

Creating standard tables and the needed data to a newly created database

In case you have a standard MySQL dump for an empty cmsWorks database for your DB-flavour you are able to import a database using i.e. the mysql command line program provided by MySQL:

```
mysql -u root --default-character-set=utf8 cmsworksdb < cmsworksdb.dmp
```

If you do not have such a database dump please refer to the packaged bin-tools (database tools of cmsWorks) to create the tables and fill them with standard values.

Connecting cmsWorks to the database

cmsWorks internally uses a database service that stores the user, password, servername etc. of the database and provides the system with database connections to that database.

A description of how to set the database parameters and configure the database connection pool of cmsWorks will be found in chapter The database connection service.

cmsWorks database tools

The following commands uses the existing database, creates the tables therein and prefills rows with the standard values.

```
dbinitcms.sh <connection protocol> <server/ip> <port> <database name> <user> <password>
```

To initially fill root folder, admin user, groups and more use

```
dbexecscript.sh <script path> <connection protocol> <server/ip> <port> <database name> <user> <password>
```

Attention! In case you use MariaDB or MySQL, make sure that the encoding is completely set to UTF-8. You may force this by adding "?useUnicode=true&characterEncoding=utf-8" to the database name in the configuration or while using the database tools.

1.4 Starting/stopping cmsWorks

Starting the server

The starting scripts reside in the directory <cmsWorks-installdir>/run/ and are named cmsworks.sh (Linux / *ix) or cmsworks.bat (Windows). These scripts start the application server and all services that are listed in the file <cmsworks-installdir>/run/batches/cmsstart.batch.

Check the cmsWorks status after start

To check the status of a newly started cmsWorks instance one may login to the telnet server shell and use the command "service status". This will list all available (micro-)services in cmsWorks and show their status ("starting", "running", "stopping", "stopped"). If all services are running, the server state should be "fully functioning".

In case problems occurred, the command "exception" used in the telnet server shell may give hints of the nature of these problems. The command "log" can also redirect the cmsWorks logging to the shell in real-time. The commands "exception" and "log" are described in detail in logging for debugging.

Stopping the server gracefully

Normally, it is not necessary to restart these services or even the complete application server. If a restart is desired, the server can be stopped via the cmsWorks built-in telnet server. To accomplish this, telnet has to be installed on the machine you connect from. The following sequence will then stop the server (use the correct port as given in the topas.properties file):

```
$ telnet localhost 8050
This is topas-server "cmsWorks server instance".
password: *******
cms# email off
cms# shutdown
```

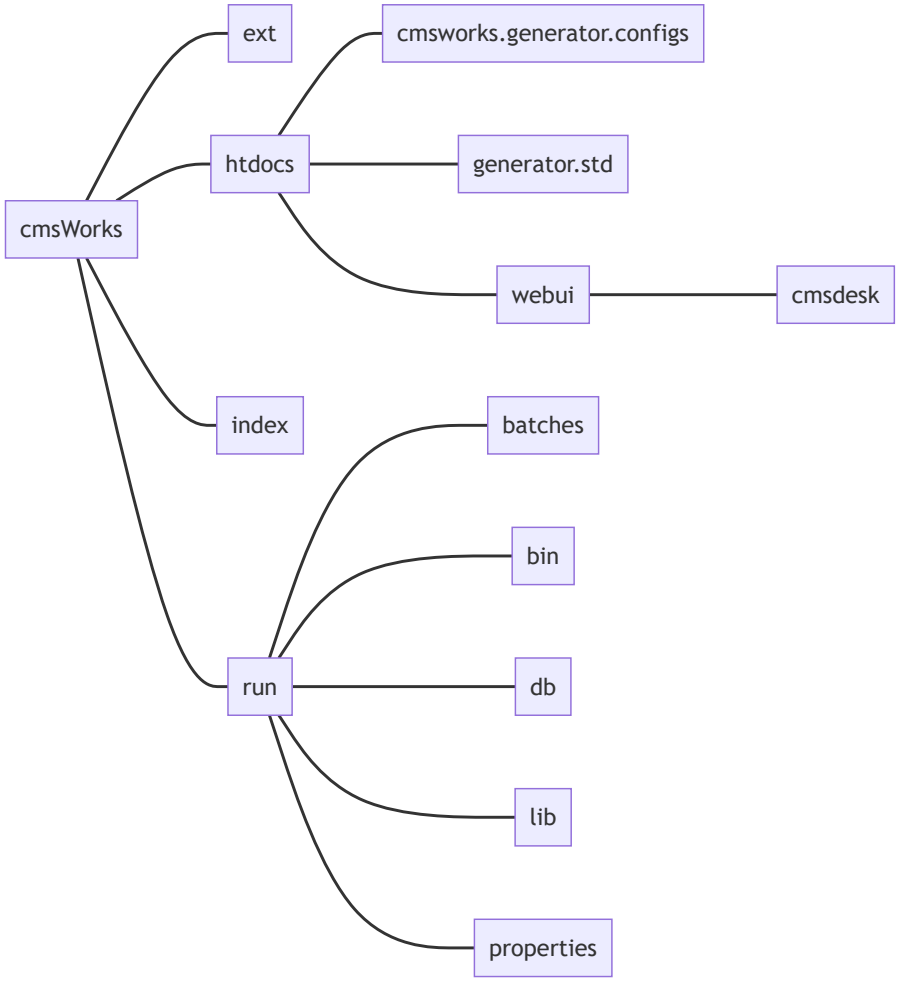
This example points to "telnet localhost 8050" and therefore assumes that the cmsWorks server is installed on the same machine. If this is not the case, "localhost" has to be replaced with the IP or name of the machine the cmsWorks server is installed at. Additionally, the port (8050) may have been altered, too.

Of course, if no internal database server service is running within cmsWorks, you could simply kill the java process of cmsWorks to shut down the complete server. If an internal database server service is running in cmsWorks, it is highly advisable to shutdown the server via the telnet server shell, so that the database is closed correctly.

1.5 Filesystem structure of cmsWorks

Having a server to install on, cmsWorks resides on that server in a simple file system structure.

Following a list of the directories and subdirectories and their purpose.



The filesystem structure of cmsWorks

2 Introduction to services

The kernel of cmsWorks consists of several so-called "services" (or "microservices") residing in the cmsWorks application stack which interact with each other or even the outer world.

Every one of these services has its own function, i.e. to perform search queries on the content, provide the editors desktop or even the connections to the database.

As a first example a GeneratorService will be described here.

A GeneratorService

- accepts http-requests and handles them
- provides methods to build URLs
- provides access to the CMS-service to JSPs so that they can display stored content
- provides methods to include (html) content from other sources
- is specialized within the task it has to fulfill (i.e. there is a preview generator and a live generator which deliver specialized views on the stored content, but due to the html-nature of the editors user interface, a specialized generator (WebUI-service) generates the cmsWorks desktop, too)

This way, different services serve different functions and to achieve this, they have to interact with the CMS-service (CMSCore). The CMS-service

- provides methods to access folders, documents and their content
- all other services (i.e. the GeneratorServices) who want to access these information have to access the CMS-service to get them

To obtain information, the CMS-service has to read and write them from and to a database. Responsible for the connection to the database is the database service (CMSDB).

A list of available services in cmsWorks - and services in general - are discussed in more detail in the chapter Services and their configuration.

3 Configuration

There are six different ways to configure cmsWorks so that it works properly:

- The configuration of the virtual machine parameters of the Java Virtual Machine
- The configuration of the cmsWorks global parameters
- The configuration of the startup behavior of the cmsWorks server in so-called batch-files
- The parameterization of the services which is done in property files
- The configuration of the cmsWorks desktop done via CSS/Javascript of the WebUI-service
- The configuration of components via specialized documents stored in the database

These different configurations are described in the chapters Startup behavior configuration, Global parameters of cmsWorks and Services and their configuration.

3.1 Startup behavior configuration

Installing cmsWorks is pretty simple but a little bit of knowledge is necessary:

1. You need to have a login/access to the machine on which you want to install cmsWorks at (i.e. to transfer files to or start cmsWorks from the command prompt).
2. This server machine needs a Java JVM already installed.
3. In best cases, at least for a productive system, a database server (MySQL) needs to be installed and a newly created (empty) database is needed.

Startup behaviour then is relying on three base mechanisms and tests:

1. When starting the cmsWorks server software, a cmsWorks batch called "startup.batch" will be executed, all your services will be fired up by this batch (or an included batch within the startup.batch).
2. cmsWorks will automatically test the compiled Java classes if there is something missing or not, please refer to the "exception.log" file (in the starting folder), if cmsWorks did not start up correctly or, rather, exits the startup with an error.
3. In case everything worked out fine, login to the cmsWorks desktop and open up the cmsWorks-Explorer to preview any previously entered page.

The startup behaviour of cmsWorks depends on the startup of the Java Virtual Machine. In practice, you have to configure the memory settings so that cmsWorks can complete it's jobs.

Memory settings of the Java Virtual Machine

The memory settings of the virtual machine may be altered. Therefore, the two entries -Xms**M and -Xmx**M have to be set (in the cmsworks.sh or cmsworks.bat scripts) to a value corresponding to the later needed use of the overall memory usage.

The memory setting is one of the few settings that cannot be altered when the virtual machine is running. Altering these values requires a restart of cmsWorks.

How to set these values to get the best performance from cmsWorks will be discussed in chapter Optimization by using the JVM.

startup.batch

The batch file "startup.batch" is the first cmsWorks-batch-file loaded and executed when the server starts. It defines the order and count of the services that will be started. In case an other startup behaviour is intended, the startup batch may be modified or extended with telnet server commands.

3.2 Global parameters of cmsWorks

The global parameters for cmsWorks can be found in the properties-file <cmsWorks-installdir>/run/topas.properties. The exception to this rule are the RemoteHosts and CMSIdentification in the file

<cmsWorks-installdir>/run/properties/cms.properties and the email properties which can be found at <cmsWorks-installdir>/run/properties/email.properties.

The telnet server port

To start up, cmsWorks needs at least one port on the host machine, otherwise starting will fail with an error. The port's standard value is 8050, if this port is taken on the server, it's value can be altered in property-file <cmsWorks-installdir>/run/topas.properties, the parameter name is /com/itechworks/topas/server/telnet/Port. The parameter /com/itechworks/topas/server/telnet/Password that contains the telnet-server password (default: "topas") should definitely be altered at the beginning, too.

cmsWorks server instance, customer folder, remote preview port and root folder name

You should change the name of the cmsWorks server-instance in the topas.properties - file by altering the line /com/itechworks/topas/server/name=<cmsWorks-instance-name>, thereby allocating a unique <cmsWorks-instance-name>.

Additionally, you should change the parameter "Customer" in the file <cmsWorks-installdir>/run/properties/cms.properties (cms.properties) at /app/cmsworks/service/cms/CMSCore/Customer=<myCustomerName> to a valid yet short customer name without special chars (i.E. whitespace or "/"). This is necessary for the UI customization, so please create the folder <cmsWorks-installdir>/htdocs/webui/cmsdesk.custom.<myCustomerName> too.

The CMSIdentification parameter (/app/cmsworks/service/webui/CMSWebUI/CMSIdentification) in the file cms.properties let you change the name of the absolute root folder in the cmsWorks explorer.

It is not necessary to restart the whole server after the parameters "CMSIdentification" and/or "RemoteHosts" were changed, a simple restart of the CMSWebUI service from the telnet server via "service restart CMSWebUI" will do the job. But remember to logout and re-login to the cmsWorks editors desktop so that the changes can apply.

In case the preview and the cmsWorks editors desktop are running on different (sub)domains, the parameter /app/cmsworks/service/webui/CMSWebUI/ForcedHost=<domain-of-the-cmsWorks-desktop> can be used, so that the preview knows the correct location of the cmsWorks editors desktop and the UI-link functions work properly.

Email parameters

The cmsWorks-server, especially the services of cmsWorks are capable to send emails in case a not solvable problem may occur (i.e. the database is not reachable, memory ran out etc.). To be able to send such mails, a reachable, external SMTP mail server is needed and the email configuration has to be done correctly for an existing email-address.

The parameters are:

Entry	Description
/com/topasworks/server/email/Sender	The name of the sending mail account
/com/topasworks/server/email/Recipients	Recipients of mails, separated by a semicolon
/com/topasworks/server/email/SMTPServerConfigurationProperties	Path to a properties file containing all parameters SMTP needs for sending emails
/com/topasworks/server/email/MaxRetries	Retry count if initial mail sending fails
/com/topasworks/server/email/SendWaitTime	Retry waiting time if sending failed
/com/topasworks/server/email/EmailShutdownTimeout	Timeout to wait before shutdown to send mails

If the parameters are set correctly and the cmsWorks-server is restarted, you may test email sending via the telnet command 'email send "Test subject" "Test text"'.

The "SMTPServerConfigurationProperties" points to a file having the credentials for SMTP mailing. For a host like, for example, "smtp.myserver.com" the parameters would look like:

Entry	Description
mail.smtp.host	The smtp host domain, like "smtp.myserver.com"
mail.smtp.port	The smtp port of the SMTP server, i.e. 465
mail.smtp.ssl.trust	In case you need to "trust", the domain of the trusted server, like "smtp.myserver.com"
mail.smtp.starttls.enable	If starttls shall be enabled
mail.smtp.ssl.enable	If SSL is enabled on your SMTP server
mail.smtp.auth	In case the server demands transport of SMTP over SSL/TLS (true or false)
account.user	Normally analogous to "/com/topasworks/server/email/Sender", see above
account.password	The password the user account needs to send mails.

So in this case the smtp property file assembled would look like this (replace the values in the angle brackets through your own configuration values and drop the brackets):

```
mail.smtp.host=<smtp.myserver.com>
mail.smtp.port=465
mail.smtp.ssl.trust=<smtp.myserver.com>

mail.smtp.starttls.enable=true
mail.smtp.ssl.enable=true
mail.smtp.auth=true

account.user=<sendingusermailaddress>@<cmsworksserver>.com
account.password=<password>
```

It is not necessary to restart the server, if not desired. Instead login to the telnet server of cmsWorks and use the email command to set the correct parameters. The command "help email" shows the necessary commandline options for the email-command.

You may alter the parameters on the fly, but do not forget to edit the "emails.properties" file accordingly to persist the changes for the next server restart.

3.3 Services and their configuration

The cmsWorks application server consists of a variety of services, each one serving a special purpose like the preview of contents, connections to a database and so forth. See the "cmsWorks service list" for more information about the services and their fields of activity/functionality.

Every service has a bundle of parameters that define the behavior of the service. Any of these parameters for service parameterization are found in the folder <cmsWorks-installdir>/run/properties and are referenced in the properties file <cmsWorks-installdir>/run/topas.properties with their name, leaded by a #include-statement.

For example in topas.properties:

```
#include properties/email.properties
# ----------------------------------------------------------------
# Including properties for different types of tasks for the cms
# package.
# ----------------------------------------------------------------

#include properties/database.properties
#include properties/cms.properties
#include properties/generator.properties
#include properties/scriptserver.properties
#include properties/search.properties
```

Description how services work in cmsWorks

Every service serves a special, well defined task, his field of activity. Services may talk to each other or better: services may take use of each other.

The CMSCore-service, for example, is the interface to obtain content resources or folders (the resources are structured in). Therefore it has to access the database via the database service CMSDB: The CMSDB-service provides database connections and database connection pooling.

The GeneratorPreview-service which renders previews of output (i.e. a HTTP page) needs to obtain resources from the CMSCore service, which eventually asks the CMSDB service for a database connection to get the resource from the database and so forth.

cmsWorks service list

Following a list of (standard) services in cmsWorks with their service name and a description of the field of activity/functions these services occupy.

Service name	Description
CMSDB	Service that connects to a database the content is stored in
CMSCore	Base service and interface for requesting information about content and caching these informations internally
CMSPublished	Service beeing a CMS service only returning published resources (omitting resources not published)
CMSWebUI	Providing the cmsWorks editors desktop user interface for the browser
GeneratorPreview	Preview generator produces a website based on all available documents with their current contents
GeneratorLive	Generator produces a website using only the latest published version of documents
ScriptServer	Service that provides the cmsWorks scripting engine
Search	Search engine managing search indexes
SearchCollectorWebUI	Collector service that passes content into the search-service for the WebUI
SearchCollectorOnline	Collector service that passes content into the search-service for the online site

Detailed configuration options and how to change the configurations of these services are explained in the following chapters.

3.3.1 Dependencies of services

Each service in cmsWorks serves a specialized task, most services rely on one or more services to fulfill their work.

Following services, showing their hierarchical dependencies from each other, are delivered with cmsWorks (from base level 1 up to level 5):

```
1 CMSDBServer
    2 CMSDB
        3 CMSCore
            4 CMSPublished
            4 CMSWebUI
            4 GeneratorLive
            4 ScriptServer
            4 Search
                5 SearchCollectorWebUI
                5 SearchcollectorOnline
```

A service without dependencies is the Database service only communicating with a database. Also, the in-memory database server service CMSDBServer (which is optional) does not rely on other services.

The CMSDB service provides a database connection pool to access content data from a database. Relying on this service, the CMSCore service provides functionality to read, write, modify an query that content informations, store userdata, blobs, mimetype information etc. . It is the base service and the heart of cmsWorks. All other services rely on CMSCore to fulfill their respective tasks.

3.3.2 Configuring services with property files

Services are configured via property-files residing in the folder <cmsWorks-installdir>/run/properties. cmsWorks has preconfigured services at delivery status after installation. The property bundles for a service always consists of

- the package of the service, followed by the separator "/",
- the service name itself, followed by the separator "/",
- the parameter name, followed by "=" and
- the parameter value

in one line. As example, one such line for the database service CMSDB consists of

- "/com/topasworks/kernel/service/database/" (the package)
- "CMSDB/" (the service name)
- "JDBCKey=" (the parameter name)
- "jdbc:mysql" (the parameter value)

in one line:

```
/com/topasworks/kernel/service/database/CMSDB/JDBCKey=jdbc:mysql
```

The property files contain descriptions for each service parameter and its functionality.

All services properties

Every service has by default some properties about starting and stopping the service. The server manager takes care of the running services and checks their states. Some services will take more more less time to allocate the needed resources or while stopping to close all used resources. So these times have to be configured for every service.

Entry	Value	Description
StartTimeout	1000	Time in milliseconds. After starting the service this is the time the server waits for checking that the service is running. If not, the service will be stopped.
StopTimeout	1000	Time in milliseconds. After stopping the service this is the time the server waits for checking that the service is stopped. If not, the service will be stopped forcefully.
AccessTimeout	1000	Time in milliseconds.
StopInformExtern	1	If the service was expectantly stopped this switch enables/disables to send an email to the configured email receivers for administration concerns reporting that this service was stopped. For instance if a database server is not available any more, the database service will stop and inform the administrators.
LogLevel	fatal error info debug	This is a list of log level names from the full set of [fatal error warn info debug trace] or all if all should be active. To disable some log levels is coming handy if the debug logging is very verbose but the service in a live server is running very reliable. For instance when logging all content, that a search service is indexing or searching for it creates so much data that nobody is going to examine it anyway. For finding some issues you can always turn on and off the logging without stopping the service.

Some services are derived AbstractHttpServer services (JSP container services). They have also some properties in common which will not be explained in detail in the actual services.

Entry	Value	Description
Port	8080	A JSP container service is a webservice that can be accessed by an HTTP request. So this service opens the port configured here to receive the request.
Htdocs	../htdocs/<servicepath>	This is the path in the file system where the static files and the JSP files are located. On a request the JSP container service will compile the jsp file to a Servlet class. Than it creates an instance of this class and executes this instance with the request. All further requests for this JSP will be processed by this instance multi threaded. If the JSP file is changed, the next request will start over compiling and executing the new instance of the Servlet. The path is relative to the JVM-Engine-Start-Folder <cmsWorks-installdir>/run.
DirectoryListAllowed	true	If set to false no filesystem content is delivered if the requesting URL targets a filesystem directory.
SessionsAllowed	true	If set to false all JSPs activating a Session cannot be compiled any more.
RelativeRedirectAllowed	true	true is the default. Otherwise the jetty creates absolute URLs when sending response.sendRedirect().
ResponseHeaderServer	<text>	Returns the license information on default as response header "Server".
ErrorResponseFooter	<text>	Returns the license information as a footer under the error message.
ErrorResponseCss	<text>	Replace the default css if more fancy error pages should be created.
MaxFormContentSize		A negative value is default. With this the Jetty handles it's predefined value. "max" is a value for unlimited form content size. Otherwise use a value in byte.

3.3.3 The database connection service

All content entered or uploaded into cmsWorks is stored in a database, this chapter describes how the parameters have to be configured so that the database is accessible for the CMS.

The service "CMSDB" manages a connection pool that holds connections to the configured database.

Property file and property key paths are:

Service properties file

<cmsWorks-installdir>/run/properties/database.properties

Service property key paths

/com/topasworks/kernel/service/database/CMSDB/

Entry	Value	Description
JDBCKey	jdbc:mysql	for MariaDB / MySQL
DatabaseName	cmsdb	The MariaDB / MySQL-name of the database
HostName	127.0.0.1	IP or host name of the db-server-machine
User	dbuser1	Username of the database (dbuser1 is an example)
Password	secretpw	Password of the database user (secretpw is an example)
Port	3306	Port the database listens to
MinConnections	4	The minimum count of connections that are established when starting the Database service, this value should be 4 or higher.
MaxConnections	20	The maximum count of connections that can be created while using the Database service. When using the Database service in multiple Threads (Parallel processing in JAVA) and the maximum count of connections has been reached, the next Thread has to wait until a connection is not in use any more.
AutoCommit	0	cmsWorks handles itself transactional behavior in the database and commits the changes when done, so AutoCommit is set to "false".
PingInterval	5000	Established connections to the database may be closed from the database side. So this is the time in milliseconds after that the Database service will send a connection keep alive query.
PingSelect	SELECT MIN(1) FROM DUALS	This is the query to keep alive the established database connections, the table DUALS exists in cmsWorks databases.
ConnectionTrace	0	JDBC connection tracing is turned off in cmsWorks by default.

Here is a full example of the Database service configuration:

```
# ##############################################################################
#
# Properties for the cms database service
#
# ##############################################################################
# service create /com/topasworks/kernel/service/database,DatabaseService,CMSDB

/com/topasworks/kernel/service/database/CMSDB/StartTimeout=10000
/com/topasworks/kernel/service/database/CMSDB/StopTimeout=10000
/com/topasworks/kernel/service/database/CMSDB/AccessTimeout=5000
/com/topasworks/kernel/service/database/CMSDB/StopInformExtern=1

/com/topasworks/kernel/service/database/CMSDB/JDBCKey=jdbc:mysql
/com/topasworks/kernel/service/database/CMSDB/Port=3306
/com/topasworks/kernel/service/database/CMSDB/DatabaseName=cmsdb
/com/topasworks/kernel/service/database/CMSDB/HostName=127.0.0.1
/com/topasworks/kernel/service/database/CMSDB/User=dbuser1
/com/topasworks/kernel/service/database/CMSDB/Password=secretpw

/com/topasworks/kernel/service/database/CMSDB/MinConnections=1
/com/topasworks/kernel/service/database/CMSDB/MaxConnections=10
/com/topasworks/kernel/service/database/CMSDB/AutoCommit=0
/com/topasworks/kernel/service/database/CMSDB/PingInterval=5000
/com/topasworks/kernel/service/database/CMSDB/PingSelect=SELECT MIN(1) FROM DUALS
/com/topasworks/kernel/service/database/CMSDB/ConnectionTrace=0
```

Configuration example of the Database service

In case you set the parameters correctly the database service can be started using the telnet server shell (if the database service is running "service restart CMSDB", otherwise simply use the command "service start CMSDB") or restart the cmsWorks-server so that it starts the service on its own.

In case the parameters are not correct an error message on the console will be printed which indicates the configuration error.

3.3.4 CMSCore service

The CMS service called CMSCore is the service managing all CMS resources. Those resources are the documents, folders, user, usergroups and usage privileges. This service should be the only one using the database service to store into and read from the database. It implements a change event API to allow change listeners to receive messages about changes in documents and folders. So a SearchCollector service can receive a document changed event and update the search index. Also the WebUI service listens to changes and updates the editors desktop.

For the best performance the CMS service holds the mostly needed resources in different memory caches so the access to them is as fast as possible. But there is never enough RAM compared to the capacities of database storage so the caches have to be restricted by the count of entries or memory usage. Some of the configuration keys are there to configure those cache sizes.

Property file and property key paths are:

Service properties file	<cmsWorks-installdir>/run/properties/cms.properties
Service property key paths	/app/cmsworks/service/cms/CMSCore/

Entry	(Standard) Value	Description
CMSDBName	CMSDB	The name of the database service.
AllowedSpecialCharsForNamingFolders	../htdocs/generator.std/	Folder names and document names are basically just some strings. But using them in URLs is restricting them to standard characters of the alphabet plus digits and the special characters '-._' without spaces which is mostly SEO relevant. With this list parameter more special characters may be allowed for folder names.
LangMessagesFilePath	../langmsg/	The CMSService provides translations for labels, terms and phrases that are used in the editors desktop or in content scripts. So this is the location of the translation files This is the name of the service that provides access to CMS documents
ResourceCacheSize	100000	The maximum count of cached documents. Reading a document from the database is quite expensive (it is slow and produces load on the database). The delivery of a document from cache is done is almost no time. But document contents sizes are very different so the count of documents in the cache has to be limited. There is no mechanism implemented yet that calculates the memory usage of a document so we have to stay with the entry count of the cache as limitation. While generating a website many documents will be used (many links for header- and footer-navigation, components, searchresults, configuration documents in category trees, ...) The documents of the most often called websites will stay long in the cache while other rarely used documents will be kicked from the cache.
ResourceCacheTypeLimits		Only these root paths are enabled as CMS folders where documents can be found to be generated. These paths have to be absolute beginning from root /
PublishedResourcesCacheSize	1024	The maximum count of cached published versions of documents that have additionally edited versions on top.
ResourceNameCacheSize	1000000	The query to retrieve all documents from a folder is quite expensive. Mostly not all documents are need just the names of those documents when accessing one by name or creating a new or renaming. The access to names of documents of a folder is supported by a ResourceNameCache. This key sets the maximum count of cached document names of folders. This cache is also quite often used when finding the document for an URL request.

BlobCacheSize	10000	The maximum count of cached blob values from documents blob properties. In most cases this cache provides fast access to pictures of the currently most called pages of the website. This cache uses by far the most RAM. Just limiting the RAM usage by the count of entries is not good enough when the cache entries are of different sizes. The key BlobCacheMemory also limits the BlobCache size.
BlobCacheMemory	1500	The maximum memory usage in Megabyte the BlobCache can hold in memory.
Customer	mywebsite	This name is used for subfolder names for editors desktop configuration (<WebUIs Htdocs path>/cmsdesk.custom/<Customer>) and content scripts (<ScriptServers Htdocs path>/<Customer>).
ParallelStoreReadingCount	10	The count of parallel threads that are allowed to read from the database.
LockResourceTimout	180000	Only one user can edit a document at a time. But if the connection to the editors desktop is broken this is the time (in milliseconds) the documents stays locked. After that the lock is released and any other user can edit this document now. The latest stored version of the content stays.

Here is a full example of the CMSCore configuration:

```
# ##############################################################################
#
# Properties for the cms kernel service
#
# ##############################################################################

/app/cmsworks/service/cms/CMSCore/StartTimeout=1000000
/app/cmsworks/service/cms/CMSCore/StopTimeout=10000
/app/cmsworks/service/cms/CMSCore/AccessTimeout=1000
/app/cmsworks/service/cms/CMSCore/StopInformExtern=1
/app/cmsworks/service/cms/CMSCore/LogLevel=fatal error info debug

/app/cmsworks/service/cms/CMSCore/CMSDBName=CMSDB
/app/cmsworks/service/cms/CMSCore/AllowedSpecialCharsForNamingFolders=
/app/cmsworks/service/cms/CMSCore/LangMessagesFilePath=../langmsg/
/app/cmsworks/service/cms/CMSCore/ResourceCacheSize=40500
/app/cmsworks/service/cms/CMSCore/ResourceCacheTypeLimits=
/app/cmsworks/service/cms/CMSCore/PublishedResourcesCacheSize=1024
/app/cmsworks/service/cms/CMSCore/ResourceNameCacheSize=100000
/app/cmsworks/service/cms/CMSCore/BlobCacheSize=10000
/app/cmsworks/service/cms/CMSCore/BlobCacheMemory=1500
/app/cmsworks/service/cms/CMSCore/Customer=mywebsite
/app/cmsworks/service/cms/CMSCore/ParallelStoreReadingCount=10
/app/cmsworks/service/cms/CMSCore/LockResourceTimeout=180000
```

Configuration example of the CMSCore service, the service handling all CMS resources

3.3.5 CMSPublished service

The CMSPublished service implements the complete API of the CMSService but for getting documents it only delivers the latest published versions of the documents. But there is also a config key that allows a temporary switch to access all current non published versions of the documents if needed.

The service mainly is used by Generator services generating only published document contents.

Property file and property key paths are:

Service properties file <cmsWorks-installdir>/run/properties/cms.properties

Service property key paths

/app/cmsworks/service/cms/CMSPublished/

Entry	Value	Description
Preview	false	Normally this is set to false to meet the purpose of this service delivering only the latest published versions of documents. If set to true, it's just working like the CMSService.
CMSServiceName	CMSCore	This is the name of the service that provides access to CMS documents.

Here is a full example of the CMSPublished configuration:

```
# ##############################################################################################
#
# Properties for the service giving a generator service a view of only published resources
#
# ##############################################################################################
# hook service   /app/cmsworks/service/cms,CMSGeneratorView
# service create /app/cmsworks/service/cms,CMSGeneratorView,CMSPublished

/app/cmsworks/service/cms/CMSPublished/StartTimeout=10000
/app/cmsworks/service/cms/CMSPublished/StopTimeout=10000
/app/cmsworks/service/cms/CMSPublished/AccessTimeout=1000
/app/cmsworks/service/cms/CMSPublished/StopInformExtern=1
/app/cmsworks/service/cms/CMSPublished/LogLevel=fatal error info

/app/cmsworks/service/cms/CMSPublished/Preview=false
/app/cmsworks/service/cms/CMSPublished/CMSServiceName=CMSCore
```

Configuration example of the CMSPublished Service an API-Filter for only published documents

3.3.6 WebUI service

The WebUI service is a webservice providing the login for the editors desktop.

Property file and property key paths are:

Service properties file

<cmsWorks-installdir>/run/properties/cms.properties

Service property key paths

/app/cmsworks/service/webui/CMSWebUI/

Entry	Value	Description
Port	8082	The editors desktop is accessable over this port of the IP or Hostname of the server the cmsWorks instance is running on.
Htdocs	../htdocs/webui	This is the path in the filesystem where the static files and the jsp-files are located.
CMSServiceName	CMSCore	This is the name of the service that provides access to all CMS resources like documents, folders, users, groups, privileges and others
CMSIdentification	mywebsite	This is the name that is shown as the root path of all folders and documents in the editors desktop explorer.
RemoteHosts	search=localhost:8091; scriptserver=localhost:8093	The editors desktop uses other services for some features. They are called via HTTP-Request so this is the place to address the other services. • search the location of the Search service, Its called to search for documents • scriptserver is the location of the ScriptServer service where contentscripts are running
PicDim	1200x694, 940x347, 600x347, 298x171	This is the list of dimensions of pictures that are proposed if using the pictures editor in the blob input field of a document.

Here is a full example of the CMSWebUI configuration:

```
# ##############################################################################
#
# Properties for editors web user interface service
#
# ##############################################################################
# hook service   /app/cmsworks/service/webui,WebUIService
# service create /app/cmsworks/service/webui,WebUIService,CMSWebUI

/app/cmsworks/service/webui/CMSWebUI/StartTimeout=10000
/app/cmsworks/service/webui/CMSWebUI/StopTimeout=10000
/app/cmsworks/service/webui/CMSWebUI/AccessTimeout=1000
/app/cmsworks/service/webui/CMSWebUI/StopInformExtern=1
/app/cmsworks/service/webui/CMSWebUI/LogLevel=fatal error info debug

/app/cmsworks/service/webui/CMSWebUI/Port=8082
/app/cmsworks/service/webui/CMSWebUI/Htdocs=../htdocs/webui

/app/cmsworks/service/webui/CMSWebUI/CMSServiceName=CMSCore
/app/cmsworks/service/webui/CMSWebUI/CMSIdentification=My Website
/app/cmsworks/service/webui/CMSWebUI/RemoteHosts=search=localhost:8091;scriptserver=localhost:8093
/app/cmsworks/service/webui/CMSWebUI/PicDim=1200x694,940x347,600x347,298x171
```

Configuration example of the WebUI service enabling the editors desktop

3.3.7 Generator services

The generator service is a webservice that can serve files from filesystem, that can execute servlets compiled from jsp files and has access to the documents of the CMS service.

The GeneratorPreview is configured to have a preview of the web page before publishing the content. It also adds some features to help the editor finding the document to edit in or showing errors why the documents are not filled properly. Also it provides direct editing in the preview website.

The GeneratorLive is configured to show only published content.

Property file and property key paths are:

Service properties file

<cmsWorks-installdir>/run/properties/generator.properties

Service property key paths

/app/cmsworks/service/generator/GeneratorPreview/
/app/cmsworks/service/generator/GeneratorLive/

Entry	Value	Description
Port	8080	The preview page is accessable over this port of the IP or Hostname of the server the cmsWorks instance is running on.
Htdocs	../htdocs/generator.std/	This is the path in the filesystem where the static files and the jsp-files are located. The name ending with .std indicates that this is the standard preview of the website.
CMSServiceName	CMSCore	This is the name of the service that provides access to CMS documents
Preview	true	True if the documents from CMS may be only changed and not published yet. This switch also acitvates Features using UILinks and Errors presentation
GenerationFolders	/en/;/de/	Only these root paths are enabled as CMS folders where documents can be found to be generated. These paths have to be absolute beginning from root /
GenerationConfigFile	../htdocs/cmsworks.generator.configs/generatorconfig-std.txt	The content of this file configures which document types are allowed to be generated in combination with a contenttype defined by the extension (i.e. .html)
IncludeHosts	search=localhost:8091	If a generation needs to call other HTTP-Server instances (i.e. a search request) the servers host and port can be listed here. With this configuration the same JSP may call different hosts in different generators (with different configurations) or in different cmsWorks instances like dev/test/live instances.

Here is a full example of the generator configuration:

```
# ##############################################################################
#
# Properties for the preview generator service
#
# ##############################################################################
# hook service   /app/cmsworks/service/generator,Generator
# service create /app/cmsworks/service/generator,Generator,GeneratorPreview

/app/cmsworks/service/generator/GeneratorPreview/StartTimeout=10000
/app/cmsworks/service/generator/GeneratorPreview/StopTimeout=10000
```

```
/app/cmsworks/service/generator/GeneratorPreview/AccessTimeout=1000
/app/cmsworks/service/generator/GeneratorPreview/StopInformExtern=1
/app/cmsworks/service/generator/GeneratorPreview/LogLevel=fatal error info debug

/app/cmsworks/service/generator/GeneratorPreview/Port=8080
/app/cmsworks/service/generator/GeneratorPreview/Htdocs=../htdocs/generator.std/

/app/cmsworks/service/generator/GeneratorPreview/CMSServiceName=CMSCore
/app/cmsworks/service/generator/GeneratorPreview/Preview=true
/app/cmsworks/service/generator/GeneratorPreview/GenerationFolders=/en/;/de/
/app/cmsworks/service/generator/GeneratorPreview/GenerationConfigFile=../htdocs/cmsworks.generator.configs/generatorconfig-st
d.txt
/app/cmsworks/service/generator/GeneratorPreview/IncludeHosts=search=localhost:8091


# ##############################################################################
#
# Properties for the live generator service
#
# ##############################################################################
# # the service was hooked before
# service create /app/cmsworks/service/generator,Generator,GeneratorLive

/app/cmsworks/service/generator/GeneratorLive/StartTimeout=10000
/app/cmsworks/service/generator/GeneratorLive/StopTimeout=10000
/app/cmsworks/service/generator/GeneratorLive/AccessTimeout=1000
/app/cmsworks/service/generator/GeneratorLive/StopInformExtern=1
/app/cmsworks/service/generator/GeneratorLive/LogLevel=fatal error info

/app/cmsworks/service/generator/GeneratorLive/Port=8081
/app/cmsworks/service/generator/GeneratorLive/Htdocs=../htdocs/generator.std/

/app/cmsworks/service/generator/GeneratorLive/CMSServiceName=CMSPublished
/app/cmsworks/service/generator/GeneratorLive/Preview=false
/app/cmsworks/service/generator/GeneratorLive/GenerationFolders=/en/;/de/
/app/cmsworks/service/generator/GeneratorLive/GenerationConfigFile=../htdocs/cmsworks.generator.configs/generatorconfig-std.t
xt
/app/cmsworks/service/generator/GeneratorLive/IncludeHosts=search=localhost:8091
```

Configuration example of the Generator services for website generation

This example configuration thereby assumes that all JSPs, JSFs, static files... for the live generator and preview generator are similiar, thus the same folder for these two is configured.

generatorconfig-std.txt

This file (normally located in the folder <cmsWorks-installdir>/htdocs/cmsworks.generator.configs) configures which document types are allowed to be generated by the Generator. This file is only read once while starting the Generator service. So after updating the configuration the service should be restarted to read the updated file.

No documents are allowed to be generated until there is a declaration in the generatorconfig-std.txt. The declarations generally do not aim at a particular document but at document types.

Here is a full example of a generatorconfig-std.txt:

```
specialurl=/special/topdiashows.json, /en/diashows/index.html?jsp=json-topdiashows.jsp

doctype=article, .html, page-article.jsp
doctype=diashow, .html, page-diashow.jsp
doctype=diashow, .json, json-diashow.jsp

doctype=navigation, .html

blobtype=medium, data, blob.jsp
```

Example of generatorconfig-std.txt allowing the generation of CMS documents contents

The goal of this configuration is to create URLs with correct mime type to ending assignments. An HTML-Page will have the ending .html, An XML-Feed is ending with .xml and so on. Also it's hiding what kind of generating processes are behind the website.

So here is how it's done:

Any line beginning with # or // is a comment and therefor ignored.

```
doctype=article, .html, page-article.jsp
```

This line assigns the page-article.jsp to handle a request for a CMS document of type **article** ending with .html in the URL (i.e.: http://mywebsite/de/acategory/somearticle.html)

For the document type diashow there are two different assignments. If the URL is ending with .html the page-diashow.jsp will be called to generate. But if the URL is endining with .json the json-diashow.jsp is in use.

```
doctype=navigation, .html
```

At this declaration the assignment of a JSP is missing. This declaration only means, that generating with a document of the type **navigation** is basically allowed. Assuming there is a document of type **navigation** storing a structure of the websites navigation links it would be preferable to call the generation of a website navigation the same way for different website page types like articles, diashows, searchresult pages or others. So the JSP producing the navigation will not build a complete website but only the navigation part of the website. This is not a content for public access. This content is only included while executing page JSPs. So not to assign a JSP in the configuration is the way for declaring only includeable content. A programming example for such includes can be found in the programmers guide.

```
blobtype=medium, data, blob.jsp
```

This sepcial declaration assumes there is a document type **medium** with a blob property called data. If creating a URL to a document of type **medium** it targets the stored blob within the document. This blob has its own mime type and special URL-Ending. If a .txt file is uploaded into the document, the URL must be ending with .txt too. If a .jpg file is uploaded, the URL ends with .jpg too otherwise the generation is not called and the request will be responded with a 404 file not found error. Currently a document having only one property of type blob is allowed.

```
specialurl=/special/topdiashows.json, /en/diashows/index.html?jsp=json-topdiashows.jsp
```

This is the task: call the JSP json-topdiashows.jsp with a cms document but do not show that a JSP is used and what its name is. Use a special URL to address this JSP as a unique call. Declare the path of the document, the JSP is called with - this would usually be a homepage document that will never change its location or name. So with this specialurl declaration the use of a JSP is most effectively disguised and the JSP did not have to be assigned to a special document type.

3.3.8 Search engine services

cmsWorks comes with an integrated search engine service to search CMS contents in on-page-searches at live pages as well as to find documents within the editors desktop.

Why dedicated search services

cmsWorks stores all content like text, pictures and linking structures in a database. Anyway, querying databases may cause performance leaks for the rest of the system. Therefore, cmsWorks uses big data technology that stores certain (searchable) content in specialized index files instead of only storing data in the database. These independent, specialized index files and the access on them respond faster than standard SQL databases.

The service Search

This service encapsulate the operations on search indexes and provide access via HTTP requests. A search index itself consists of file(s) in several folders of the file system on the server. Deleting a folder containing an index will erase that index. The services Search is responsible for the editors desktop search results of all edited documents in one index and for online search results in a separate index containing only published results.

The service stores entries identified by an unique ID (the document ID), it stores the document type, the date, content which is searchable and the content that is returned as result.

To create, update and delete these entries, the service is derived from the AbstractHttpServer (a JSP-engine) including the corresponding JSPs for these CRUD-functionalities being the interface that is queried by the SearchCollector services.

Property file and property key paths are:

Service properties file

<cmsWorks-installdir>/run/properties/search.properties

Service property key paths

/app/cmsworks/service/search/lucene3/Search/

Entry	Value	Description
Port	8091	The port is opened on the server for HTTP-Request to the Search service
Htdocs	../htdocs/search/cms	The path in the filesystem relative to the installation directory, where the JSP files of the Search service reside in ("CRUD"). This is a standard value, meaning that the JSP files placed in that directory provide the interface for the SearchCollector services and must not be altered or deleted to provide full functionality.
DefaultRootIndexFolder	../index/cms	A path in the filesystem that is used for the search indexes, its folders and files.
IndexFields	searchId,yes,un_tokenized,true; searchType,yes,un_tokenized,false; searchDate,yes,un_tokenized,false; searchable,no,tokenized,false; searchContent,yes,no,false	The index fields defined in this parameter configure the structure of the search index. This value reflects the common use of cmsWorks document searches (ID, date, searchable content, returned content). The fields of one search entry are: • the document ID stored in searchId • the document type ID stored in searchType • a date for a document stored in searchDate • content from the document that can be found in searchable • content from the document that can be returned as a search result without asking the CMSCore service stored in searchContent These values of this parameter must not be changed.

```
# ##############################################################################
#
# Properties for the core search engine service
#
# ##############################################################################
# hook service   /app/cmsworks/service/search/lucene3,SearchLucene
# service create /app/cmsworks/service/search/lucene3,SearchLucene,Search


/app/cmsworks/service/search/lucene3/Search/StartTimeout=10000
/app/cmsworks/service/search/lucene3/Search/StopTimeout=10000
/app/cmsworks/service/search/lucene3/Search/AccessTimeout=1000
/app/cmsworks/service/search/lucene3/Search/StopInformExtern=1
/app/cmsworks/service/search/lucene3/Search/LogMode=LogLevel=fatal error info

/app/cmsworks/service/search/lucene3/Search/Port=8091
/app/cmsworks/service/search/lucene3/Search/Htdocs=../htdocs/search/cms

/app/cmsworks/service/search/lucene3/Search/DefaultRootIndexFolder=../index/cms
/app/cmsworks/service/search/lucene3/Search/IndexFields=searchId,yes,un_tokenized,true;searchType,yes,un_tokenized,false;sear
chDate,yes,un_tokenized,false;searchable,no,tokenized,false;searchContent,yes,no,false
```

Configuration example of the Search service for internal search

The service SearchCollector

The SearchCollector is a service that updates the search index on any changes of documents in the project.

The service SearchCollectorWebUI is configured to update all document contents to the search index indexintern so the editors desktop can find every document within the search features of the editors desktop. The service SearchCollectorOnline is configured to update the search index indexonline only when documents are published. Also the index does not contain all contents but only the relevant field contents of relevant document types.

The parameters for the service configuration of the SearchCollector services are found in the properties-file <cmsWorks-installdir>/run/properties/search.properties. In the following table, the "Entry" values paths are /app/cmsworks/service/search/collect/SearchCollectorWebUI/ or /app/cmsworks/service/search/collect/SearchCollectorOnline/

Entry	Value	Description
RemoteHost	127.0.0.1	Addressing the search service this is the host name or IP of the server where the search service is started
RemotePort	8091	Addressing the search service this is the port of the service
CMSService	CMSCore	This is the name of the CMSService where document contents are fetched to be sent to the search service
EventProviderService	CMSCore	This is the name of a service implementing the interface CMSEventProviderable sending Events on changed documents.
PublishedOnly	false	If true this service will only send updates to the search service if documents are published or deleted. Otherwise any document change event will trigger an update in the search index
ResourcePropertyConfig	all	Defines the content that will be send to the search index. There are some rules to this value described below this table.
ResourceAttributeConfig	noname; noreferences; notypes	Without a declaration a documents name, the outgoing references to other documents and special type properties of the document are send to the search index. This can be rejected with the following values: • noname will not send the documents name • noreferences will not send the documents references to other documents • notypes will no send special type properties of a document The optional values are separated by ";".
NonSearchablePaths	/config;/testpages	Documents that are stored under one of these paths will not be sent to the search index. The paths are are separated by ";".
LuceneSearchIndexName	indexintern	This is the index name of the search index for the Search service. It also is the name in the file system where the Search service will store the search index. This name has to be used when searching for documents in that search index.

ResourcePropertyConfig

This service parameter contains the description of the content that will be sent to the search service.

The value all is used mostly in the configuration of SearchCollectorWebUI to send all available content into the search index. Therefor the search features of the editors desktop will work properly and find any stored content in your project.

But if a search index is to be built for an online web page the content to be found should be restricted. Therefor each property of each document type to be indexed is to be declared like this:

```
4:date(optional),category(searchableonly),keywords(searchableonly),headline(searchableonly),text(searchableonly);
8:date(optional),category(searchableonly),keywords(searchableonly),description(searchableonly optional)
```

This declaration defines

Value	Description
4:	document type with id 4 (this is the article document type)
date(optional)	The property date of the document type article may be filled or not. If not the documents last changed date property will be used instead. The date property is the only sorting criterion for search results in this case for articles to be found. If the date field is not declared always be the documents last changed property will be used.
category(searchableonly)	The category of the article is a document reference to a category document (another document type) The value category[documentId]a will be written into the search index. Because this value is not declared optional, the article will not be sent to the search index if this property is not filled. The property content is not stored in the search index and not returned in the search result.
keywords(searchableonly)	The keywords property of the article must be filled or else the document is not sent to the search index. The property content is not stored in the search index and not returned in the search result.
headline(searchableonly)	The headline property of the article must be filled or else the document is not sent to the search index. The property content is not stored in the search index and not returned in the search result.
text(searchableonly)	The text property is not optional and will not be returned in the search result. The property content is not stored in the search index and not returned in the search result.

When declaring more document types use ; as separator. When listing the properties of a document type use , as separator.

The content control values (names after fields in braces) are:

Value	Description
searchable	The field value is sent to the search index. The field value can be found and is returned in the search result.
searchableonly	The field value is sent to the search index. The field value can be found but is not returned in the search result. Furthermore the field value is not stored in the search index so less disk space is required.
optional	The documents fields will be sent to the search index if the field is filled or not.
required	The field content will not be sent to the search index but if the field is not filled the document will not be sent to the search index.
	Using no value means that the content is not searchable but is returned in the search result.

Here is a full example of the searchcollector configuration:

```
# ##############################################################################
#
# Properties for a service listening to cms events to collect data from cms documents
# and send them to a search service
#
# The SearchCollectorWebUI is the most common searchcollector service used in a cms
# project. This service collects nearly all data to provide a search within the
# editors web access.
#
# ##############################################################################
# hook service   /app/cmsworks/service/search/collect,SearchCollector
# service create /app/cmsworks/service/search/collect,SearchCollector,SearchCollectorWebUI

/app/cmsworks/service/search/collect/SearchCollectorWebUI/StartTimeout=10000
/app/cmsworks/service/search/collect/SearchCollectorWebUI/StopTimeout=10000
/app/cmsworks/service/search/collect/SearchCollectorWebUI/AccessTimeout=1000
/app/cmsworks/service/search/collect/SearchCollectorWebUI/StopInformExtern=1
```

```
/app/cmsworks/service/search/collect/SearchCollectorWebUI/LogLevel=fatal error info

/app/cmsworks/service/search/collect/SearchCollectorWebUI/RemoteHost=127.0.0.1
/app/cmsworks/service/search/collect/SearchCollectorWebUI/RemotePort=8091
/app/cmsworks/service/search/collect/SearchCollectorWebUI/CMSService=CMSCore
/app/cmsworks/service/search/collect/SearchCollectorWebUI/EventProviderService=CMSCore
/app/cmsworks/service/search/collect/SearchCollectorWebUI/PublishedOnly=false
/app/cmsworks/service/search/collect/SearchCollectorWebUI/ResourcePropertyConfig=all
/app/cmsworks/service/search/collect/SearchCollectorWebUI/ResourceAttributeConfig=
/app/cmsworks/service/search/collect/SearchCollectorWebUI/NonSearchablePaths=
/app/cmsworks/service/search/collect/SearchCollectorWebUI/LuceneSearchIndexName=indexintern

# ###############################################################################
#
# Properties for a service listening to cms events to collect data from cms documents
# and send them to a search service
#
# ###############################################################################
# service create /app/cmsworks/service/search/collect,SearchCollector,SearchCollectorOnline

/app/cmsworks/service/search/collect/SearchCollectorOnline/StartTimeout=10000
/app/cmsworks/service/search/collect/SearchCollectorOnline/StopTimeout=10000
/app/cmsworks/service/search/collect/SearchCollectorOnline/AccessTimeout=1000
/app/cmsworks/service/search/collect/SearchCollectorOnline/StopInformExtern=1
/app/cmsworks/service/search/collect/SearchCollectorOnline/LogLevel=fatal error info

/app/cmsworks/service/search/collect/SearchCollectorOnline/RemoteHost=127.0.0.1
/app/cmsworks/service/search/collect/SearchCollectorOnline/RemotePort=8091
/app/cmsworks/service/search/collect/SearchCollectorOnline/CMSService=CMSCore
/app/cmsworks/service/search/collect/SearchCollectorOnline/EventProviderService=CMSCore
/app/cmsworks/service/search/collect/SearchCollectorOnline/PublishedOnly=true
/app/cmsworks/service/search/collect/SearchCollectorOnline/ResourcePropertyConfig=4:date(optional),category(searchableonly),k
eywords(searchableonly),headline(searchableonly),text(searchableonly);8:date(optional),category(searchableonly),keywords(sear
chableonly),description(searchableonly optional)
/app/cmsworks/service/search/collect/SearchCollectorOnline/ResourceAttributeConfig=
/app/cmsworks/service/search/collect/SearchCollectorOnline/NonSearchablePaths=/config;/testpages
/app/cmsworks/service/search/collect/SearchCollectorOnline/LuceneSearchIndexName=indexonline
```

Configuration example of the service SearchCollectorWebUI

Usecases

The internal search is used to provide a search window in the editors desktop. There can be found all documents ordered by the date of the latest change. Also it is required to find documents referencing a selected document answering the question "Who references me?".

The online search may only find content documents like articles or diashows, if a user searches on the web page. But also it can be used to show the newest articles of a category.

A special online keyword search index may only make keywords searchable so an automatic related articles search can be build.

Re-indexing the search index

The search services of cmsWorks depend on an internal index that is updated when changes are made. Anyway, in case the index got invalid or even lost (i.e. after an installation without the index files, because of deleted or corrupted index files etc.) the index can be rebuilt from the existing data.

Therefore, cmsWorks provides two commands via the telnet server: searchcollector and directsearchcollector. The searchcollector command is the more flexible command and can handle remote search servers. The directsearchcollector is the by far faster collector and should be used to build an index from scratch but it only works if the search services are local services of this cmsWorks instance.

For example, the directsearchcollector is used here to fulfill a complete reindexing of all search indexes. The simple command

```
directsearchcollector reindex
```

will do the reindexing. In case you need a more fine grained behavior, please refer to the help pages of the commands (by typing "help directsearchcollector" or "help searchcollector" into the telnet server shell).

3.3.9 ScriptServer service

The ScriptServer service is a webservice that contains ContentScripts that will run in the editors desktop. A ContentScript is a programmed wizard where labels, text and input elements appear in the editors desktop and after filling them and a click on OK the script receives the input data and performs some document changes or

anything else. It contains a transaction context so if any document changes fail all previous actions can be rolled back.

Property file and property key paths are:

Service properties file

<cmsWorks-installdir>/run/properties/scriptserver.properties

Service property key paths

/app/cmsworks/service/contentscript/ScriptServer/

Entry	Value	Description
Port	8093	The ContentScript JSPs are accessible over this port of the IP or Hostname of the server the cmsWorks instance is running on.
Htdocs	../htdocs/scriptserver/	This is the path in the filesystem where the ContentScript JSP files are located.
CMSServiceName	CMSCore	This is the name of the service that provides access to CMS documents
ThreadPoolSize	20	This is the maximum count of ContentScripts that may be running simultaneously.
RemoteHosts	preview=localhost:8080; webui=localhost:8082; search=localhost:8091	While executing the ContentScript it may need content from other services or produce links to other services contents. These servers host and port can be listed here. With this configuration the same JSP may call different hosts in different cmsWorks instances like dev/test/live instances.

Here is a full example of the ScriptServer configuration:

```
# ##############################################################################
#
# Properties for the script server running ContentScripts
#
# ##############################################################################
# hook service   /app/cmsworks/service/contentscript,ScriptServer
# service create /app/cmsworks/service/contentscript,ScriptServer,ScriptServer

/app/cmsworks/service/contentscript/ScriptServer/StartTimeout=10000
/app/cmsworks/service/contentscript/ScriptServer/StopTimeout=10000
/app/cmsworks/service/contentscript/ScriptServer/AccessTimeout=1000
/app/cmsworks/service/contentscript/ScriptServer/StopInformExtern=1
/app/cmsworks/service/contentscript/ScriptServer/LogMode=LogLevel=fatal error info

/app/cmsworks/service/contentscript/ScriptServer/Port=8093
/app/cmsworks/service/contentscript/ScriptServer/Htdocs=../htdocs/scriptserver/

/app/cmsworks/service/contentscript/ScriptServer/ThreadPoolSize=20
/app/cmsworks/service/contentscript/ScriptServer/CMSServiceName=CMSCore
/app/cmsworks/service/contentscript/ScriptServer/RemoteHosts=preview=localhost:8080;webui=localhost:8082;search=localhost:809
1
```

Configuration example of the ScriptServer service for executing content scripts in the editors desktop

3.4 Using a web server in front of cmsWorks

Using a web server in front of cmsWorks has many advantages - as example you may use the full feature set of any modern web server with the generational functionalities of a full featured content management system in the backend and prosper of a fast delivery speed.

In principle the web server (here an example Apache configuration) accepts client requests and sends them (via mod_rewrite) to cmsWorks. Assuming the site's name is "www.mywebsite.com" and the cmsWorks live generator is running on port 8081 on the same machine as the Apache server, a pretty stripped down example configuration may look like this using the proxying of Apache:

```
<VirtualHost *:80>

  ServerName www.mywebsite.com
  DocumentRoot /var/www/html/mywebsite.com

  ProxyRequests off
  ProxyPassReverse / http://localhost:8081/

  <Location />
    ProxyPass http://localhost:8081/ retry=2
  </Location>

  ErrorLog /var/www/html/mywebsite.com/logs/error.log
  CustomLog /var/www/html/mywebsite.com/logs/access.log combined

</VirtualHost>
```

Of course any other web server (such as nginx etc.) may do the same work.

4 Operating systems and security

cmsWorks is completely written in Java and has no dependencies to the underlying operating system. It only needs an actual Java Virtual Machine (JVM) and a database to run. Given these prerequisites it runs on a couple of operating systems.

Operating system restrictions for development and servers

As long as you meet the criterias (JVM and database have to be available): None.

The development and deployment of finished programs and modules is independent of either the client or the server you are using: Simply compile / release the files and administrate the server.

Security issues using cmsWorks

As long as you take care of your operating system, there are no known security issues using cmsWorks. In fact, cmsWorks does not rely on the underlying operating system in terms that it does not read / use operating system variables, services or anything else being OS dependent or known to be relevant to security matters.

Anyway, cmsWorks has to reveal itself to the outer world to be able to create and deliver content. That means that a couple of ports are open which need special treatment to harden any cmsWorks installation. See cmsWorks and security for more information about that.

4.1 cmsWorks on Linux OS

Nowadays most web sites in the world are delivered using Linux or similar operating systems.

Using Linux as hosting OS for cmsWorks

Though cmsWorks may run on a variety of different types of operating systems, Linux is a little bit predestined to serve it. This is because

- Support of JVMs for Linux servers (OpenJDK)
- Database systems generally are freely available on Linux (here: MariaDB / MySQL for the community edition of cmsWorks)
- Webservers are freely available on Linux
- Linux networking is pretty advanced, extensive monitoring is available
- Linux servers are available from nearly every hoster
- Additional software (i.e. intrusion detection etc.) is easily available

The deployment of finished programs and modules is independent of the client or server you are using: Simply compile / release the files and administrate the server.

Despite the operating system, be it Linux or some other operating systen, some security issues have to come into consideration using cmsWorks, see cmsWorks and security.

4.2 cmsWorks on Microsoft Windows

Microsoft has the Microsoft Windows Server product line and cmsWorks is able to run on these.

Using Microsoft Windows as hosting OS for cmsWorks

cmsWorks runs on a variety of different types of operating systems, Microsoft Windows (Server) is one of them. The reasons are

- Administrator used to Microsoft Windows should be able to run cmsWorks without having to learn new operating systems
- Supported JVMs for Microsoft (OpenJDK)
- The underlying Database of the cmsWorks community edition (MariaDB / MySQL) is freely available on Windows
- cmsWorks works with any webserver, especially with the Microsoft IIS-server, too

In fact, the most *development* of cmsWorks-sites is done using Microsoft Windows as the programmers OS. The deployment of finished programs and modules is nothing other than it is on other operating systems: Simply compile / release the files and administrate the server.

Despite the operating system, be it Microsoft Windows or some other operating systen, some security issues have to come into consideration using cmsWorks, see cmsWorks and security.

4.3 cmsWorks and security

The cmsWorks server comes in its delivery status with several open ports. These ports should be protected to prevent misusage.

Security - harden the cmsWorks-server

The following table shows the ports cmsWorks reveals to the world. The severity of security levels to take in account for these ports is categorized into "**Very high**", "**High**", "**Middle**" and "**Low**".

Port	Description	Importance of protection
8050	Telnet-server	**Very high**: Accessing this port via telnet let you execute system commands within the server.
8080	Preview of cmsWorks	**Middle**: (normally) Non-published content can be accessed through this port. If you are creating internal views or services using the preview service, this port must not be reachable from the outside world either.
8081	Live view of cmsWorks	**Low**: This port reveals only published content.
8082	cmsWorks Desktop (WebUI)	**High**: Via this web interface content can be created, altered or deleted after a login mask was passed.

Attention: Access to the cmsWorks server instance on a productive system always should be secured through a firewall.

In best case, only a webserver like Apache, nginx or IIS should be accessible via port 80, if possible not running on the same server as cmsWorks. The webserver then only points to the preview / live views or the WebUI via mod_proxy or similar methods. This way, additional mechanisms for security can be added using the possibilities of standard webservers (i.e. .htaccess with password protection).

5 Managing users, groups, documents...

cmsWorks comes with a unique kind of user interface called the "cmsWorks editors desktop". The administration of users, document types, roles and privileges, categories, navigation and so forth is mostly handled in the cmsWorks editors desktop itself.

Users with administrator privileges or user administrator privileges get access to admin tools directly in the cmsWorks desktop (WebUI), in most cases the daily administrative tasks may be done there. You may find the user administration and privileges management using user groups in the users guide of cmsWorks.

Document types

All content in cmsWorks is stored in documents having a unique document type. A brief introduction on how to create document types may be found here.

Categories in cmsWorks

Categories support categorizing the content in a category tree. Categories are special documents that are managed within the cmsWorks editors desktop. More about creating and using categories may be found here.

Navigation in cmsWorks

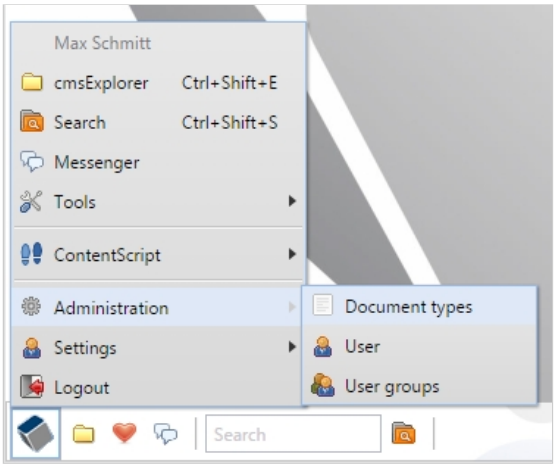
A navigation document structure supports a transparent and flexible way to create navigations for websites. More about managing navigation structures may be found here.

5.1 Managing documents

In cmsWorks content is stored in documents of several types. A document type defines the purpose of documents of this type and also the number and types of input fields that are needed. A user with administration privileges can change them while the system is running.

A document type has a view name, a database name and an ID. Each field of the document type has a view name, a database field name and a field type.

The administration of document types is integrated in the cmsWorks editors desktop.



Main menu -> Administration

All details of handling document types and properties are described in the users guides section Document types.

Property types

The property types depend on the configuration of the database. The following property types are currently in use:

Property type	Description
INTEGER	A number (integer) ranging between -2147483648 and 2147483647
DATE	Date stores the date and time with an accuracy down to the second
STRING	Up to 254 characters as a string, stored in format UTF-8
TEXT	Floating text property that is treated in the desktop using the richtext editor or displayed as grid
BLOB	Storing an uploadable binary large object (blob) such as images, PDFs, etc.
LINK	Representing a list of one or more references to other documents, useful for related resources etc.

5.2 Configuration of categories

In cmsWorks categories are organized in a tree structure. This tree is a special folder beside the folder structure containing the content documents. A category has one parent and child categories. Content documents containing a category property choose the category from a special tree selection based on the category documents from the category folder created for this purpose.

To create a category tree a document type "category" has to be created first. It must contain at least one string property "viewname". Create the special folder i.e. /categories/ and within this folder a document named "category" as the root of the category tree.

Then create a subfolder /categories/cars/ and create a document also named "category" filling the field "viewname" with "The most beautiful cars".

Configure the Additional CMS properties declaring

- the folder of the category tree - /categories
- document type id of the catergory document - [doctypeid]
- the name of the document in the tree - category
- the name of the viewname field - viewname

Open a content document containing a category input field and open the category selection. It contains a first entry called "The most beautiful cars".

With more subfolders and their document of document type "category" the tree is filled.

More about processing categories is described in the Programmers guide.

5.3 Navigation

Creating a separate navigation structure allows the full control over the navigation but it requires several documents to be created and to be linked to each other.

Create a navigation tree

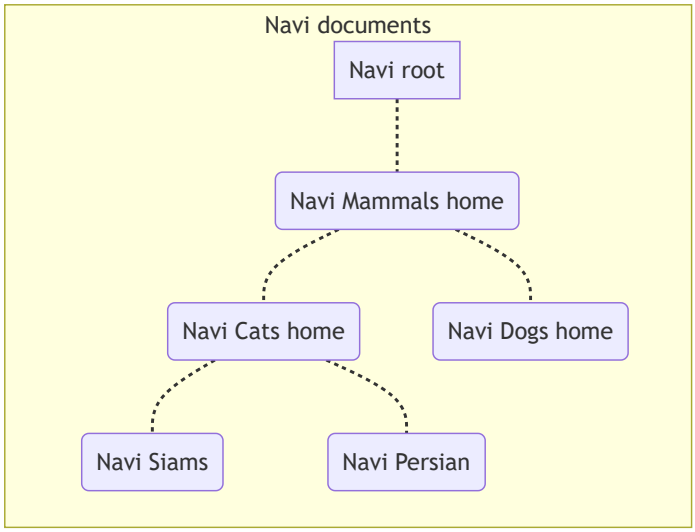
At first create a document type navi. Fill the document type with the following fields:

- title - string
- linkinternal - link
- linkexternal - string
- linktype - integer
- linkparameter - string
- children - link

With this document type create one root navi document and a list of navi entries. Now add the navi entries into the root navi document into the linklist children in a respecive order. In our example the root navi document has no link and no title. This is just the container to list the children in order.

Fill the navi entries with Titles (shown as Navi text) and add link targets either as internal links to content pages into linkinternal or external URLs into linkexternal.

Now the document structure of Navi document could look like this:

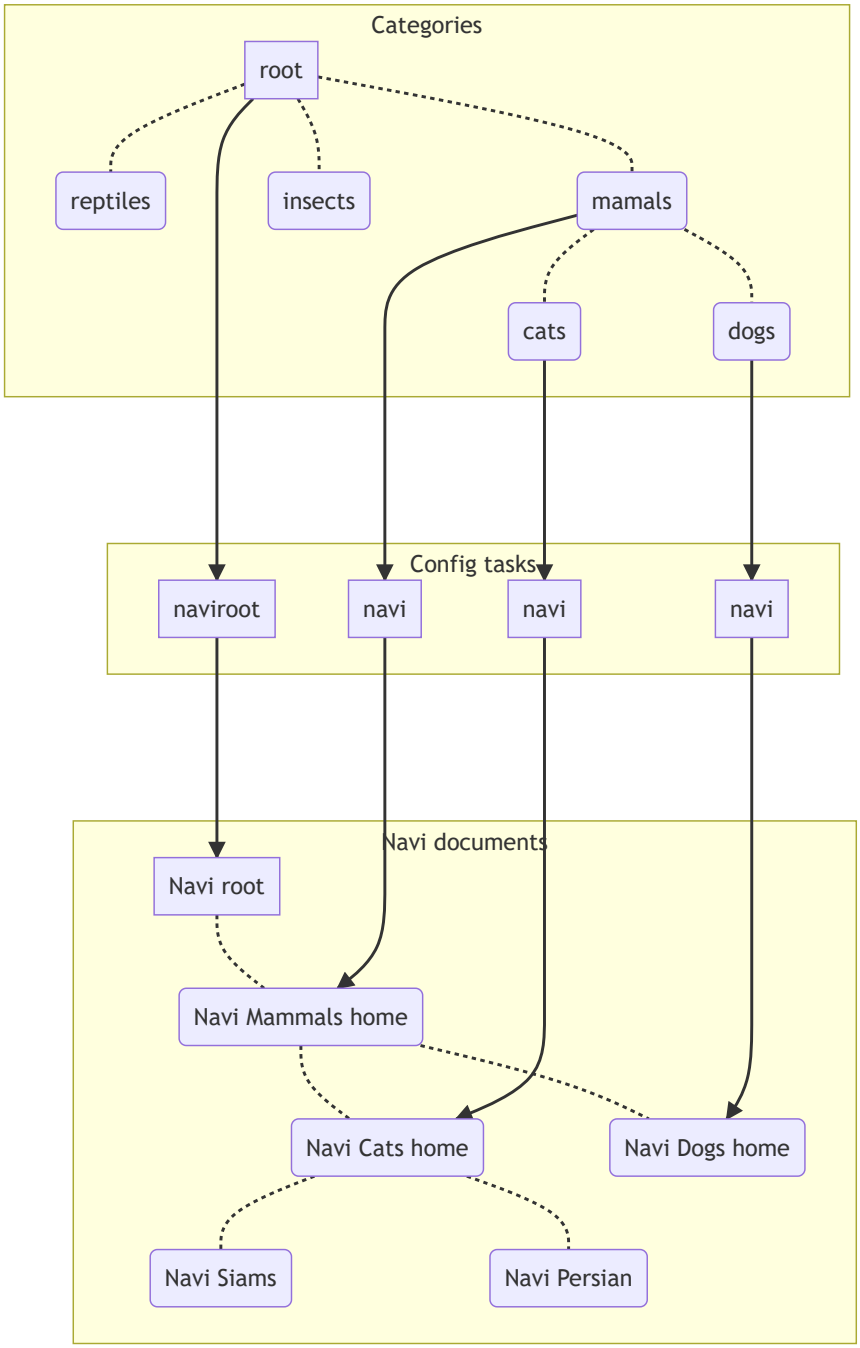


The navigation structure of cmsWorks

In combination with the category tree

Now combine the Navigation structure to the category tree. Assuming there is a category "Cats". This category can be referencing a configuration task of type "Navi" which is referencing the Navigation document "Navi Cats home". With this declaration any content page (incuding the cats home itself) selecting the category "Cats" will automatically select the navigation entry "Navi Cats home". Additionally there would be needed a second configuration task type "Navi root" referencing the root document of the Navigation. This configuration task would be placed at the category root so from every category the configuration task "Navi root" can be found from the inheritance mechanism of the categories.

A linking structure in combination with the category tree looks like this:



The navigation structure and category integration in cmsWorks

6 Tailoring the cmsWorks desktop UI

The cmsWorks desktop is a powerful tool for editors to get their work done. But every client's site has different requirements to a content management system, so the interface to enter content into the system has to be flexible enough to reflect these requirements in the best possible way.

Therefore the interface for editors in cmsWorks can be tailored individually to the needs of these requirements.

Understanding UI customization - files and structures

The cmsWorks editors desktop allows you to be customized in various ways. This chapter explains the necessary files and their structure here.

Basic customizations

Basic customization includes:

- Assigning a variety of input types to the fields of the document types.
- Configuring the features available for richtext input fields.
- Creating option lists for content types and add hints on the effects of these options.
- Declaring standards for media document types and search options.
- Assigning icons to document types.
- Assigning links to external pages aka ExtWin appering as window within the editors desktop.
- Assigning ContentScripts to different contexts or user privileges to support the content creation.

These functional customizations are using JavaScript within the desktop and are described here.

Additional visual customizations

This chapter is mainly about the customization of the richtext content while editing.

Additionally one may brand the desktop by inserting a branding logo at the desktop background. The configuration is described in this chapter using CSS.

6.1 Custom UI-files and file structures

Every cmsWorks server comes with a (web) service called WebUI. This service provides you with the cmsWorks editors desktop. That desktop can be customized to fullfill your needs respecting functionality and look-and-feel using JavaScript, CSS and Images.

Therefore every WebUI-folder which is located in the filesystem at <cmsWorks-installdir>/htdocs/webui has a subfolder cmsdesk.custom with another subfolder named after the customer parameter - called the customer folder resulting in:

<cmsWorks-installdir>/htdocs/webui/cmsdesk.custom/<myCustomerName>/

Withing this folder resides:

q-custom.js

The q-custom.js file provides control over functionalities of the desktop. Especially it controls the different properties of the input window of documents, configures the richtext editor, provides content scripts and external windows in the desktop environment. This is explained in the chapter found here.

q-custom.css

Fill CSS declarations here to customize the look of the richtext elements while editing content in the cmsWorks editors desktop. Either apply a style more similar to the Website to be produced or maybe apply a more generic style since well structured content enables to not only create a website but other products too.

Also optionally place a product logo at any position on the editors desktop on top of a chosen background picture for a nice branding. Examples are listed in the CSS chapter.

doctypeicons

This folder contains the icons to be assigned to document types. So in any document list in the cmsWorks editors desktop or on the window title bar this icon appears giving a hint on the document type. (The currently provided icons are 16x16px PNG files)

backgrounds

Place any selection of background pictures in any folder depth here to let the cmsWorks editors desktop user choose from. Declare the first impression or even a default background color in the q-custom.js

6.2 Configuring the cmsWorks desktop in JavaScript

The cmsWorks editors desktop needs to be configured adjusting input fields for documenttypes, declaring which document types result in web pages to be previewable, to configure all features of richtext fields, to spread ContentScripts to support content managing and more.

The location of the configuration is

<cmsWorks-installdir>/htdocs/webui/cmsdesk.custom/<myCustomerName>/q-custom.js

The target is to create a JSON structure with configuration information but also not to be restricted to simple JSON. So this file also enables to process Javascript for dynamic configuration if needed.

The basic structure of the q-custom.js - file looks like this:

```
var customer_modification = {
  "documenttypeicon":{}, // Document type icons
  "desktopBackground": {}, // Background color and/or default background image
  "generatorPrefixes": [], // URL prefixes to be ignored when searching documents from URLs
  "no-preview-doctype": [], // Document types having no preview
  "tooltip-field": {}, // Field content for a tooltip of a document
  "documentpropertiespicture":[], // Listing picture properties for picture search
  "searchpictures": {}, // Configuring picture search
  "richtext": {}, // Richtext
  "contentscript": [], // Listing ContentScripts and assigning context and privileges
  "extwin": [], // External pages as native cmsWorks desktop window
  "document":{} // Configuring each field of each document type if needed
}
```

The configuration sections are described below.

documenttypeicon - Document type icons

Document type icons are used beside the window title of documents or in document lists. You may configure these icons for each document type of your CMS document type model. We assume we have the document types "medium", "article", "category", configtask", example:

```
"documenttypeicon": {
  "medium" : "/doctypeicons/icn-resTypeMedium.png", /* for the document type with database name "medium" */
  "article" : "/doctypeicons/icn-resTypePage.png",
  "category" : "/doctypeicons/icn-resTypeCategory.png",
  "configtask" : "/doctypeicons/icn-resTypeConfigtask.png"
}
```

The location of the directory "doctypeicons" is right next to the q-custom.js file.

This entry is optional.

desktopBackground - Desktop background color / image

Just for the visual effect a default start image or a default start background color can be configured. If a user enters the cmsWorks desktop for the first time these entries are used to make a first impression. The user can choose another background image from the Menu / Settings / Background or not but anyway his choise will be stored into the user settings and is used for any further cmsWorks desktop session.

```
"desktopBackground": {
  "backgroundColor": "#ccc",
  "backgroundPicture": "default.jpg",
  "showOptionNoPicture": true
},
```

The path of the backgroundPicture is relative to <cmsWorks-installdir>/htdocs/webui/cmsdesk.custom/<myCustomerName>/backgrounds/. Feel free to provide a wide choise of

background pictures with any folder depth.

This entry is optional.

generatorPrefixes - URL prefixes to be ignored when searching documents from URLs

A generator can have configured a prefix used in every URL this generator creates or interprets. To support finding a Document when putting a URL into the search input field, a configured generator URL prefix should be ignored.

The property generatorPrefixes lists all prefixes that are used by Generators of this project.

```
"generatorPrefixes": ["dyn", "syn"]
```

Just use the prefix name, not using the /.

This entry is optional.

no-preview-doctype - Document types having no preview

This is a list of all document types you must not call a preview for, so to say "nonvisual" document types. In the example above this would be the document types category and configtask, example:

```
"no-preview-doctype": [
  "category",
  "configtask"
]
```

If a preview is called for a document that has no preview page an error page from the preview will be called with a rather technical error message only developers will understand properly. If this document type is declared as a no-preview-doctype a warning message "No preview available" appears in the cmsWorks editors desktop.

This entry is optional.

tooltip-field - Field content for a tooltip of a document

```
"tooltip-field" : {
  "article":"headline",
  "person":"name",
  "container":"type"
},
```

The property tooltip-field assigns for a document type the field containing a key information about the document. That fields content is presented in a tooltip about a document within the explorer document list or any other document list like link lists or search results. So without opening the document an additional glimps of the content is presented.

This entry is optional.

documentpropertiespicture - Listing picture properties for picture search

This special configuration for a document type containing a blob input field. When uploading a picture the input field itself read the dimension and uses this entry to fill the assigned input fields for with and height if this is configured here.

Also this configuration supports the window "Picture search" enabling the search for dimensions.

```
documentpropertiespicture: [
  {type: "medium", blob: "data", width: "width", height: "height"}
],
```

One entry of the configuration contains an object with the properties

- type - the document type containing a blob field
- blob - the field name of the blob field of this document type
- width - the field name of the width field of this document type
- height - the field of the height field of this document type

This field is mandatory but may be empty.

searchpictures - Configuring picture search

In this section the behavor of the Serarch pictures window is configured. Currently there is only one entry that is used: dimensionsoffer.

Filling this property a dropdown appears containing default dimensions for pictures to be searched.

```
searchpictures: {
  dimensionsoffer: [
    [-1,-1, "empty"],
    [1600,-1, "1600"],
    [1200,-1, "1200"]
  ]
},
```

One entry is filled with width, height, label. If a width value is -1 it means that the width will not be checked. If a height value is -1 it means that the height of the found picture is not checked. If both values are -1 it means any dimension is accepted.

This entry is optional.

richtext - Richtext

the property richtext combines the default configuration of link types and the default TinyMCE configuration.

```
"richtext": [
  "linkTypes": [
    [0,"Standard"],
    [1,"New window"]
  ],
  "tinymce": {...},
  "components": true,
  "componenttypes" : {
    "m":getLang("Centered"),
    "i":getLang("Inline"),
    "l":getLang("Left"),
    "r":getLang("Right")
  }
]
```

The linkTypes section list fills a selectbox in the link dialog for links within the richtext. The link in the richtext is not the standard HTML link, it's filled with more properties like internal document id to another cmsWorks document or an external URL and the link type id. So the generator can decide how to create an URL to the targeted document depending on the link type. For example if the type 1 is set for "New window" the generator link creation process could add a property target="_blank".

The property tinymce is filled with a complex set of properties configuring TinyMCE and is described in a separate chapter .

The property components enables the placement of text components in the text (only if the plugin and tooltip entries are filled).

The property componenttypes fills a set of options for the component positioning. The key references the translated label.

This entry is optional. There is a fallback basic configuration if this entry is missing.

contentscript - Configuring ContentScripts

ContentScripts are self-programmed workflows which can be executed within the the cmsWorks desktop. A ContentScript

- has a corresponding JSP in the ContentScript server
- may be restricted by user names, user group names or privileges like 'admin' or 'manageUsersGroups'
- works in a context "resource" (works only on documents optionally restricted to special document types), "folder" (works only on folders) or "global"
- can have an initial width / height definition

Example configurations for the (yet imaginary) ContentScripts scriptcreatearticle.jsp, scripthandlehttps.jsp, scriptchangefolder.jsp and scriptaddarticleheadline.jsp (the properties title, name and context are mandatory):

```
"contentscript": [
  {
    "title": "Create an article", /* this is the visible title of the ContentScript in the editors desktop */
    "name": "createarticle", /* createarticle as in "scriptcreatearticle.jsp" */
    "context": "global"
  },
  {
    "title": "Handle some https stuff",
    "name": "handlehttps",
    "visibilityRestriction": {
      "userGroups": [],
      "users":      [],
      "privileges": ["admin"] // only for users belonging to a group with the Administration privilege selected.
    },
    "context": "global",
    "appearance": { "width": 750, "height": 500 } /* set an initial width and height for the ContentScript window*/
  },
  {
    "title": "Change the folder",
    "name": "changefolder",
    "context": "folder"
  },
  {
    "title: "Add an article headline",
    "name: "addarticleheadline",
    "context: "resource",
    "contextDetail": { /* this ContentScript is only available for documents of type article or category*/
      "resourceType": [ "article", "category" ]
    }
  }
]
```

The script createarticle works global and not depending on a folder or document.
The script handlehttps may only be called by users of user groups having the privilege Administration checked and has a defined initial with and height when showing up the first time for a user.
The script changefolder only works on folders.
The script addarticleheadline only works on resources (documents) of type **article** or **category**.

With the configuration property visibilityRestriction the visibility/accessibility of the single ContentScript may be restricted to some users only. These users may be selected by login names, user groups or privileges of assigned user groups. The combination of all restriction entries are combined by OR. A user may see and use the ContentScript if their login name is listed or at least one of their assigned user groups and so on.

This entry is optional. If not filled the Menu entry "ContentScript" is simply not available.

extwin - External pages as native cmsWorks desktop window

Sometimes it's useful to integrate external web pages into the cmsWorks desktop as a window that is treated like all other windows, too. For that entries can be placed in the extwin list as the example shows:

```
"extwin": [
  {
    "title": "Home of the cmsWorks developers",
    "url: "https://www.itechworks.de/portal/index.html",
    "visibilityRestriction": {
      "userGroups": ["Editors"],
      "users":      [],
      "privileges": ["admin", "manageUsersGroups"]
    },
    "appearance": { "width": 980, "height": 700 }
  }
]
```

Setting an extwin like this, the line "Home of the cmsWorks developers" appears in the cmsWorks desktop main menu. You may create as many extwin entries as you need.

The property visibilityRestriction works similar to the one in the ContentScript section.

This entry is optional.

document - Property input fields of document types

In cmsWorks there are in general six types of properties that can be stored in a document type. They may be combined in any way needed and are described in the chapter about Document type management.

Having a new document type created you can configure the input fields using the q-custom.js - 'document' section. How to configure the different input fields is described in the next chapter.

This entry is optional.

6.2.1 Property input field customization

Documents contain properties of types String, Text, Blob, Integer, Date or Link (see the chapter about document type management).

There are standard input fields for those property types available that may be configured or replaced by other input fields storing their results in one of the basic property types.

Each property of each document type can be configured separately. If not there is always a predefined input field for the property active.

Different input field appearances can be managed by selecting the correct "xtype" from the table below.

xtype	Type	Description	Additional actions / options
itwblob	BLOB	Blob input field (upload)	-
itwdatetime	DATE	Date input field	-
itwtext	STRING	String input field consisting of one line	May use the horizontal width as an appearance property regulating the length of the input field
itwdocumenttable	LINK	Linklist of other cmsWorks documents	-
itwcategory	LINK	Specialized input field for cmsWorks categories	-
itwrichtext	TEXT	Richtext input field	May override the initial configuration of tinyMCE by using 'tinymce: { ... }'
itwdatagrid	TEXT	Grid input field	The underlying property type has to be of type "text"
itwinteger	INTEGER	Number (integer) input field	Has a validator for numbers as default validator
itwcheckbox	INTEGER	Checkbox input field	In case the checkbox should be checked by default please set defaultValue = 1
itwselectbox	INTEGER	Selectbox input field	The list of options for the selectbox is declared in an options array property of type [number,value], you may default to an option by using defaultValue = <numberOfOption>
itwcombo	STRING	Selectbox with string input	A special selectbox which additionally allows to enter a string instead of the selections provided

To get a visual of the input type visit in the Document input fields from the cmsWorks user guide.

Example configuration: Creating a simple article

Assuming we have a document type named "article", this document may have the fields

- articledate (DATE)
- articletype (INTEGER, because it shall be a selectbox)
- title (STRING)
- abstract (TEXT)
- text (TEXT)
- related (LINK, because it is a list of other related articles)
- exportable (INTEGER, because it shall be a checkbox)
- source (STRING, because it shall be a selectbox with additional string input)
- parameters (TEXT, this shall be a grid to show the configuration of it)

So the overall document section of the configuration looks like this:

```
"document": {
  "article": {
    "articledate_prop": { ... }, // Actually not necessary because dates cannot be tailored
    "articletype_prop": { ... },
    "title_prop": { ... },
    "abstract_prop": { ... },
    "text_prop": { ... },
    "related_prop": { ... },
    "exportable_prop": { ... },
    "source_prop": { ... },
    "parameters_prop": { ... }
  },
  "insertnextdocumenttypehere": {
    ...
  }
}
```

Date input field

The articledate property as date property cannot be tailored and therefore is skipped here.

Selectbox

The articletype property being a selectbox with the values "Background", "Preview", "Standard" exemplary looks like

```
"articletype_prop": {

  // The xtype changes the input field to a SelectBox.
  "xtype": "itwselectbox",

  /* The options list defines the assignment of the
     Integer value to a Label (in this case a type name).
     An optional third value should contain a String describing the effects
     of the selection and is shown at the info button beside the dropdown input element.
  */
  "options": [
    [0,"Background","Use as a background article"],
    [1,"Preview"],
    [2,"Standard"]
  ],

  // In a newly created document the option 'Standard' (2) will be preselected.
  "defaultValue": 2

},
```

String (single line)

The title property is a one-line input field, appearance-width and defaultValue apply.

```
"title_prop": {

  // Setting the width to 250 pixel
  "appearance": { "width": 250 },

  // Not really reasonable to set a default text here, for demonstration purposes only
  "defaultValue": "Some default text"

},
```

Richtext

Configuring the text properties of the article document is optional. If nothing is configured, the richtext input field will be used for text fields, the linktypes and richtext configuration is used from the richtext property of the customer_modification which are described in detail here.

Setting a configuration will optionally override the linktypes or replace the properties of the richtext configuration.

```
"abstract_prop": {
  "xtype": "itwrichtext",
  "tinymce": {
    "config": {
      "toolbar": "undo redo | styleselect removeformat | bold italic underline | formatselect | bullist numlist | link unlink"
    },
  }
},

"text_prop": {
   "xtype": "itwrichtext",
   "tinymce": {
     "components": true,
     "componenttypes" : {
       "m":"centered",
       "i":"inline",
       "l":"left",
       "r":"right"
    }
  }
},
```

In this example for the abstract text only the toolbar is adjusted. Code view and table creation is removed. For the text property components (special cmsWorks text components) are beeing enabled and configured.

Linklist

The "related" property is a link list and cannot be tailored any further.

```
"related_prop": {
  "xtype": "itwdocumenttable"
},
```

Checkbox

The "exportable" property is a checkbox.

```
"exportable_prop": {

  // Define input field CheckBox.
  "xtype": "itwcheckbox",

  // Set an (optional) alternative Label to explain the effect using the checkbox.
  "label": "This article may be exported for further use.",

  // Setting the default to "1" automatically checks the box in the beginning, skip this to keep it unselected
  "defaultValue": 1

},
```

Combobox

The source property is a selectbox with the possibility to alternatively enter text, too. This field type is only usable on string input fields.

```
"source_prop": {

  // Define the type as combo
  "xtype": "itwcombo",

  // The user may enter text
  "editable": true,

  // Preselections are stored in data
  "options": [
    ["Please select / enter a value"],
    ["Associated Press (AP)"],
    ["Agence France Press (AFP)"],
    ["Thomson Reuters Corporation"],
    ["IPS"]
  ],

  // Defaulting to this text
  "defaultValue": "Please select / enter a value",

}
```

Grid

The grid property type cannot be tailored any further.

```
"parameter_prop": {
  "xtype": "itwdatagrid"
}
```

6.2.2 Customizing richtext input

The richtext input field is the most flexible input field and thus has a own way to be customized.

As editor for handling richtext cmsWorks integrats the 3rd party product TinyMCE. TinyMCE is highly configurable thus in features, behavior and design. Also integrated cmsWorks specific plugins to enable linking to other cmsWorks documents or setting text components using other cmsWorks resources. All configuration properties of the following list belong to the current release of TinyMCE that is included in the current cmsWorks release.

```
var customer_modification = {
  ...
  "richtext": {
    "tinymce": {
      "block_formats": "Paragraph=p;Headline 2=h2;Headline 3=h3;", // the block formats dropdown of the toolbar
      "style_formats": [ // the style formats dropdown / menu of the toolbar
        {"title": "Text", "items": [ // this is a submenu
          {"title": "Interview name", "inline": "span", "classes:" "ivname"},
          {"title": "Cite", "inline": "span", "classes": "quote"},
          {"title": "Label", "inline": "span", "classes": "label"},
          {"title": "Live", "inline": "span", "classes": "live"}
        ]},
        {"title": "Links", "items": [
          {"title": "External link", "selector": "a", "classes": "linkExt"},
          {"title": "PDFDownload", "selector": "a", "classes": "linkPDF"}
        ]},
        {"title": "Tables", items: [
          {"title": "Standard Table", "selector": "table", classes": "taStd"},
          {"title": "Table 100%", "selector": "table", "classes": "wide"},
          {"title": "Alternating colors light/dark", "selector": "table", "classes": "ac"},
          {"title": "Table Headline", "selector": "tr", "classes": "taHead"}
        ]}
      ],
      "valid_elements": // valid elements of the HTML code
        "a[!href|class|title]," +
        "#p[class]," +
        "br," +
        "-h2/h1," +
        "-h3/h4/h5," +
        "#h6," +
        "-span[!class]," +
        "-table[class|style|width]," +
        "-tr[class|style]," +
        "#td/th[class|colspan|rowspan|style|align]," +
        "-ol[class]," +
        "-ul[class]," +
        "-li," +
        "-b/strong," +
        "-i/em," +
        "-u/strike",
      "plugins": [
        "searchreplace advlist lists link anchor table code paste itwcomponent"
      ],
      "toolbar":component uncomponent | table | styleselect removeformat | bold italic underline | " +
        "formatselect | bullist numlist | link unlink code | undo redo searchreplace"
    }
  },
  ...
}
```

The properties and their functionalities depend on the respective functionalities of the used TinyMCE plugin version.

TinyMCE produces richtext encoded as HTMLm that means you one is working on tags and properties and text in an hierarchical structure. Some tags are defined as block formats (p, h2, div), some tags are used as inline formatter (i, b, span) and some tags have kinds of special abilities (for instance table/tr/td for table structures, ul/li for unordered list entries or ol/li for ordered list entries). Attributes like "class" are used to customize the layout of a tag and may be used on any available tag.

The used configuration properties above are a summarized example of previous cmsWorks projects. Let's explain the properties in detail:

block_formats

```
"block_formats": "Paragraph=p;Headline 2=h2;Headline 3=h3;", // the block formats dropdown of the toolbar
```

Using these block formats the Blocks-dropdown in the TinyMCE toolbar will then show the following three entries:

- Paragraph - using the p-tag
- Headline 2 - using the h2-tag
- Headline 3 - using the h3-tag

Thereby the default block tag in TinyMCE is the "p-tag".

Selecting another block tag by using the dropdown changes the current block to the selected block type. You do not have to select the block completely to change it, just placing the cursor into the block wil suffice. Selecting the same block type again will fallback to the default p-tag.

style_formats

```
"style_formats": [ // the style formats dropdown / menu of the toolbar
  {"title": "Text", "items": [ // this is a submenu
    {"title": "Interview name", "inline": "span", "classes:" "ivname"},
    {"title": "Cite", "inline": "span", "classes": "quote"},
    {"title": "Label", "inline": "span", "classes": "label"},
    {"title": "Live", "inline": "span", "classes": "live"}
  ]},
  {"title": "Links", "items": [
    {"title": "External link", "selector": "a", "classes": "linkExt"},
    {"title": "PDFDownload", "selector": "a", "classes": "linkPDF"}
  ]},
  {"title": "Tables", items: [
    {"title": "Standard Table", "selector": "table", classes": "taStd"},
    {"title": "Table 100%", "selector": "table", "classes": "wide"},
    {"title": "Alternating colors light/dark", "selector": "table", "classes": "ac"},
    {"title": "Table Headline", "selector": "tr", "classes": "taHead"}
  ]}
],
```

In the style_formats section menu classes are assigned to tags according to the styling purpose of the project.

```
{"title": "Interview name", "inline": "span", "classes": "ivname"},
```

The "title" is shown in the formats dropdown. "inline" defines the tag we are working on. "classes" are the classes that are set for the tag. There may be added also multiple classes like for instance "ivname special"

```
{"title": "External link", "selector": "a", "classes": "linkExt"},
```

There is a difference in using "selector" or "inline".

If "selector" is used, the cursor can be placed anywhere in the text of the tag, selecting the style via the formats combobox will set the style only to the declared tag.

If "inline" is used, a new tag will be inserted (like ...) or removed (if already existing).

For table tags or links it is not advisable to use the "inline" method because the format combobox should only set or remove styles, rather it must not remove links or destroy table structures.

valid_elements

```
"valid_elements": // valid elements of the HTML code
  "a[!href|class|title]," +
  "#p[class]," +
  "br," +
  "-h2/h1," +
  "-h3/h4/h5," +
  "#h6," +
  "-span[!class]," +
  "-table[class|style]," +
  "-tr[class|style]," +
  "#td/th[class|colspan|rowspan|style|align]," +
  "-ol[class]," +
  "-ul[class]," +
  "-li," +
  "-b/strong," +
  "-i/em," +
  "-u/strike",
```

This is the list of elements allowed in the richtext editor. A tag inserted that is not in this list will be removed. The full explanation of the rules can be found on the TinyMCE documentation pages, anyway, here are some examples:

Code	Description
br	The tag br is allowed
-h2/h1	The - means that empty tags are removed. h2/h1 means that h2 is allowed and h1 will be turned into h2. This will be the case if you insert a copied text from Word or another Webpage
-span[!class]	The - means that empty tags are removed. span tags without a class property will be removed.
-table[class|style]	This shows how multiple properties are allowed
#p[class],	The # means that an empty Tag will be filled with an

The above listing of allowed tags does not contain a div-Tag since this is not really part of a text. It also has no image tag defined because cmsWorks uses text components to place images, embeds or other objects of that kind. That is because text components normally would have more content fields like (for an image) a title, origin, dimensions or subformats. Keeping the amount of tags enabled for richtext low supports an easier distribution of the text in various content formats (HTML, XML, Plain-Text, pdf, ...).

Text components

Just as a tiny digression, a text component is a special tag containing a reference to another document storing its ID combined with a placement (normally one of Left|Right|Centered|Inline). The text component can be added by adding "component uncomponent" to the toolbar and "itwcomponent" as plugin. The configuration of the placement is done in the richtext section of the customer_modification.

plugins

```
"plugins": [
  "searchreplace advlist lists link anchor table code paste itwcomponent"
],
```

This is the list of Plugins currently in use at cmsWorks. Any additionally desired plugins may be placed in here. Only the itwcomponent is a self created plugin handling text component placements.

toolbar

```
"toolbar": "component uncomponent | table | styleselect removeformat | bold italic underline | " +
  "formatselect | bullist numlist | link unlink | undo redo searchreplace"
```

The definition of the elements shown in the TinyMCE toolbar.

More properties

Using the documentation of TinyMCE there are much more properties that can be used to adjust the richtext input field. Just put them into the q-custom.js, reload the cmsWorks editors desktop and test their functionality.

6.3 Customizing richtext and branding

Within the q-custom.css style adjustments are available for richtext elements. Also a branding logo could be definded appearing on top of the background on the cmsWorks editors desktop.

The q-custom.css is fully qualified as:

<cmsWorks-installdir>/htdocs/webui/cmsdesk.custom/<myCustomerName>/q-custom.css

Customizing the richtext in the richtext editor

Customizing the richtext elements targets size and coloring of headlines, defining line-heights and distances between elements or special layout adjustments for links or lists.

```
/* Richtext styles examples */
.richtext p {
  margin: 5px 0; /* setting a distance between paragraphs*/
}
.richtext p a { /* setting the links of the richtext to black and underlined */
  color: inherit;
  text-decoration: underline;
}
.richtext h2 { /* Adjusting the Headline2 definitions */
  margin: 4px 0;
  font-size: 1.25rem;
  color: #ed7529;
}
.richtext h3 { /* Adjusting the Headline3 definitions */
  margin: 4px 0;
  font-size: 1em;
  color: #ed7529;
}
.richtext ul { /* Setting styles for unordered lists */
  margin: 0 0 0 20px;
  list-style-type: square;
  padding: 0;
}
.richtext ol { /* Setting styles for ordered lists */
  margin: 0 0 0 20px;
  list-style-type: decimal;
  padding: 0;
}
```

Styling a table in the richtext editor is possible, too:

```
.richtext table.taSty tr.taStyHead td { font-weight: bold; font-style: italic; font-size: 1.2em; }
.richtext table.taSty { background-color: #ec722a; }
```

Remember to define the in here used styles ("taSty", "taStyHead", ...) in your q-custom.js richtext configuration section.

Event if there would be a whish to style the richtext of a special property from a special document individually, there could be

```
.richtext.dtcontentpage.ptmetadescription p { /* setting the paragraph style only for the special field */
  font-family: "Courier"
  background: #ccc;
}
```

This would only target the field metadescription of the document type contentpage.

Usage of classes: .richtext.dt<documenttypename>.pt<propertytypename>

Adding a branding logo

Add the following css into the q-custom.css. And save the logo file beside the q-custom.css

```
.deskbg .logo {
  position: absolute;
  top: 10px; /* positioning */
  right: 10px;
  left: initial;
  bottom: initial;
  width: 400px; /* dimension */
  height: 509px;
  background: url(logo.gif) no-repeat center center; /* place the logo right beside the q-custom.css file*/
  opacity: 0.4;
}
```

7 Performance and memory optimization

In general, cmsWorks was developed having performance and memory optimization in mind. But every web site has own kinds of structure and content, so the web sites needs and configuration are different for any new project.

Furthermore, a cmsWorks server is meant to run thousands of hours without restarting it. This means that memory usage should be monitored to keep the system well performing. Having this in mind, you should optimize the memory allocation within cmsWorks and use the performance optimizations the JVM provides.

This chapter only discusses optimizations that can be done by administrative tasks and configuration. Optimizing performance while programming cmsWorks is dealt in the Programmers Guide of cmsWorks.

This chapter is divided into the optimization done by using the JVM, a general description of caching mechanisms in cmsWorks and how to optimize these caches, the benefits of using a web server in front of the CMS, the logging system of cmsWorks for monitoring purposes and additional helping tools.

7.1 Optimization using JVM options

The Java Virtual Machine is undergoing permanent optimiziations. These can be used to enhance performance in speed and memory allocation of cmsWorks.

Despite the programming skills and quality of algorithms that surely are capable to enhance the overall performance of a system, the JVM provides additional mechanisms to enhance the throughput of cmsWorks.

The "-server" flag of the JVM

In general it is a good idea to activate the server JVM by including the flag "-server" into the batch file that starts a long living java application like cmsWorks.

The -server flag increases the starting time of a java application (the starting, for a server, should not happen too often), but also maximizes peak operation speed. Additionally, on the compiler level, the -server flag has more optimization methods while compiling the Java classes ("advanced adaptive compiler") resulting in faster running code in the long term.

In cmsWorks the -server flag is configured on by default in the starting batch file.

Optimizing JVM memory allocation for cmsWorks

The JVM can be parameterized with the memory flag "-Xmx" at starting time which is the maximum heap size (). In fact, cmsWorks can start up and run without touching the "standard" memory settings. cmsWorks itself comes with a very small footprint at startup (normally less than 20 MB) but uses configurable caches to speed up the generation of content.

So in a running environment, when bigger caches in cmsWorks are configured, it is necessary to grant more memory to the JVM. This way the system does not run out of memory. Depending on the estimated size of the memory used by the cmsWorks caches, the JVM maximum heap size has to be increased.

And, as a rule of thumb, the JVM should at very least have 1 GB spare memory in case the caches are filled completely. This is because cmsWorks needs memory for internal procedures and the generation of content, too.

Assuming the cmsWorks caches will need 2GB at most, then the "-Xmx" - flag should be set to at least 3 gigabyte ("-Xmx3072M"), better to 3.5 gigabyte ("-Xmx=3584m"). You can check the real memory usage with the "memory" - command in the telnet server.

Be aware that bigger memory settings will result in longer garbage collections of the JVM so the JVM has to stop-the-world (STW) more often, if not otherwise configured. The bigger the heap memory of a JVM is, the longer these STW-intervals will last. In the stop-the-world time intervals cmsWorks will not be responsive and generate content. So taking the server hardware in account, its memory, processor cores and processor speed, it is a good advice to not exaggerate the value for the maximum heap.

In cmsWorks the "-Xmx" - flag is configured in the startup shell "cmsworks.sh" (Linux / *ix) or "cmsworks.bat" (Windows).

Mandatory JVM options

In any case the JVM option

```
--add-exports=java.base/com.sun.crypto.provider=ALL-UNNAMED com.topasworks.Server
```

has to be included into the startup shell "cmsworks.sh" (Linux / *ix) or "cmsworks.bat" (Windows).

7.2 Caching within cmsWorks: Overview

This chapter gives an overview of the caching methods and how to configure them to enhance the overall performance and memory usage of the system.

In cmsWorks, multiple caching methods on multiple different calling layers are implemented to speed up the generation-of-content process. Caching hereby means that information is read from the database and stored in the application server as plain old java objects (POJOs).

This way, database accesses are minimized optimizing database and cmsWorks throughput performance.

LRU-cache for dynamic caches

Caching is mostly performed by using a LRU (last-recently-used) caching method. So in case the cache quota is exceeded, the least used fragment of the cache is dumped to grant space for a new fragment not already cached. (Re)Calling an already cached fragment will result in re-queuing that fragment to the first place in the cache rather than dumping it.

List of cached data

At first, find here a list of data that is cached in the server rather than always re-read from the database.

Name	Pre-filled	Description
Resource Type	Yes	Holding the document types of the system.
Mime Type	Yes	Complete list of mime types the system can handle.
Resource Name	No	Simple cache to map the name of a resource to its ID for faster access.
Additional Param	Yes	Keeping additional parameters in memory.
Folder	Yes	Holding all (content) folders, that describe the content structure wherein resources (documents) are in.
Published resource	No	(Mostly) Small cache for published resources.
Resources of Folder	No	Keeping a list of IDs of resources (documents) of one folder.
User Group	Yes	Caching the user groups.
Blob	No	Caching media-data (binary large objects).
Resource	No	The cache consisting of already (re)called resources (documents).
User	Yes	Caching the user data.

Pre-filled caches and Resource caches

The caches in cmsWorks roughly divide into two types of caches: "Pre-filled caches" and "Resource caches".

Pre-filled caches

Pre-filled caches are internal caches that need no administration. Rather these caches are filled by the system at starting time and maintained by the system itself. They are normally not related to content stored in the system (with the exception of the Folder cache - this cache always holds all folders of the content structure).

Resource caches

Resource caches are caches that store (meta) information about content or even store the content itself. Following a description what the caches store and how they work in detail.

Name	Description
Resource Name	To enhance the speed of resource name lookups, this cache simply returns the names of resources to a given ID. The "Resource Name" cache should be (at least) twice as high as the "Resource" cache or triple as high as the "Folder" cache settings. This cache has a low memory footprint per entry.
Resources of Folder	When looking up all resources stored in a folder, this cache is used to bypass costly database queries. The cache stores all IDs of resources for a given folder. This cache has a low memory footprint per entry. One entry of one folder consists of int-values for the resource-IDs stored in an int array, so it's size is restricted by the overall count of resources and folders. This cache is not limited in size.
Blob	The blob cache only stores the contents of the media document field. Gathering blobs is expensive in two ways: Getting it from the database is costly and storing it in a cache needs much memory (through its normally bigger size). By dissolving the blob resource field from the other resource fields of the resources in the resource cache, the cache acts more granulated (i.e. fetching a resource to read certain resource fields does not invoke the read of blob field if not needed). This cache has a high memory footprint per entry. Therefore it can be limited in two ways: Quantity-sized and/or memory-sized. Reaching the limit of one of these two sizes and calling a not cached blob means that the oldest cache entry is dumped in favor for the new not-yet cached entry.
Resource	This cache stores the resources (meta data and the belonging fields without the blob fields). The resource cache only stores the last version of a resource, regardless if it is published or unpublished (i.e. the resource is "work in progress"). Blobs are separately stored in the Blob cache. In case a blob is requested from a resource, the resource cache reads it (transparently) from the blob cache and returns it. This cache has a medium memory footprint per entry, depending on the quantity and type of the fields the resources hold.
Published resource	The Published resource cache acts alike the Resource cache with one exception: In case the last version of a resource in the Resource cache is not in state "published", the cache would have to do an expensive database lookup to get the last published resource. Therefore the Published resource cache was created to intermediately store the last published resource and avoid the lookup. This cache has a medium memory footprint per entry but through its nature (called only if a requested resource is not in state published in the Resource cache) it can be kept relatively small. A size of 5000 entries will suffice in most cases.

Configuring cache sizes

Using the telnet server, the parameters for caches can be reset in a life working environment. But be aware that if the server is restarted the values of these parameters will be overwritten by the parameter values stored in the cms.properties - file.

The cms.properties - file contains the initial (after start) values for the different caches. The caching parameters reside in the domain of the CMSCore - service, so their properties-prefix is "/com/itechworks/develop/topas/services/cms/server/CMSCore". The parameters and a description can be found here:

Parameter	Description
ResourceCacheSize	The maximum size of the Resource cache. This is a number starting at "1".
ResourceCacheTypeLimits	Every resource type can get a limit that determines how much resources of one resource type is stored at maximum in the cache. Syntax of the value String is "5:10000;6:2000" which would mean that the quota for resources of resource type "5" is 10000, and the quota of resource type "6" is 2000.
PublishedResourcesCacheSize	Similar to the parameter "ResourceCacheSize".
PublishedResourceCacheTypeLimits	Works like the "ResourceCacheTypeLimits" and should not be used, though this cache is mostly managed through request of the Resource cache.
ResourceNameCacheSize	The maximum size of the Resource name cache. This is a number starting at "1".
BlobCacheSize	The maximum size of the Blob cache. This is a number starting at "1".
BlobCacheMemory	The maximum memory used by the Blob cache. This is a number starting at "1" and represents "megabyte" (so a value of "100" will use at most 100 megabytes for the blob cache)

Monitoring and optimizing caches

The Pre-filled caches can only show off their size, due to their nature that they are always loaded completely into the memory. Using the telnet server, most Resource caches can be monitored by showing the configured size and the fill level of each cache.

Anyway, the Resource cache and the Published resource cache can list the resource types of cached elements containing their quantity separately, a caching ratio and a histogram (a simple time line).

The blob cache shows the limits of quantity and/or memory-size, if configured with either a quantity or a memory-size (or both of them). Additionally, a caching ratio can be reviewed.

Starting to monitor the caches, you can log in to the cmsWorks telnet server and enter the command "cmscaches CMSCore".

Resource cache / Published resource cache

The resource caches are important for the optimization of runtime-behavior of cmsWorks. A resource itself consists of the resource data and the resource fields. Thus, each resource triggers several database queries (one for the resource, the others for the fields of the resource).

This way, a request for a resource that provokes a cache-miss and calls the database is thousands of times more expensive than requesting a resource already cached. So optimizing the resource cache will greatly improve the overall performance of cmsWorks.

Both resource caches ("Resource cache" and "Published resource cache") implemented in cmsWorks are LRU-caches of similar type in administration. They store further meta data about how the caches perform. These meta data are:

- The maximum size and the fill level of the caches
- The distinct number of resources in the cache grouped by their resource type
- The cache ratio including hits/misses grouped by their resource type (including the overall cache ratio)
- A histogram showing the ages of resources in the LRU-cache grouped by time intervals

You can access these meta data via the telnet server. The command to do so is called "cmscaches" and allows various parameters.

Having these data it is possible to optimize the size of the caches and restrictions on singular resource types. It is possible to

- Enhance or reduce the maximum quota (total size) of the resource cache
- Limit every resource type with a maximum quota of allowed resources in the cache

Increasing the size of the resource cache helps, of course, but it uses more memory, too. That is not always an option (see "Optimizing JVM memory" and "Selecting the right Garbage Collector"). The cache will consume memory and, at least, expand the time a garbage collection needs to process a stop-the-world collection. This may result in bad responsiveness of the content management system.

Blob cache

The blob cache also is a LRU-cache that stores all media types in cmsWorks. Blobs can consist of any type of stored media, be it pictures, PDFs or word documents. Even raw text can be stored in blobs. Hence it is most probably the cache which needs the most memory per cache item.

And, reading binary large objects (blobs) from a database normally is very costly for both, the application and the database.

Having such an impact on database performance and the overall memory of the cmsWorks server, the blob cache can be configured in three ways:

- Limiting the quantity of cached blobs by assigning a maximum count to the cache
- Limiting the memory-size of cached blobs by assigning a maximum memory size (in megabytes) to the cache
- Assigning both, the quantity limit and the memory-size limit to the cache

In the latter case, the first limit that is reached will limit the cache. If a cache is configured to hold 10,000 blobs (quantity) and its memory-size is limited to 100m (100 megabyte), then the cache is full if either 10,000 blobs are reached or the total size of the blobs exceeds 100 megabyte.

7.3 Using a web server for static content and further configurations

Most web pages do provide more possibilities than the singular purpose to deliver content like it is stored in cmsWorks. Typically web sites consist of content, shop systems, user generated content like forums etc. . Having a Webserver in front of cmsWorks greatly helps to divide and split these tasks according to the services running in your own server environment.

Furthermore, there are certain limitations belonging to the configuration of the cmsWorks server stack. As example, cmsWorks itself does not - out of the box - implement the possibilities of restricted access of parts of the page which may be (simply) done using a standard Webserver by deploying something like a simple .htaccess file or SSL like Let's Encrypt of which you may be familiar and your workflow operates with. Please remember, the main task of cmsWorks is to deliver content - not to restrict it.

Additionally, performance gains may be obtained by dividing the delivery of static content (like often used and statically placed images, CSS or JavaScript files) from dynamically generated files of cmsWorks (like articles, distribution pages and so forth).

cmsWorks and Webservers

cmsWorks itself relies on the application server topasWorks. This means that - by default - you do not have to setup and configure a Webserver to run cmsWorks. That, greatly, helps whilst developing in cmsWorks: cmsWorks is, at first, designed to be a full-stack-application on it's own, meaning that you do not have to install a Webserver on your development environment at all.

Anyway, for a productive environment it is a good idea to put a Webserver "in front of" cmsWorks as mentioned above and to configure it correctly, at least for production aka "heavy" use. This is pretty simple to do.

Deliver static files via a web server rather than cmsWorks

Every website has some standard files used by pretty all pages of it's domain: Be it something like logo (or similar) graphics, a "JavaScript used by (most of) the pages" or CSS files. In case you are using a Webserver it definitely makes sense to deliver these files from the Webserver rather than using the detour over cmsWorks just to reduce the load on your delivering server(s) in a very simple way.

Simple approach for the Apache Web Server in front of cmsWorks

One example using ProxyPass is found here. Another using the website "www.mywebsite.com" as an Apache server instance that accepts requests on IP 10.0.0.1 and your cmsWorks server resides on IP 10.0.0.2 follows. The cmsWorks live generator is serving http configured on port 8081, a minimum Apache config for this would be, by example

```
<VirtualHost *:80>

ServerAdmin info@mywebsite.com
ServerName mywebsite.com
ServerAlias mywebsite.com www.mywebsite.com

# your ongoing configurations here (or below)

RewriteEngine On
ProxyPass / http://10.0.0.2:8081/ retry=0 timeout=10
ProxyPassReverse / http://10.0.0.2:8081/
ProxyErrorOverride On

</VirtualHost>
```

You may ignore the first lines but this way the Apache server on IP 10.0.0.1 takes all requests and passes them to the cmsWorks server on IP 10.0.0.2 to port 8081. SSL or gzip-encoding etc. should be handled before the requests are passed to cmsWorks. And please bear in mind that this is only a very simple example configuration.

8 Integrated tools

cmsWorks comes with a number of helping tools for administrators to setup, maintain, alter and monitor the overall system.

Among these tools are the database tools of cmsWorks which help to setup a new database by creating tables and inserting the first standard values, creating cmsWorks export files of whole databases and importing them as well.

Additionally, cmsWorks has a built in telnet server to execute commands in the cmsWorks - CLI (command line interface) to manage system settings, monitor behaviour and logging.

Then a set of tools cummulated in the so called "ADM tools" which are accessible in the cmsWorks desktop via a browser (you have to be logged in with administrative rights, the ADM tools are found in the start menu under "ExtWin"). This collection of tools help to see into internal (generator) configs, list exceptions raised in programming and export or import documents, trees of documents and document types.

8.1 The ADM-Tools of cmsWorks

The ADM-Tools (Administration) are to be found within the cmsWorks desktop under the Menu "ExtWin". Only if your account you access the cmsWorks desktop with has granted administrative rights the ADM-Tools are available.

The ADM-Tools are a collection of tools to i.e. import / export content, for inspecting the configuration or even changing it.

Document model

The cmsWorks desktop has a section Administration where document types can be created/changed/removed where fields can be created/changed/removed already. So why is a document model configuring section also in the ADM-Tools?

This is more an import/export tool for the document model. Imagine creating a new feature for an existing project developing it in a development environment. This tool then enables you to simply export the current document model and import it when releasing the new feature on a productive installation.

Or if you have a large document model containing a lot of document types this tool enables a quick comparison to another model or another state of the model.

Export / Import documents

While developing a new feature for an existing project and new documents are created this tool helps to simply export these documents from a development instance and import it into a production installation.

The export tool allows to find documents by providing the ID or by simply browsing the folders. When a document is selected the linked documents from this document are examined and shown. So either linked documents are to be chosen for the export also or it's a linked document that exists in the target installation already.

In case such a linked document is not chosen and not existing in the target installation the link will not be set into the document when importing it. Finally put in a name for the file of the collection of document to be exported and execute the export. A download link appears to download the export-XML. This XML contains document references always as combination of folder path, document name and document type. Also all binary content is stored as base64 encoded text.

Using the import tool you first have to select an XML file to be imported. It verifies all document references and document contents against the document model:

- Does the document type exist?
- Do the fields in this document type exist and match?
- Is a document for the path already existing?
- Is the existing document of the same document type?

If all checks have passed successfully a list of documents (to be imported) appears with a button to start the import process.

Messaging and desktop sessions

A tool of the ADM-Tools Messaging allows to send a broadcast message for all currently loggedin users of the cmsWorks desktop. In most cases this is used to inform all users to stop the work and to leave the cmsWorks desktop for a short maintenance shutdown.

The desktop sessions tool lists the sessions of currently loggedin users of the cmsWorks desktop with the option to kill the session so the user is logged out.

Exceptions

The server collects exceptions from all services in the order of occurrence. This tool shows the exceptions in the order of the amount of occurrence. If there is a larger project with several different pages or even generators and many scripts sometimes an error appears at a never thought of possible combination of parameters and repeatedly the same Exception is logged.

So this tool shows you on top if such errors exists and how often they have occurred. After fixing the errors the Exception list could be cleared to see if the fix did the job. Because remember: An empty exception list is always an administrators delight ;-).

Precompile

This is a method to safely check - before their initial execution - if all JSPs / JSFs are valid after an update.

Every service based on an AbstractHttpServer (every Generator, WebUI, ScriptServer, Search) has defined a htdocs folder where JSP / JSF files exist. This tool request every JSP of these services to trigger a compilation of it rather than waiting for it's execution. If a compile fails, the stacktrace of this JSP / JSF file is listed.

Server instance

"Server instance" shows license information and enables you to either shutdown the complete server or restart the whole server instance.

Generators

The menu "Generators" lists all Generator services showing port, all remote ports, htdocs path, generation folder and the JSP assignment configuration from the generatorconfig-std.txt with the option to reload the generatorconfig-std.txt without restarting the service.

Additional CMS properties

Some special properties are used to describe the category implementation (if used), they may be edited here. Additional, all other stored values of the additional CMS Properties are shown and are made editable here.

It shows a list of all AbstractHTTPServer (except Generators) listing its ports and showing configured remote ports (also checking if they are available/active). The SearchCollector configurations for the internal and external big data search engines may be inspected here too.

8.2 The telnet server and useful telnet commands

The content management system cmsWorks is shipped with an internal telnet server. This means that you can connect to the server by simply using a normal "telnet" - command of your operating system. After connection you get a shell having certain cmsWorks-internal commands that help to administrate, change, enhance and even shut down a running cmsWorks instance.

Connecting to the telnet server

The standard telnet server port is "8050", the standard telnet server password is "topas", so if you are trying to get a connection to a cmsWorks telnet server running on the machine you work on, try "telnet localhost 8050" and enter the password.

Enter "help" for help

Starting with the telnet server, the most helpful command, of course, is the "help" command, simply entering this will list all available telnet commands. To gain more information about the single commands type "help <commandname>" to get a description of that specific command. Without any further argument the "help" command lists all commands of the telnet server.

Fast overview: Memory usage and logging

To simply get the memory usage of the server or to obtain a log snippet you may look here or here. Anyway, the corresponding commands are described in detail in the chapter system-commands.

Simple use: the command repeater (".")

In case the used operating system telnet command is not compatible using the arrow-up and arrow-down keys to recall already entered command lines, you could use the dot (".") command.

This command can list all already entered command lines of one telnet session (via ". list"), it can repeat the last entered command line (simply enter "."), repeat one numbered command line (i.e. ". 3" will repeat the third command line) or an interval of command lines (i.e. ". 3 5" will repeat the third up to the fifth command line).

8.2.1 Memory and garbage collection

Using the telnet server you can receive the memory allocation at any time by simply entering the "memory" command.

Also, it is possible to trigger a complete stop-the-world garbage collection using the telnet command "gc". This way you could, for example, at first call "memory", then "gc" and again "memory" to get a good feeling of how much memory a complete GC will free.

Be aware that while the development of different garbage collectors enhanced, it is not any more advisable to trigger garbage collections by hand. This may disturb internal optimization routines of the JVM.

8.2.2 Logging for debugging

The telnet command to control the logging in the content management system is simply called "log". With this command you can add or delete file loggers, switch log levels on and off and even redirect the actual server log into a running shell.

Additionally, the cmsWorks log mechanism may store exceptions, too. These exception can be viewed using the "exception" command in the telnet server shell.

Description of these two commands:

Command	Description
exception	cmsWorks is able to store exceptions in its log messages. These exceptions can be listed in intervals, exported in an exception list file, flushed (emptied) or they can simply be printed on screen using the "exception" command.
log	The "log" command is the central command to maintain logging in cmsWorks. "log status" gives a list of all active loggers (i.e. file loggers, loggers that print to the telnet consoles etc.).

Logging in cmsWorks

Logging in cmsWorks is done using one of the following log levels: "debug", "trace", "info", "warn", "error", "fatal", "inform extern". The latter tries to send an email to pre-defined email receivers, the first ones are for internal logging within cmsWorks.

The log command manages the logging and can be used to log into files ("log add file <filename> <loglevels>") or even log to the actual telnet server session ("log link info" writes the info-level logs to the telnet server). Instead of enumerating one or more log levels (i.e. "log link debug info") you can use the log level word "all" (i.e. like "log link all" instead of "log link debug trace info warn error fatal").

For more information type "help log" into the telnet server.

(Re)Viewing exceptions in the cmsWorks log

Examples for the telnet server exception command are

- "exception count" (returns the number of exceptions stored in the exception log)
- "exception print 3" (prints the third exception in the exception log)
- "exception clear" (clears the exceptions from the log and frees the used memory of these exceptions)
- "exception export <filename>" (exports the exception log into a file on the server).

For more information type "help exception" into the telnet server.

8.2.3 Telnet server system commands

cmsWorks comes with a handful of system commands that can be useful while administering the server, this page lists the most useful of them with a brief description.

Command	Description
.	The dot (".") command shows a list of already entered command lines of one telnet session, it repeats the last / a specific / a specific interval of command(s). The "." command, when entered into the telnet server console, repeats one or more commands stored in the telnet session history list. Every time a command (valid or invalid) is entered, this command is stored in a list called "history". The command "." can repeat the last command entered or a list of previously entered commands. The repeater must not be executed in batch files, because the client environment (history list) is missing within batches, an appropriate error message will be returned in this case.
batch	Lists, (re)loads or executes batch files. Batch files are files with command lines stored on the server, they help to repeat commands. Batches are user-defined command lists, which can be executed using the telnet-server of cmsWorks. They are written using a simple editor and stored in the batch-path of the server's file system with the file extension ".batch". There are several points to take care of while using batches: • The batch-path can be set and reviewed using the "batch" command i.e. to switch to a set of testing batches. Be aware that this could influence the server though other batches, eventually having the same name, will be loaded (automatically). • Batches should be (re)loaded before they can be executed. This is done by issuing the command "batch load xyz", which will load the batch "xyz.batch" from the batch-path. To execute a batch, call "batch exec xyz" afterwards. • A loaded batch will normally stay in memory until it is reloaded or removed with this command, even if the batch-file will change meanwhile unless the autoloading option for batches is set to "on". At server startup, the batch "startup.batch" is loaded and executed automatically. The administrator may add additional commands in there, i.e. to automatically start services, execute other batches etc. . Having a cmsWorks enterprise license enables you to even hook self-written telnet commands, jobs and services into the cmsWorks server.
cmscaches	This command show details about the caches the CMS is using and provides some changing of these caches while in runtime.
echo	This command prints a message onto the console or re-prints each command after it was entered, when activated. Echo is turned off by default. The command mainly is useful on telnet clients that do not support automated echoing or to review the entered commands or simply to print informational messages from batches. Without any parameter the command returns the current echo-state.
email	Command to administer emails, to send emails or to list emails stored in the server (if they were not sent before). The command "email" is used to use the cmsWorks server to send emails to the addresses defined in the topas.properties file or the log ("state "OM_INFORM_EXTERN"). With this command, emails may be send manually or switched off for maintenance, so that people are not notified of changes done in that time.
exception	This command prints all (or a specified interval of) exceptions logged in the cmsWorks-server. Exceptions are logged by using the cmsWorks logging while programming cmsWorks.

	The command shows one or a list of (all) exceptions occurred since exception logging was enabled. Additionally it can print these exceptions containing a complete stack trace. In cmsWorks-logging, the exception-numbers are printed in brackets so they can easily be printed using the "exception print <number>" command. It is possible to save (export) the exception log into a file called "exception.log". In case the cmsWorks server is shut down normally, the stored exceptions are written into the file "exceptions.log" in the <cmsworks-install> directory. It is recommended to clear the exception log from time to time because the exceptions in the exception lists do consume memory. In some cases, it may even be advisable to disable the logging of exceptions i.e. to safe memory in application servers which are well tested, therefore the option "exception off" was implemented.
gc	Manually triggers a vm-wide garbage collection. The use of this command is not recommended on production-level servers though it interfereces with JVM internal optimizations.
help	Lists all commands or prints help messages of certain commands. As a start enter "help" to get a list of available commands or even "help help" to get the help of the "help" command while connected to a cmsWorks telnet server.
host	Returns the host name or the IP of the server cmsWorks is running on.
license	Prints the actual cmsWorks license.
log	The log command is a very powerful command while programming or administering a system because it gives complete control over the log services of a cmsWorks-server at runtime (in the telnet console), sending mails and file logging included. New (so-called) "loggers" can be added at runtime, the level of existing loggers can be changed or removed. The logging-target of newly added loggers can be the current (telnet) console, System.out (i.e. standard output), a user-defined file. The following log levels may be set in cmsWorks logging: • DEBUG • TRACE • INFO • WARN • ERROR • FATAL • INFORM_EXTERN • ALL Each log level can be used in any combination with the others. The INFORM_EXTERN log level sends the log message as email. Another special log level is ALL which is the combination of all log levels. To see all configured loggers simply enter "log status" into the telnet console. To remove a logger simply enter "log remove <loggerid>" where <loggerid> is the id you get from the list of the "log status" command. To bind the logging to your running telnet console, the command "log link all" (or "log link debug", "log link debug info" etc.) may be used. The logging will then be printed into your telnet console session. To stop logging into the telnet console simply type the command "log unlink". To start logging into a file you can use the command "log add file <filename> <loglevel>". As example, logging all "info" and "error" messages into one file requires the command "log add file info-and-error.testlog info error". The position of the file in the file system is printed by using the command "log status".

memory	The memory command prints the amount of free and total memory of the virtual machine of the cmsWorks-server. It has no parameters.
service	The command "service" gives full control over all registered microservices of the cmsWorks server instance (via "service status"). Services can be listed, their status can be retrieved, they can be started, stopped, and their parameters can be listed. The command stops ("service stop <servicename>") or starts ("service start <servicename>") services.
shutdown	This command shuts down the cmsWorks-server *immediately*. Thus, all services will be terminated and the virtual machine is terminated. If the normal shutdown command fails due to a service-stop-error, you can shut down the server via the command "shutdown abort".
time	Prints the current time and date of the server.
uptime	The uptime command prints the hostname, time and date the cmsWorks-server was started, alternatively (via "uptime count") the running time in days, hours, minutes and seconds since the last start.
version	Prints the version of the underlying application server and the server property name.
wait	Waits for the given value in milliseconds (i.e. "wait 1500" will wait 1.5 seconds) before the next command will be executed, useful i.e. for batches.

The parameters of every command and how they work can be retrieved with the help command (i.e. type "help email" to get a full parameter list with descriptions from the email-command)

8.2.4 Content management system specific commands

cmsWorks has specialized telnet commands which work on the CMSCore service for caching and the internal big data search engines for editors and the live page.

You may receive a more detailed help when you log in to the telnet server and type "help <command>", i.e. "help cmscaches".

Command	**Description**
cmscaches	This command show details about the caches the CMS is using and provides some changing of these caches.
directsearchcollector	This executes the update of a searchindex by using the services directly.
searchcollector	This command uses a SearchCollectorService to send updates to the search index via http calls and is part of the cmsWorks Enterprise Edition.

8.3 Database tools of cmsWorks

cmsWorks is coming with an inbuild memory HSQL database which support to just install the software and just run it. For small content amount and for experimenting it is sufficient. But if placing a cmsWorks into a productive context with unlimited content growth using a separate database is recommended. This than support better performance, scalability, stability and a separate backup strategy.

After an external database is available and a database for cmsWorks is create and a user can connect and has rights to create database tables the cmsWorks database tools come into play.

Every command uses information for the database connect:

host/ip
: The host name or IP of the server with the database

port
: The port the database can be reached at

databasename
: The name of the database for cmsWorks

user
: The user name

password
: The users password

In the filesystem of the cmsWorks installation change into folder <cmsWorks-installdir>/run/bin/ .

The files setenv.sh (Linux / *ix) or setvars.bat (Windows) declare the correct assignment of the java installation. The same way it is described in Configuring the start script.

The following database handling scripts are available:

dbexecquery.sh
: Execute a select-query directly to gather data.

dbexecscript.sh
: Executes a sript to be found in the subfolder sqlscripts. Available are initfilldb.script (creates default cmsWorks folder, the standard users and groups) or removeall.script (clear all database tables belonging to cmsWorks)

dbexportimport.sh
: This exports the database content from all tables of the cmstables.txt database description into a zip file or imports formerly exported contents into the database after deleting the existing database contents (so use with care).

dbinitcms.sh
: Create all database tables. Depeding on the connection protocol the scripts sqlscripts/inithsqldbdb.script, sqlscripts/initmysqldb.script, sqlscripts/initoracledb.script or sqlscripts/initpostgresqldb.script will be executed.

For windows the same scripts are available using the corresponding .bat files. All Scripts show a usage and examples when no parameters are given.

To fill an empty fresh created database with cmsWorks tables use (.sh on Linux / *ix or .bat on Windows)

```
dbinitcms.sh <connection protocol> <server/ip> <port> <database name> <user> <password>
```

To initially fill root folder(s), base users, groups and more use

```
dbexecscript.sh sqlscripts/initfilldb.script <connection protocol> <server/ip> <port> <database name> <user> <password>
```

After executing "dbinitcms.[sh|bat]" and then "dbexecscript.[sh|bat]" as described, you will have a complete cmsWorks database filled with base credentials but without having any content.

Attention! In case you use MariaDB or MySQL, make sure that the encoding is completely set to UTF-8. You may force this by adding "?useUnicode=true&characterEncoding=utf-8" to the database name in the configuration or while using the database tools.