hyperCMS

Content & Digital Asset Management Server

Version 10.0

Developer Guide

2022-11-02

Index

1 Introduction

The following chapters deal with the function libraries of the hyper Content & Digital Asset Management Server and thus provide the documentation of the API (Application Programming Interface).
All libraries are located within the hyperCMS installation in the folder "function" and can be integrated and used in the respective scripts or templates. This can be used, for example, to create dynamic pages (applications) using the XML content repository.
If you run your application on a physically separated server, it is important that the function libraries are also available on the publication server. In this case you need to have access to the corresponding files on the publication server as well.

2 hyperCMS XML-Content-Repository

The XML content repository includes all XML Content Container and thus provides all content in native XML. The structure (schema) within an XML content container is dynamically generated based on the template used and has the following appearance:

```
<?xml version="1.0" encoding="UTF-8" ?>
<container>
  <hyperCMS>
    <contentcontainer>0000023.xml</contentcontainer>
    <contentxmlschema>object/page</contentxmlschema>
    <contentorigin>%page%/Publication/testpage.php</contentorigin>
    <contentobjects>%page%/Publication/testpage.php|%page%/ Publication/linkedcopy_of_testpage.php
|</contentobjects>
    <contentuser>demouser</contentuser>
    <contentcreated>2002-12-01 10:02:40</contentcreated>
    <contentdate>2004-11-26 14:32:33</contentdate>
    <contentpublished>2004-11-26 14:39:41</contentpublished>
   <contentstatus>active</contentstatus>
  </hyperCMS>
  <head>
    <pagetitle>test</pagetitle>
    <pageauthor>Mr. Content</pageauthor>
    <pagedescription>just a small demonstration</pagedescription>
    <pagekeywords>demo of XML</pagekeywords>
    <pagecontenttype>text/html; charset=UTF-8</pagecontenttype>
    <pagelanguage>de</pagelanguage>
    <pagerevisit></pagerevisit>
  </head>
  <textcollection>
    <text>
      <text_id>headline</text_id>
      <textuser>demouser</textuser>
      <textcontent>fgfdgfdg</textcontent>
    </text>
    <text>
      <text_id>summary</text_id>
      <textuser>demouser</textuser>
      <textcontent><![CDATA[This is a
      <STRONG><EM>summary</EM></STRONG>]]></textcontent>
    </text>
  </textcollection>
  <mediacollection>
    <media>
      <media_id>logo</media_id>
      <mediauser>otheruser</mediauser>
      <mediafile>Publication/demo_hcms0000033.jpg</mediafile>
      <mediaobject>%page%/Publication/Multimedia/demo.jpg</mediaobject>
      <mediaalttext>demoimage</mediaalttext>
      <mediaalign></mediaalign>
      <mediawidth>200</mediawidth>
      <mediaheight>100</mediaheight>
    </media>
```

```
  </mediacollection>
  <linkcollection>
    <link>
      <link_id>verweis</link_id>
      <linkuser>demouser</linkuser>
      <linkhref>http://localhost/index.php</linkhref>
      <linktarget>_blank</linktarget>
      <linktext>click me</linktext>
    </link>
  </linkcollection>
  <componentcollection>
    <component>
      <component_id>teasers</component_id>
      <componentuser>otheruser</componentuser>
      <componentcond>$customer == "private"</componentcond>
      <componentfiles>%comp%/Publication/teaser_1.php|%comp%/Publication/teaser_2.php|</componentfiles>
    </component>
    <component>
      <component_id>banner</component_id>
      <componentuser>demouser</componentuser>
      <componentcond></componentcond>
      <componentfiles>%comp%/banner.php</componentfiles>
    </component>
  </componentcollection>
  <articlecollection>
    <article>
      <article_id>news</article_id>
      <articletitle>Top News</articletitle>
      <articledatefrom>2002-10-01</articledatefrom>
      <articledateto>2002-11-01</articledateto>
      <articlestatus>active</articlestatus>
      <articleuser>demouser</articleuser>
      <articletextcollection>
        <text>
          <text_id>news:headline</text_id>
          <textuser>demouser</textuser>
          <textcontent>News from Scene</textcontent>
        </text>
      </articletextcollection>
      <articlemediacollection>
      </articlemediacollection>
      <articlelinkcollection>
      </articlelinkcollection>
      <articlecomponentcollection>
      </articlecomponentcollection>
    </article>
    <article>
      <article_id>special</article_id>
      <articletitle>Special Info</articletitle>
      <articledatefrom>2002-01-01</articledatefrom>
      <articledateto>2002-01-01</articledateto>
      <articlestatus>inactive</articlestatus>
      <articleuser>otheruser</articleuser>
      <articletextcollection>
        <text>
          <text_id>special:informations</text_id>
          <textuser>otheruser</textuser>
          <textcontent><![CDATA[<STRONG><FONT color=#cc0033>What is really going on behind the
Scene</FONT></STRONG>... find it out]]></textcontent>
        </text>
      </articletextcollection>
      <articlemediacollection>
      </articlemediacollection>
      <articlelinkcollection>
      </articlelinkcollection>
      <articlecomponentcollection>
      </articlecomponentcollection>
    </article>
  </articlecollection>
</container>
```

After a review of the content container, a structure can be seen, which is composed of the following main elements for content storage:

- hyperCMS specific information
- Meta-information
- Text
- Media (images or other multimedia files)
- Links
- Components
- Articles

The entire content is made up of these basic elements whose information is stored within XML tags.
Articles include the elements text, media and links as well. The entire contents of a page or component can be obtained from the associated content containers.

2.1 hyperCMS specific information

The data collected in this XML node represent primarily relevant information for the management of the container.

```
<hyperCMS>
  <contentcontainer>0000023.xml</contentcontainer>
  <contentxmlschema>object/page</contentxmlschema>
  <contentorigin>%page%/testpage.php</contentorigin>
  <contentobjects>%page%/testpage.php|%page%/linkedcopy_of_testpage.php |</contentobjects>
  <contentuser>demouser</contentuser>
  <contentdate>2002-11-26</contentdate>
  <contentpublished>2002-11-26</contentpublished>
 <contentstatus>active</contentstatus>
</hyperCMS>
```

Description:

contentcontainer	Name of the Content Container (unique for all publications)
contentxmlschema	Schema of the object: page (page) or component (comp)
contentorigin	Object (page or component) that led tot he creation of the Content Containers
contentobjects	All objects which use the Container
contentuser	Object owner (user)
contentdate	Date of the last changes of the Containers
contentpublished	Date of the last publishing of the object based on the Content Container
contentstatus	Status can be "active" if an object using the Cotainer exists. If all objects using the Container have been removed, the status will be set to "deleted". The Container therefore holds the last published information, but it can not be used anymore.

2.2 Meta-Information

The standard meta-information of a HTML page is described in this XML node.

```
<head>
  <pagetitle>test</pagetitle>
  <pageauthor>Mr. Content</pageauthor>
  <pagedescription>just a small demonstration</pagedescription>
  <pagekeywords>demo of XML</pagekeywords>
  <pagecontenttype>text/html; charset=UTF-8</pagecontenttype>
  <pagelanguage>de</pagelanguage>
  <pagerevisit></pagerevisit>
</head>
```

Description:

pagetitle	Page title
pageauthor	Author
pagedescription	Description of the content of a page
pagekeywords	List of keywords oft he page
pagecontenttype	Content-Type (character set) of the page or component
pagelanguage	Language shortcut of the page
pagerevisit	Search engine revisit of the page

2.3 Text

This XML-node stores the text.

```
<text>
  <text_id>headline</text_id>
  <textuser>demouser</textuser>
  <textcontent>fgfdgfdg</textcontent>
</text>
```

Description:

text_id	Text identification
textuser	Text owner (last changes of the Text by a user)
textcontent	text content

2.4 Media

This XML-node describes an included media file.

```
<media>
  <media_id>logo</media_id>
  <mediauser>otheruser</mediauser>
  <mediafile>Publication/demo_hcms0000033.jpg</mediafile>
  <mediaobject>%page%/Publication/Multimedia/demo.jpg</mediaobject>
  <mediaalttext>demoimage</mediaalttext>
  <mediaalign></mediaalign>
  <mediawidth>200</mediawidth>
  <mediaheight>100</mediaheight>
</media>
```

Description:

media_id	Media identification
mediauser	Media owner (last changes of the Media by a user)
mediafile	included multimedia file and declaration of the Publication
mediaobject	Location of the multimedia file
mediaalttext	Alternative text of the multimedia file
mediaalign	Alignment of the multimedia file
medawidth	Displayed width of the multimedia file
mediaheight	Displayed height of the multimedia file

2.5 Links

This XML-node describes the link to a page or file.

```
<link>
  <link_id>link</link_id>
  <linkuser>demouser</linkuser>
  <linkhref>http://localhost/index.php</linkhref>
  <linktarget>_blank</linktarget>
  <linktext>click me</linktext>
</link>
```

Description:

link_id	Link identification
linkuser	Link owner (last changes of the Link by a user)
linkhref	Reference (Link) to a page or file
linktarget	Target of the reference (name of the target frame)
linktext	Text describing the link

2.6 Components

This XML-node describes the reference to Components.

```
<component>
  <component_id>teasers</component_id>
  <componentuser>otheruser</componentuser>
  <componentcond>$customer == "private"</componentcond>
  <componentfiles>%comp%/teaser_1.php|%comp%/teaser_2.php|</componentfiles>
</component>
```

Description:

component_id	Component identification
componentuser	Component owner (last changes of the Component reference by a user)
componentcond	assigned customer Profile to the Component
componentfiles	Reference (Component-link) to a single or multiple Components

2.7 Articles

This XML-node describes the article information.

```
<article>
  <article_id>news</article_id>
  <articletitle>Top News</articletitle>
  <articledatefrom>2002-10-01</articledatefrom>
  <articledateto>2002-11-01</articledateto>
  <articlestatus>active</articlestatus>
  <articleuser>demouser</articleuser>
  <articletextcollection>
  </articletextcollection>
</article>
```

Description:

article_id	Article identification
articletitle	Title of the Article
articeldatefrom	Publishing date of the Article
articeldateto	End date of publishing the Article
articlestatus	Publishing settings of the Article: active = always published/displayed inactive = never published/displayed timeswitched = scheduled publishing/display
articleuser	Article owner (last changes of the Article by a user)
articlecollection	Holds all content of the Article

3 Function libraries

3.1 Including a library

The inclusion of a configuration or library requires that you know the absolute or relative path to the library. By using the function "require" or "require_once" and specifying the path to the library file all functions contained in the library will be available. Once the library is included, all functions can be used in the script.
To use the hyperCMS functions, the file "hypercms_api.inc.php" needs to be included. This file contains all functions required for programming.

```
// absolute path on MS Windows
require_once ("C:/inetpub/wwwroot/hypercms/function/hypercms_api.inc.php");

// relative path on MS Windows or Linux/UNIX
require_once ("function/hypercms_api.inc.php");
```

3.2 Loading the configuration

3.2.1 Content Management Server

To use the main configuration of hyperCMS the appropriate configuration file must be loaded. The main configuration will be loaded when including the hyperCMS API. However you can also load it in your script.
Using the variable $site for the identification of a publication, the publication can be loaded as well. The hyperCMS config file is located in "hypercms/config" and is named "config.inc.php". The publication config files are located in hyperCMS Data directory in the directory "data/config". Its filename holds the name of the publication as well as the ending "inc.php.", example: site.inc.php.

```
// Inlcude the main config file (please set the correct path):
require_once ("C:/inetpub/wwwroot/hypercms/config.inc.php");

// Include publication management config file
// Attention: Please use valid_publicationname to verify the name before including the file
if (valid_publicationname ($site))
{
  require_once ($mgmt_config['abs_path_data']."config/".$site.".conf.php");
}
```

The config files can be opened and read. Each parameter is described therein. Therefore, please take a look at the configuration to learn more about the parameters and their names.

If you want to set a specific language language, the variable $lang need to be set. $lang contains the language code, which is defined in the main configuration file "hypercms/config/config.inc.php".

```
// Set the language for messages in functions, German (de)
$lang = "de";
```

Since you want to use the hyperCMS API you need to include the hyperCMS API loader.

```
// Include the hyperCMS API:
require_once ($mgmt_config['abs_path_cms']."/function/hypercms_api.inc.php");
```

Now you can start using the API functions. For instance loading the content container of an object using various methods:

```
// Loading the page
$pagedata = loadfile ("%page%/MyPublication/home/", "index.php");

// Reading the name of the content container
$contentcontainer  = filepointer ($pagedata, "content");

// Loading the live content container from the content repository
$containerdata = loadcontainer ($contentcontainer, "published", $user);

// Or even more simple by using the direct path to the object
$containerdata = getobjectcontainer ("MyPublication", "%page%/MyPublication/home/",
"index.php", $user);
```

The functions will also load the publication specific configuration in case it is not provided. Since many features require the settings of a publication, it is advisable to include the configuration before you plan any actions.

3.2.2 Publication Server

Note that the configuration of the publication server (publication target) is stored separately in an INI file. If you will need the publication target settings, you must load and parse the INI file. After that you can access the settings as an array.
The INI file of the publication target is located in the external repository in the directory "repository/config". The file name corresponds to the name of the publication with the file extension ".ini".

```
// Load and parse the INI file using PHP
$publ_config = parse_ini_file ("C:/inetpub/wwwroot/repository/config/Mandant_1.ini");

// Access the settings oft he publication target
echo "This is the document root of the publication:".$publ_config[abs_publ_page];
```

3.3 Global variables

Many functions use global variables that are stored in the configuration and are available to functions as global. You should therefore take care that those global variable names of hyperCMS are not changed in your scripts.
The following list shows all global variables of hyperCMS, which must not be changed with in your own scripts:

```
$mgmt_config
$lang
$lang_name
$lang_shortcut
$lang_codepage
$lang_shortcut_default
```

Many global variables of hyperCMS are useful for use in hyperCMS scripts and PHP scripts, these are only available if the corresponding configuration has been loaded, or a hyperCMS script (used only during the publication process) is in use . Since this happened in the preview as well when publishing pages and components, these variables can be used in hyperCMS scripts. For dynamic applications that are executed each time a visitor accesses a page or component, the configuration must be integrated directly in the template, if hyperCMs variables are required.

Content Management Server:

$lang	language shortcut according to config.inc.php
$mgmt_config['url_path_cms'**]**	URL of the hyperCMS root directory according to config.inc.php
$mgmt_config['abs_path_cms'**]**	absolute path to the hyperCMS root directory according to config.inc.php
$mgmt_config['url_path_page'**]**	URL of the document root of the publication in the management system
$mgmt_config['abs_path_page'**]**	absolute of the document root of the publication in the management system
$mgmt_config['url_path_comp'**]**	URL of the component root directory of the publication in the management system
$mgmt_config['abs_path_comp'**]**	absolute path of the component root directory of the publication in the management system

Publication Server:
hyperCMS scripts can access variables at any time. The values are stored in the array $publ_config, but are also optionally available without the array. If the script/application will be executed at each access of a page or component on the publication target, the configuration file must be loaded separately.

$publ_config['url_publ_page'**]**	URL of the document root of the publication target
$publ_config['abs_publ_page'**]**	absolute path of the document root of the publication target
$publ_config['url_publ_comp'**]**	URL of the document root of the publication target
$publ_config['abs_publ_comp'**]**	absolute path of the document root of the publication target

Optional (deprected):

$url_publ_page	URL of the document root of the publication target
$abs_publ_page	absolute path of the document root of the publication target
$url_publ_comp	URL of the document root of the publication target
$abs_publ_comp	absolute path of the document root of the publication target

3.4 Template variables

There is also the possibility to use hyperCMS template variables in templates. These variables are a special feature, since they don not need to be used in hyperCMS script. Rather, they are placeholder for the value of a variable and can be used in any template.
This neutral form of the variables should primarily be used in templates, providing a more technology-neutral usage.
Please pay attention to the lower case of all variables!

%container%	for the name of the content container of an object
%container_id%	for the ID of the content container of an object
%objecthash%	for the hash of an object
%object_id%	for the ID of an object
%template%	for the template file name of an object used
%publication%	for the publication where the object is located
%url_location%	for the absolute path (URL) of the location where the object is located
%abs_location%	for the absolute path in the filesystem of the location where the object is located
%object%	for the name of the object
%date%	describes the current date (format: JJJJ-MM-TT)
%view%	describes the display mode: publish … published cmsview … Editing view in EasyEdit mode preview … Preview formedit … Editing mode in form view formlock … form view with editing locked formmeta … Metadata in form view template ... Template preview

To integrate media files (assets) we can make use of a path variable. When publishing a page or component, the path variable is for instance replaced by the URL (address) of the target of the publication:

%tplmedia%	for the absolute path (URL) of the template media repository
%url_media%	for the absolute path (URL) of the content media repository (Alternative **%media%** can be used)
%abs_media%	for the absolute path in the filesystem of the content media repository

The document root of pages and components of the publication target can be provided as well:

%url_page%	for the absolute path (URL) of the page root folder
%abs_page%	for the absolute path in the filesystem of the page root folder
%url_comp%	for the absolute path (URL) of the component root folder
%abs_comp%	for the absolute path in the filesystem of the component root folder
%url_rep%	for the absolute path (URL) of the external repository root folder
%abs_rep%	for the absolute path in the filesystem of the external repository root folder
%url_ hypercms%	for the absolute path (URL) of the hyperCMS root folder
%abs_hypercms%	for the absolute path in the filesystem of the hyperCMS root folder

Please keep in mind to add a slash “/” to the end of the path variable, if you would like to complement the variable with a continuative path.

Definition of the date format when the format-attribute is used in the textd-Tag:

%a	'am' or 'pm'
%A	'AM' or 'PM'
%d	day of the month, 2 digits with leading zeroes (01 to 31)
%D	day of the week, textual, short, eg "Fri"
%F	month, textual, long; eg "January"
%h	hour, 12-hour format (01 to 12)
%H	hour, 24-hour format (00 to 23)
%g	hour, 12-hour format without leading zeros (1 to 12)
%G	hour, 24-hour format without leading zeros (0 to 23)
%i	minutes (00 to 59)
%j	day of the month without leading zeros (1 to 31)
%l	day of the week, textual, long, eg "Friday"
%L	1 if leap year, otherwise - 0
%m	month (01 to 12)
%n	month without leading zeros (1 to 12)
%M	month, textual, short, eg "Jan"
%s	seconds (00 to 59)
%t	number of days in the month (28 to 31)
%w	day of the week, numeric (0, Sunday to 6, Saturday)
%Y	year, 4 digits, eg 2007
%y	year, 2 digits, eg "07"
%z	day of the year (1 to 366)

If you are using the hyperCMS APIs, it is often advisable to use the place holders %page% and %comp% to access the document root of pages and components. This path variables can be used only on the management side.
It should be noted that the variable is always paired with the publication name to form the root directory, eg:

%page%/%publication%/ …. Pages document root of the current publication

%page%/Publikationsname/ provides the path to root directory of the pages document root.
%comp%/Publikationsname/ provides the path to root directory of the components document root.

3.5 Object operation library

This library contains all functions for the manipulation of objects (pages, components or files). You should only use these functions to access objects that are managed by the system.

3.5.1 createfolder

Syntax:
createfolder ($site, $location, $foldernew, $user)

Description:
Creates a new folder.

Example:
$result = createfolder ("%publication%", "%page%/%publication%/", "company", "brown");

Input-Parameters:

$site	Name of the publication
$location	absolute path (location of the new folder)
$foldernew	Name of the new folder
$user	User name

globale Input-Parameters:
The following global input parameters need to be passed to the function:

$lang	Language setting or language shortcut, e.g. „en“, „de“

Output:

Array	Array $result holds the following information:
$result[result]	True/False (has the folder been created successfully)
$result[add_onload]	JavaScript code for the onLoad event
$result[message]	Message regarding the result of the action or error message
$result[folder]	Name of the folder

3.5.2 deletefolder

Syntax:
deletefolder ($site, $location, $folder, $user)

Description:
Removes an existing folder. The folder is removed only if it contains no more objects. All objects must therefore be removed by using the function deleteobject.

Example:
$result = deletefolder ("%publication%", "%page%/%publication%/", "company", "brown");

Input parameters:

$site	Name of the publication
$location	absolute path (location of the folder)
$folder	Name of the folder
$user	User name

global input parameters:
The following global input parameters need to be passed to the function:

$lang	Language setting or language shortcut, e.g. „en“, „de“

Output:

Array	Array $result holds the following information:
$result[result]	True/False (has the folder been removed successfully)
$result[add_onload]	JavaScript code for the onLoad event
$result[message]	Message regarding the result of the action or error message
$result[folder]	Name of the existing folder is not successful, otherwise empty

3.5.3 renamefolder

Syntax:
renamefolder ($site, $location, $folder, $foldernew, $user)

Description:
Renames an existing folder.

Example:
$result = renamefolder ("%publication%", "%page%/%publication%/", "company", "news", "Miller");

Input parameters:

$site	Name of the publication
$location	absolute path (location of the folder)
$folder	old folder name
$foldernew	new folder name
$user	User name

global input parameters:
The following global input parameters need to be passed to the function:

$lang	Language setting or language shortcut, e.g. „en“, „de“

Output:

Array	Array $result holds the following information:
$result[result]	True/False (Could the folder be renamed successfully)
$result[add_onload]	JavaScript code for the onLoad event
$result[message]	Message regarding the result of the action or error message
$result[folder]	Name of the folder

3.5.4 createobject

Syntax:
createobject ($site, $location, $object, $template, $user)

Description:
Creates a new page or component based on a template. Please note that the location ($location) defines the category of the object (page/component) as well. This implies further that it the value of the parameter $template must provide a valid page or component template.

Example:
$result = createobject ("%publication%", "%page%/%publication%/", "index", "page_main", "Miller");

Input parameters:

$site	Name of the publication
$location	absolute path (location of the object)
$object	Name oft he new object (page or component)
$template	Name oft he page or component template (name of the template or template file name)
$user	User name

global input parameters:
The following global input parameters need to be passed to the function:

$lang	Language setting or language shortcut, e.g. „en“, „de“

Output:

Array	Array $result holds the following information:
$result[result]	True/False (result of the action)
$result[add_onload]	JavaScript code for the onLoad event
$result[message]	Message regarding the result of the action or error message
$result[object]	File name of the page or component
$result[objectname]	Name of the page or component
$result[objecttype]	File-type or file extension of the file

3.5.5 deleteobject

Syntax:
deleteobject ($site, $location, $object, $user)

Description:
Removes an existing page, file or component.

Example:
$result = deleteobject ("%publication%", "%page%/%publication%/", "sales.doc", "Miller");

Input parameters:

$site	Name of the publication
$location	absolute path (location of the Object)
$object	Name of the object
$user	User name

global input parameters:
The following global input parameters need to be passed to the function:

$lang	Language setting or language shortcut, e.g. „en“, „de“

Output:

Array	Array $result holds the following information:
$result[result]	True/False (result of the action)
$result[add_onload]	JavaScript code for the onLoad event
$result[message]	Message regarding the result of the action or error message
$result[publication]	Name of the publication where the object is located
$result[location]	absolute path (location of the Object)
$result[object]	File name of the page, file or component
$result[objectname]	Name of the page, file or component
$result[objecttype]	File-type or file extension of the file

3.5.6 renameobject

Syntax:
renameobject ($site, $location, $object, $objectnew, $user)

Description:
Renames an existing page, file or component.

Example:
$result = renameobject ("%publication%", "%page%/%publication%/", "sales.doc", "best.doc", "Miller");

Input parameters:

$site	Name of the publication
$location	absolute path (location of the Object)
$object	old name of the object
$objectnew	new name of the object (without file extension)
$user	User name

global input parameters:
The following global input parameters need to be passed to the function:

$lang	Language setting or language shortcut, e.g. „en“, „de“

Output:

Array	Array $result holds the following information:
$result[result]	True/False (result of the action)
$result[add_onload]	JavaScript code for the onLoad event
$result[message]	Message regarding the result of the action or error message
$result[publication]	Name of the publication where the object is located
$result[location]	absolute path (location of the Object)
$result[object]	File name of the page, file or component
$result[objectname]	Name of the page, file or component
$result[objecttype]	File-type or file extension of the file

3.5.7 cutobject

Syntax:
cutobject ($site, $location, $object, $user)

Description:
Cut an existing page, file or component.

Example:
$result = cutobject ("%publication%", "%page%/%publication%/", "index.php", "Miller");

Input parameters:

$site	Name of the publication
$location	absolute path (location of the Object)
$object	Name of the object
$user	User name

global input parameters:
The following global input parameters need to be passed to the function:

$lang	Language setting or language shortcut, e.g. „en“, „de“

Output:

Array	Array $result holds the following information:
$result[result]	True/False (result of the action)
$result[add_onload]	JavaScript code for the onLoad event
$result[message]	Message regarding the result of the action or error message
$result[object]	File Name of the page, file or component
$result[objectname]	Name of the page, file or component
$result[objecttype]	File-type or file extension of the file
$result[clipboard]	temporary entry in the clipboard (can be passed as global variable $clipboard to the function pasteobject, so reading the temporary file is not necessary)

3.5.8 copyobject

Syntax:
copyobject ($site, $location, $object, $user)

Description:
Copy an existing page, file or component.

Example:
$result = copyobject ("%publication%", "%page%/%publication%/", "index.php", "Miller");

Input parameters:

$site	Name of the publication
$location	absolute path (location of the Object)
$object	Name of the object
$user	User name

global input parameters:
The following global input parameters need to be passed to the function:

$lang	Language setting or language shortcut, e.g. „en“, „de“

Output:

Array	Array $result holds the following information:
$result[result]	True/False (result of the action)
$result[add_onload]	JavaScript code for the onLoad event
$result[message]	Message regarding the result of the action or error message
$result[object]	File name of the page, file or component
$result[objectname]	Name of the page, file or component
$result[objecttype]	File-type or file extension of the file
$result[clipboard]	temporary entry in the clipboard (can be passed as global variable $clipboard to the function pasteobject, so reading the temporary file is not necessary)

3.5.9 copyconnectedobject

Syntax:
copyconnectedobject ($site, $location, $object, $user)

Description:
Connected copy an existing page, file or component sharing the same content container.

Example:
$result = copyconnectedobject ("%publication%", "%page%/%publication%/", "index.php", "Miller");

Input parameters:

$site	Name of the publication
$location	absolute path (location of the Object)
$object	Name of the object
$user	User name

global input parameters:
The following global input parameters need to be passed to the function:

$lang	Language setting or language shortcut, e.g. „en“, „de“

Output:

Array	Array $result holds the following information:
$result[result]	True/False (result of the action)
$result[add_onload]	JavaScript code for the onLoad event
$result[message]	Message regarding the result of the action or error message
$result[object]	File name of the page, file or component
$result[objectname]	Name of the page, file or component
$result[objecttype]	File-type or file extension of the file
$result[clipboard]	temporary entry in the clipboard (can be passed as global variable $clipboard to the function pasteobject, so reading the temporary file is not necessary)

3.5.10 pasteobject

Syntax:
pasteobject ($site, $location, $user)

Description:
Paste Einfügen an existing page, file or component.

Example:
$result = pasteobject ("%publication%", "%page%/%publication%/", "Miller");

Input parameters:

$site	Name of the publication
$location	absolute path (location of the Object)
$user	User name
$clipboard	temporary entry in the clipboard (can be passed as global variable $clipboard to the function pasteobject, so reading the temporary file is not necessary)

global input parameters:
The following global input parameters need to be passed to the function:

$lang	Language setting or language shortcut, e.g. „en“, „de“

Output:

Array	Array $result holds the following information:
$result[result]	True/False (result of the action)
$result[add_onload]	JavaScript code for the onLoad event
$result[message]	Message regarding the result of the action or error message
$result[publication]	Name of the publication where the object is located
$result[location]	absolute path (location of the Object)
$result[object]	File name of the page, file or component
$result[objectname]	Name of the page, file or component
$result[objecttype]	File-type or file extension of the file

3.5.11 lockobject

Syntax:
lockobject ($site, $location, $object, $user)

Description:
Locking of one or more existing pages or components based on the same content containers for the exclusive use of a user.

Example:
$result = lockobject ("%publication%", "%page%/%publication%/", "index.php","Miller");

Input parameters:

$site	Name of the publication
$location	absolute path (location of the Object)
$object	Name of the object
$user	User name

global input parameters:
The following global input parameters need to be passed to the function:

$lang	Language setting or language shortcut, e.g. „en“, „de“

Output:

Array	Array $result holds the following information:
$result[result]	True/False (result of the action)
$result[add_onload]	JavaScript code for the onLoad event
$result[message]	Message regarding the result of the action or error message
$result[object]	File name of the page, file or component
$result[objectname]	Name of the page, file or component
$result[objecttype]	File-type or file extension of the file

3.5.12 unlockobject

Syntax:
unlockobject ($site, $location, $object, $user)

Description:
Unlocking of one or more existing pages or components based on the same content containers for the exclusive use of a user.

Example:
$result = unlockobject ("%publication%", "%page%/%publication%/", "index.php", "Miller");

Input parameters:

$site	Name of the publication
$location	absolute path (location of the Object)
$object	Name of the object
$user	User name

global input parameters:
The following global input parameters need to be passed to the function:

$lang	Language setting or language shortcut, e.g. „en“, „de“

Output:

Array	Array $result holds the following information:
$result[result]	True/False (result of the action)
$result[add_onload]	JavaScript code for the onLoad event
$result[message]	Message regarding the result of the action or error message
$result[object]	File name of the page, file or component
$result[objectname]	Name of the page, file or component
$result[objecttype]	File-type or file extension of the file

3.5.13 publishobject

Syntax:
publishobject ($site, $location, $object, $user)

Description:
Publishing a page or component. All connected copies of the object and its content container will be published as well. If a workflow is in use and does not permit the publsihing, the object will not be published.

Example:
$result = publishobject ("%publication%", "%page%/%publication%/", "index.php", "Miller");

Input parameters:

$site	Name of the publication
$location	absolute path (location of the Object)
$object	Name of the object
$user	User name

global input parameters:
The following global input parameters need to be passed to the function:

$lang	Language setting or language shortcut, e.g. „en“, „de“

Output:

Array	Array $result holds the following information:
$result[result]	True/False (result of the action)
$result[add_onload]	JavaScript code for the onLoad event
$result[message]	Message regarding the result of the action or error message

3.5.14 unpublishobject

Syntax:
unpublishobject ($site, $location, $object, $user)

Description:
Unpublishing a page or component. Link and task management will be executed automatically. All connected copies of the object and its content containers will be unpublished as well.

Example:
$result = unpublishobject ("%publication%", "%page%/%publication%/", "index.php", "Miller");

Input parameters:

$site	Name of the publication
$location	absolute path (location of the Object)
$object	Name of the object
$user	User name

global input parameters:
The following global input parameters need to be passed to the function:

$lang	Language setting or language shortcut, e.g. „en“, „de“

Output:

Array	Array $result holds the following information:
$result[result]	True/False (result of the action)
$result[add_onload]	JavaScript code for the onLoad event
$result[message]	Message regarding the result of the action or error message

3.5.15 getlinkedobject

Syntax:
getlinkedobject ($site, $location, $object, $cat)

Description:
This function extracts all objects that have a reference to the given object. This may be page or component links. If the object is a page, then all objects which have a page link to the object will be determined. If the object is a component, all objects which have a component link to the object will be determined.

Example:
$result = getlinkedobject ("%publication%", "%page%/%publication%/", "index.php", "page");

Input parameters:

$site	Name of the publication
$location	absolute path (location of the Object)
$object	Name of the object
$cat	optional: Objekt category [page, comp]

Output:

Array	Array $result holds the following information:
$result	False (action was not successful)
$result[publication]	Name of the publication where the object is located
$result[location]	absolute path (location of the Object)
$result[object]	Name of the object
$result[category]	Object category [page, comp]

3.5.16 getconnectedobject

Syntax:
getconnectedobject ($site, $container)

Description:
This function determines all objects that are based on the same content container. The name of the content container of an object can be by extracted by the function "getfilename".

Example:
$result = getconnectedobject ("%publication%", "0000127.xml");

Input parameters:

$site	Name of the publication
$container	Name of the content container

Output:

Array	Array $result holds the following information:
$result	False (action was not successful)
$result[publication]	Name of the publication where the object is located
$result[location]	absolute path (location of the Object)
$result[object]	Name of the object
$result[category]	Object category [page, comp]

3.5.17 getobjectcontainer

Syntax:
getobjectcontainer ($site, $location, $object, $user)

Description:
This function loads the content containers (XML string) of a particular object. The object can be a page, file, component or folder.
The desired content can be extracted from the XML-based container using the function “getcontent” or “selectcontent”.

Example:
$xmldata = getobjectcontainer ("%publication%", "%page%/%publication%/", "index.php", "demouser");

Input parameters:

$site	Name of the publication
$location	absolute path (location of the Object)
$object	Name of the object
$user	User name

Output:

XML-String	Content of the content container
False	An error occured

3.6 Edit content

The programmatic editing of an object contents is shown by the following example. Note the function settext, which is used to manipulate text contents of an object, for more details see the Set API function library.

```
// load object file information
$objectinfo = getobjectinfo ("%publication%", "%page%/%publication%/Home/", "index.php", "demouser", $container_version="");

// load content container (work status)
$contentdata = loadcontainer ($objectinfo['container_id'], "work", "demouser");

// set a new text
$text = array();
$type = array();
$textuser = array();
$text['Title'] = “My new title”;
$type['Title'] = “u”;
$textuser['Title'] = “demouser”;
$text['Description'] = “My new description”;
$type['Description'] = “f”;
$textuser['Description'] = “demouser”;

$containerdata = settext ("%publication%", $contentdata, $objectinfo['container'], $text, $type, "no", $textuser, "demouser");

// save working xml content container file
if (!empty ($containerdata)) $result = savecontainer ($objectinfo['container_id'], "work", $containerdata, "demouser");
```

3.7 File operation library

The following functions for file operations should never be used to load or save objects (pages, components or files).
However you can use them to load and save XML content container, if you intend to develop extensions or applications.

3.7.1 loadfile

Syntax:
loadfile ($abs_path, $filename)

Description:
This function loads the content of a file. The absolute path and the filename itself must be provided as input parameters. The function waits usually up to 3 seconds to load locked files. If the user parameter $user is set, the function can also read locked files of the given user.

Example:
$data = loadfile ("%page%/%publication%/home/", "index.php");

Input parameters:

$abs_path	absolute path of the file, %page% and %comp% can be used as the root elements of the path
$filename	file name

Output:

File content	The function was executed successfully and returns the content of the file
False	An error occured

3.7.2 savefile

Syntax:
savefile ($abs_path, $filename, $filedata)

Description:
This function saved content in files. The absolute path of the file name, and the content that will be written to the file needs to be passed as parameters. If the file is locked, it will not be saved and False will be returned.

Example:
$result = savefile ("%page%/%publication%/home/", "index.php", "text content");

Input parameters:

$abs_path	absolute path of the file, %page% and %comp% can be used as the root elements of the path
$filename	file name
$filedata	Content that will be saved in the file

Output:

True	The function was executed successfully
False	An error occured

3.7.3 loadlockfile

Syntax:
loadlockfile ($user, $abs_path, $filename)

Description:
This function allows to load the content of a file like the function ”loadfile”, but it is also triggers a locking mechanism for the file.
The function should only be used when the data will be saved again using the function “savelockfile”. This ensures that no other write access by other users can take place.
The user, the absolute path and the filename itself must be passed as a parameter to load and lock the file. To save and unlock the file the function “savelockfile” must be used.

Example:
$data = loadlockfile ("Miller", "%page%/%publication%/home/", "index.php");

Input parameters:

$user	User name of the user who locked the file
$abs_path	absolute path of the file, %page% and %comp% can be used as the root elements of the path
$filename	file name

Output:

File content	The function was executed successfully und liefert den Inhalt der Datei
False	An error occured

3.7.4 savelockfile

Syntax:
savelockfile ($user, $abs_path, $filename, $filedata)

Description:
The function “savefile” saved data and unlocks previously opened files using “ loadlockfile”. For this purpose, the user, the absolute path, the file name, and the content that needs to be written to the file must be passed as parameters.

Example:
savelockfile ("Miller", "%page%/%publication%/home/", "index.php", "file content");

Input parameters:

$user	User name of the user who locked the file
$abs_path	absolute path of the file, %page% and %comp% can be used as the root elements of the path
$filename	file name
$filedata	Content that will be saved in the file

Output:

True	The function was executed successfully
False	An error occured

3.7.5 lockfile

Syntax:
lockfile ($user, $abs_path, $filename)

Description:
The function “lockfile” locks a file for a specific user, so its available for the exclusive use. For this purpose, the user, the absolute path and the file name must be passed as a parameters.

Example:
lockfile ("Miller", "%page%/%publication%/home/", "index.php");

Input parameters:

$user	User name of the user who locked the file
$abs_path	absolute path of the file, %page% and %comp% can be used as the root elements of the path
$filename	file name

Output:

True	The function was executed successfully
False	An error occured

3.7.6 unlockfile

Syntax:
unlockfile ($user, $abs_path, $filename)

Description:
The function “unlockfile” unlocks files that have been previously locked by “lockfile” or opened by “loadlockfile”. For this purpose, the user, the absolute path and the file name must be passed as a parameters.

Example:
unlockfile ("Miller", "%page%/%publication%/home/", "index.php");

Input parameters:

$user	User name of the user who locked the file
$abs_path	absolute path of the file, %page% and %comp% can be used as the root elements of the path
$filename	file name

Output:

True	The function was executed successfully
False	An error occured

3.7.7 deletefile

Syntax:
deletefile ($location, $file, $recursive)

Description:
With “deletefile” files and (empty) folders can be deleted. The absolute path, the file or directory name, and a parameter "recursive", which is either (0) or (1), need to be passed. If recursive is set to 1 the entire contents of the directory will be processed, including subdirectories and their files, using the value 0 only the file or directory (if empty) will be removed.

Example:
deletefile ("%page%/%publication%/home/", "index.php", 0);

Input parameters:

$abs_path	absolute path of the file, %page% and %comp% can be used as the root elements of the path
$file	file name
$recursive	0 or 1, if subdirectories should removed recursively as well

Output:

True	The function was executed successfully
False	An error occured

3.7.8 appendfile

Syntax:
append ($abs_path, $filename, $filedata)

Description:
With “appendfile” content can be added to a file. The function does not overwrite existing data of a file, it appends the data at the file end. For this the absolute path, the file name, and the content that needs to be written to the file must be passed as parameters.

Example:
appendfile ("%page%/%publication%/home/", "index.php", "© 2003 …");

Input parameters:

$abs_path	absolute path of the file, %page% and %comp% can be used as the root elements of the path
$filename	file name
$filedata	Content that will be appended to the file

Output:

True	The function was executed successfully
False	An error occured

3.8 XML library

The following functions allow yout to read and write content from XML content container. You can optionally query the contents of the container with other technologies that can deal with XML. However, the Edit Content library offers a very simple and performant way of doing this.

3.8.1 setxmlparameter

Syntax:
setxmlparameter ($xmldata, $parameter, $value)

Description:
Set a specific value oft he XML declaration (1.row).

Example:
$xmldata = setxmlparameter ($xmldata, "encoding", "UTF-8");

Input parameters:

$xmldata	XML string that should be manipulated
$parameter	Name of the tag that should be manipulated
$value	Value saved in the tag

Output:

XML-String	Return of the manipulated XML string
False	An error occured

3.8.2 getcontent

Syntax:
getcontent ($xmldata, $tag)

Description:
Retrieves the XML content from the content container that is located inside the tags $tag. An array containing all content or childs found will be returned.

Example:
```
// Get all text-childs from the content container
$text_array = getcontent ($xmldata, "<text>");

// Show all text-childs
foreach ($text_array as $text) echo $text;
```

Input parameters:

$xmldata	XML string holding the content
$tag	Name of the tag holding the information or child nodes

Output:

Array	Array holding all found values, the first value can be accessed using the first array element (Array[0])
False	An error occured

3.8.3 getxmlcontent

Syntax:
getxmlcontent ($xmldata, $tag)

Description:
Retrieves the XML content from a content container that is located inside the tags $tag and leaves in contrast to the function “getcontent” the XML tags in the return value (array). An entire node (well-formed) will therefore be returned.
An array containing all content and childs found will be returned and can be stored and used in a variable of type array.

Example:
```
$text_array = getxmlcontent ($xmldata, "<text>");
foreach ($text_array as $text) echo $text;
```

Input parameters:

$xmldata	XML string holding the content
$tag	Name of the tag holding the information or child nodes

Output:

Array	Array holding all found values, the first value can be accessed using the first array element (Array[0])
False	An error occured

3.8.4 selectcontent

Syntax:
selectcontent ($xmldata, $parenttag, $childtag, $childvalue)

Description:
Retrieves the XML content defined by the tag $parenttag from the content container, where the childtag $childtag has a certain value $value.
An array with all items found will be returned and can be stored and used in a variable of type array.

Example:
Extract of a content container:
```
…..
<text>
  <text_id>summary</text_id>
  <textuser>editor1</textuser>
  <textcontent>This is my summary!</textcontent>
</text>
…..

// Get all text-childs with id=summary
$text_array = selectcontent ($xmldata, "<text>", "<text_id>", "summary");

// Extract the summary from the found content
foreach ($text_array as $text)
{
  $summary = getcontent ($text, "<textcontent>");
}
```

Input parameters:

$xmldata	XML string holding the content
$parenttag	Name of the tag holding the information or child nodes
$childtag	optional: XML tag that encloses the information that must be of a certain value
$childvalue	optional: Value of the condition, the wildcard character * can be used at the beginning and/or end of the term and is a wildcard for any further characters.

Output:

Array	Array holding all found values, the first value can be accessed using the first array element (Array[0])
False	An error occured

3.8.5 selectxmlcontent

Syntax:
selectxmlcontent ($xmldata, $parenttag, $childtag, $childvalue)

Description:
Retrieves the XML content defined by the tag $parenttag from the content container, where the childtag $childtag has a certain value $value. In contrast to the function “getcontent” the parent tags are included in the return value (array).
An array with all items found will be returned and can be stored and used in a variable of type array.

Example:
Extract of a content container:

```
…..
<text>
  <text_id>summary</text_id>
  <textuser>editor1</textuser>
  <textcontent>This is my summary!</textcontent>
</text>
…..

// Get all text-childs with id=summary
$text_array = selectxmlcontent ($xmldata, "<text>", "<text_id>", "summary");

// Extract the summary from the found content
foreach ($text_array as $text)
{
  $summary = getcontent ($text, "<textcontent>");
}
```

Input parameters:

$xmldata	XML string holding the content
$parenttag	Name of the tag holding the information or child nodes
$childtag	optional: XML tag that encloses the information that must be of a certain value
$childvalue	optional: Value of the condition, the wildcard character * can be used at the beginning and/or end of the term and is a wildcard for any further characters.

Output:

Array	Array holding all found values, the first value can be accessed using the first array element (Array[0])
False	An error occured

3.8.6 deletecontent

Syntax:
deletecontent ($xmldata, $tagname, $condtag, $condvalue)

Description:
Removes the entire XML content defined by the tag $tagname. For the selection of a certain child the appropriate XML childtag $condtag and the enclosed information as $condvalue as condition can be passed.

Example:
Extract of a content container:

```
…..
<text>
  <text_id>condition</text_id>
  <textuser>editor1</textuser>
  <textcontent>This is my summary!</textcontent>
</text>
…..

$xmldata = deletecontent ($xmldata, "<text>", "<text_id>", "bedingung");
```

Input parameters:

$xmldata	XML string holding the content
$parenttag	Name of the tag holding the information or child nodes that should be removed
$condtag	optional: XML tag that encloses the information that must be of a certain value
$condvalue	optional: Value of the condition

Output:

XML-String	Return of the manipulated XML string
False	An error occured

3.8.7 setcontent

Syntax:
setcontent ($xmldata, $parenttagname, $tagname, $contentnew, $condtag, $condvalue)

Description:
An XML string is passed and within a certain parent node ($parenttagname) a certain parameter ($condtag) must exists and must have a certain value ($condvalue). If the condition is satisfied, the value of the parameter $tagname will be replaced by the new value $contentnew.

Example:
Extract of a content container:

```
…..
<text>
  <text_id>condition</text_id>
  <textuser>editor1</textuser>
  <textcontent>This is should set!<textcontent>
</text>
…..
```

```
$xmldata = setcontent ($xmldata, "<text>", "<textcontent>", "This is my new value!",
"<text_id>", "condition");
```

Input parameters:

$xmldata	XML string holding the content
$parenttagname	optional: XML parent tag
$tagname	optional: XML child tag, that value should be replaced (if the condition is met)
$contentnew	new value for the XML child tag $tagmame
$condtag	optional: XML tag that encloses the information that must be of a certain value
$condvalue	optional: Value for the condition

Output:

XML-String	Return of the manipulated XML string
False	An error occured

3.8.8 updatecontent

Syntax:
updatecontent ($xmldata, $xmlnode, $xmlnodenew)

Description:
All XML strings $xmlnode will be replaced by a new XML string $xmlnodenew in $xmldata. This method is faster than „setcontent" when the updated XML node has already been extracted from the container.

Example:
Extract of a content container:

```
…..
<text>
  <text_id>condition</text_id>
  <textuser>editor1</textuser>
  <textcontent>This is old content!<textcontent>
</text>
…..
```

```
$xmldata = updatecontent ($xmldata, "<textcontent>This is old content!<textcontent> ",
"<textcontent>This is my new content!<textcontent>");
```

Input parameters:

$xmldata	XML string holding the content
$xmlnode	XML string to be replaced (node or substring of $xmldata)
$xmlnodenew	optional: new XML string, if empty, the existing XML string will be removed.

Output:

XML-String	Return of the manipulated XML string
False	An error occured

3.8.9 insertcontent

Syntax:
insertcontent ($xmldata, $insertxmldata, $tagname)

Description:
Inserts an XML string (child node) before the end tag of the given XML parent tag. The modifiied XML string will be returned.

Example:
Extract of a content container:

```
…..
  <articletextlist>
    <text>
      <text_id>art1:summary</text_id>
      <textuser>editor1</textuser>
      <textcontent>This is my summary!</textcontent>
    </text>
---------- The new child node will be inserted here ------------
    <text>
      <text_id>art1:longtext</text_id>
      <textuser>editor1</textuser>
      <textcontent>This is my summary!</textcontent>
    </text>
-------------------------------------------------------------
  </articletextlist>
…..
```

```
$xmldata = insertcontent ($xmldata, $insertxmldata, "<articletextlist>");
```

Input parameters:

$xmldata	XML string holding the content
$insertxmldata	XML string that will be inserted
$tagname	optional: Include xml string before the end tag of the given XML parent tag

Output:

XML-String	Return of the manipulated XML string
False	An error occured

3.8.10 addcontent

Syntax:
addcontent ($xmldata, $sub_xmldata, $grandtagname, $condtag, $condvalue, $parenttagname, $tagname, $contentnew)

Description:
Within a parent node a child node will be added, provided that a certain value in the overlying grandparent node meets the condition. In the child node a value can be set as well. The modified XML string will be returned.

Example:
Extract of a content container:

```
…..
<article>
  <article_id>art1</article_id>
  <articletitle></articletitle>
  <articledatefrom></articledatefrom>
  <articledateto></articledateto>
  <articlestatus>active</articlestatus>
  <articleuser></articleuser>
  <articletextlist>
    <text>
      <text_id>art1:summary</text_id>
      <textuser>editor1</textuser>
      <textcontent>This is my summary!</textcontent>
    </text>
---------- The new child node will be inserted here ------------
    <text>
      <text_id>art1:longtext</text_id>
      <textuser>editor1</textuser>
      <textcontent>This is my summary!</textcontent>
    </text>
-------------------------------------------------------------
  </articletextlist>
</article>
…..
```

```
$xmldata = addcontent ($xmldata, $sub_xmldata, "<article>", "<article_id>", "art1", "<articletextlist>", "<text_id>", "art1:longtext");
```

Input parameters:

$xmldata	XML string holding the content
$sub_xmldata	XML string that will be inserted
$grandtagname	XML child tag where the $sub_xmldata should be embedded
$condtag	optional: XML tag that encloses the information that must be of a certain value
$condvalue	optional: Value for the condition
$parenttagname	optional: XML child tag, where $sub_xmldata should be embedded
$tagname	optional: Child tag of the embedded XML string where a value will be set
$contentnew	optional: Value for the $tagname

Output:

XML-String	Return of the manipulated XML string
False	An error occured

3.9 Meta Data Generator library

This function library allows you to automatically create keyword lists and a description from a given content. This can be used to automatically generate and fill in metadata for pages. The meta data from multimedia files can be extracted and stored in the container of an object.

3.9.1 getmetakeywords

Syntax:
getkeywords ($text, $language, $charset)

Description:
The function requires the text content to be passed as input. All keywords are determined from the text and returned as a keyword list.

Example:
$keywords = getkeywords ("This is just a short text.", "en", "UTF-8");

Input parameters:

$text	Content als String
$language	optional: Language [en, de], default is "en"
$charset	optional: Character set, default is "UTF-8"

Output:

Keywords	Comma seperated list of keywords
False	An error occured

3.9.2 getmetadescription

Syntax:
getdescription ($text, $charset)

Description:
The function requires the text content to be passed as input. A brief description from the given text will be extracted and returned.

Example:
$keywords = getdescription ("This is just a short text.", "UTF-8");

Input parameters:

$text	Content as string
$charset	optional: Character set, default is "UTF-8"

Output:

Keywords	Short description of the content
False	An error occured

3.10 Notifications library

This function library sends automated messages to a user based on limits of a certain value of a particular field.
The user receives a pre-formatted message with information (links) to all objects that are within the search area (date upper and lower limit).

3.10.1 licensenotification

Syntax:
licensenotification ($site, $cat, $folderpath, $text_id, $date_begin, $date_end, $user)

Description:
The function returns all objects due to the specified search range (location and date limits) and sends an e-mail to a specific user with links to all the affected objects.

Example:
```
// set language for mail message
$lang = "en";

// send mail to Miller
$result = licensenotification ("%publication%", "%comp%/%publication%/images/", "comp", "valid_date", "2012-09-01", "2012-09-30", "Miller");
```

Input parameters:

$site	Name of the publication
$cat	Object category [page, comp]
$folderpath	Location for the defintion of the search area
$text_id	Text ID of the XML node that need to be analyzed
$date_begin	Start date for the seach (YYYY-MM-DD)
$date_end	End date for the seach (YYYY-MM-DD)
$user	User name

Output:

True	Mail wurdwas send successfully
False	An error occured

4 Components and applications

If applications are integrated into components and variables need to be passed from a page to a component, you need to pay attention to the following:
The components must be integrated via the file system (not via HTTP).
All variables to be passed to the component need to be defined in the component as global.

Example:
A page passes a variable to a component.

Code example of a page:

```
<html>
<head>
<title>page</title>
<head>
<body>
<?php $test="This is just a test!"; ?>
[hyperCMS:components id='component']
</body>
</html>
```

The code of the component must be as followed:

```
<p>
<?php
global $test;
echo $test;
?>
</p>
```

In the example, the variable $test and its value "This is just a test!" will be passed to the component and will be displayed in the presentation of the component.

5 Database Connectivity

The Database Connectivity of the hyper Content Management Server allows the connection of different databases for the storage and retrieval of content. Relational databases are widely used as an external content repository.
For this purpose, a corresponding hyperCMS tag for the Database Connectivity need to be present in each template, which then refers to a DB-Connect file.
In this file functions are stored that hyperCMS will call, provided that the template points to the function file.
The contents are read from the database and will be displayed to the editor. If the editor modifies the content, it will be written to the database again. For read and write access different databases can be accessed as well. The functions in the DB Connect File offer only the shell or standardized interface to hyperCMS that needs to be filled by the developer.

The subject of database integration is complex and need to be treated individual, since existing databases and their information must be integrated. hyperCMS does not provide any ER model or commits itself to specific database vendors. In general it can be said that all the possibilities of PHP can be exploited in order to connect to various data sources.

Besides the necessary parameters for queries to relational databases, the entire content conatiner is passed as an XML string. This allows documents or content from the content repository to be stored as a node in XML databases as well.

You therfore decide from which sources you read and save data. With PHP you have a powerful language that gives you access to all major databases.

More information regarding the functions of PHP can be found here: http://www.php.net

5.1 Creating a Database Connectivity

If you want to create a Database Connectivity, you need to make a copy of the the file “db_connect_default.php”. This file can be found in the root directory for storing the management data under the following path: /data /db_connect/
The copy of the file should be named according to the database you want to connect with.

Open the file and gain insight into the functions. In the source code you will also find a description of the functions and parameters passed as well as the output parameters.

The following example will show a read access to a MySQL database in order to extract a text content. We assume that in a table "TextContent" the content will be presented by the primary key "container_id" and "text_id" as well as the text content "text" itself and the text-type "type". The user and the article ID is not stored separately, this is also not necessary for the uniqueness of the content, because the ID of the content container as well as the ID of the element already provides the primary key.

```php
// ============================ db connect  ==============================
// this file allows you to access a database using the full PHP functionality.
// you can read or write data from or into a database:

// ========================= read from database ==========================
// the following parameter values are passed to each function for
// retrieving data from the database:
// name of the site: $site [string]
// name of the content container: $container_id [string] (is unique
// inside hyperCMS over all sites)
// content container: $container_content [XML-string]
// identification name: $id [string]

// ------------------------------- text -----------------------------------
// if content is text
function db_read_text ($site, $content_id, $container_content, $id, $art_id, $user)
{
  //-------------------------------------------------------------------------
  // input variables: $id [string], optional: $artid [string], $user [string
  // return value: $text [array]
  //             the array must exactly look like this:
  //             $text[text], optional: $text[type]
  //             constraints/accepted values for article type, see note below
  // note: special characters in $text are escaped into
  //       their html/xml equivalents.
  //       you can decide between unformatted, formatted and
  //       optional text using $type:
  //       unformatted text: $text[type] = textu
  //       formatted text: $text[type] = textf
  //       text option: $text[type] = textl
  //-------------------------------------------------------------------------

  $user = "username";
  $password  = "password";
  $database = "database";

  // connect to database
  mysql_connect ("localhost", $user, $password);
  @mysql_select_db ($database) or die ("Unable to select database");

  // fire SQL-query
  $result = mysql_query ("SELECT Text, Type FROM TextContent WHERE
          container_id=$container_id AND text_id=$id);

  // count returned rows, must be 1 if unique
  $num_of_rows = mysql_num_rows ($result);

  // get the result into an array namend $row
  if ($num_of_rows == 1)
  {
    $row = mysql_fetch_row ($result);

    // set values
    $text[text] = $row[0];
    $text[type] = $row[1];
  }
  else $text = false;

  // close connection
  mysql_close ();

  // return result
  return $text;
}
```

6 Event System

The hyper Content & Digital Asset Management Server provides an event system that allows automated execution of actions when events in the system will be executed. This can be used, for example, to automate manual processes.
Events are usually started by the user by selecting an action, e.g. publishing a page. If he corresponding event is enabled, the event "onpublishobject" will be called after successful execution of the publication process of the page. The functions defined therein will therfore be executed.
The events of the event system can be defined in the "hypercms_eventsys.inc.php" file. This file is located in the internal repository in the folder "data/event system". In this file there are also other important instructions that must be followed during the execution of events.

The event system is valid within the management system for all publications. The system is part of the hyperCMS API and is thus performed on each invocation of the API functions.

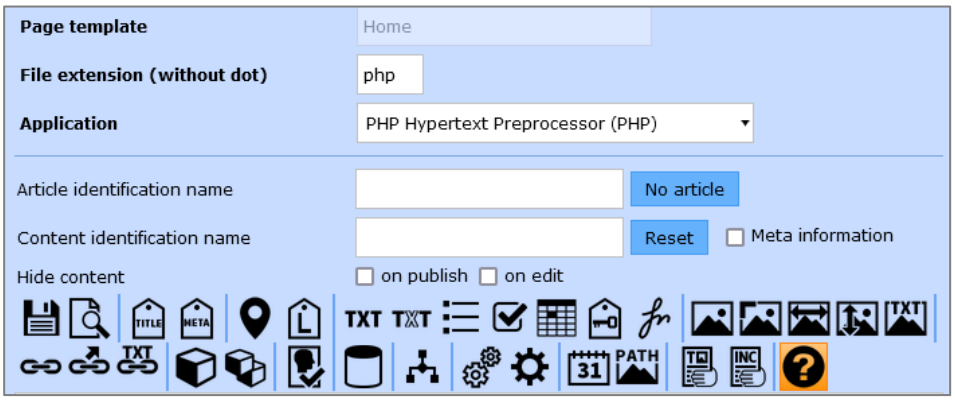
Events can be activated and deactivated, so that the use of its defined events can be easily controlled in the "hypercms_eventsys.inc.php" file.
There is a distinction between PRE and POST events. The PRE event will be fired before the actual execution of the called action, while the POST event is called after the successful execution of the action.

Example:
When publishing an object, the page “index.php” located at the same position should be automatically published as well, since the page “index.php” is used to generate an overview using a hyperCMS script all objects of the same folder.

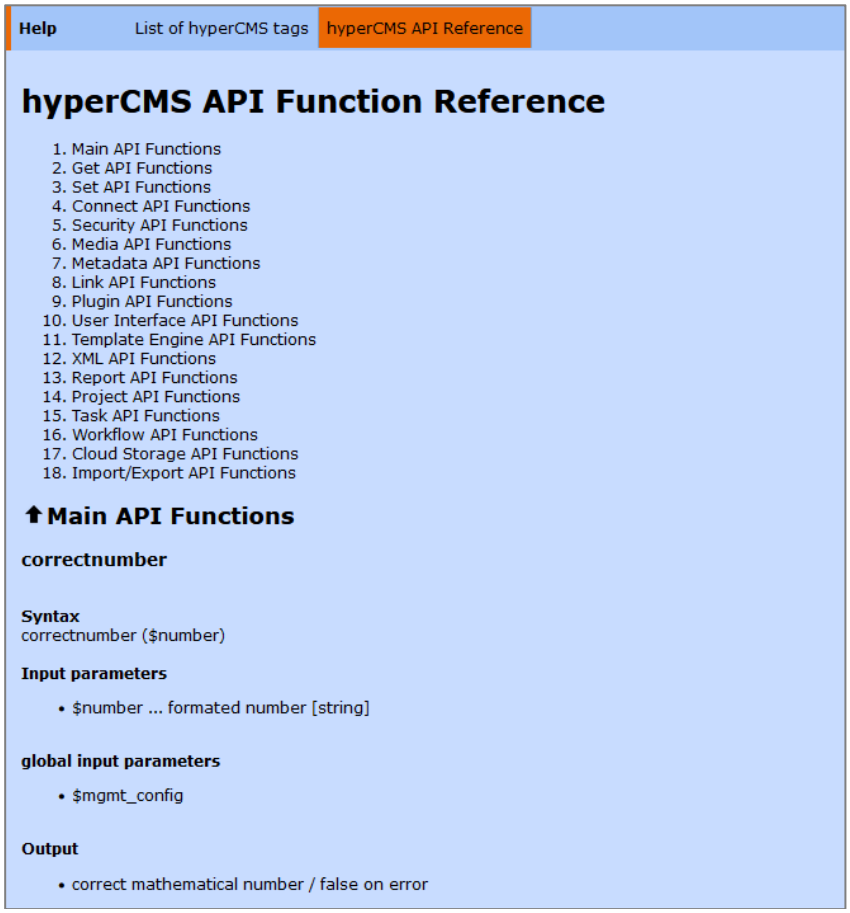
```
// ------------------------------------ on publish object POST event ---------------------------------------
function onpublishobject_post ($site, $cat, $location, $object, $user)
{
  // load configuration
  include_once ("config.inc.php");

  // hide the event used in your action (1) otherwise execute event (0)
  $eventsystem[hide] = 1;

  // insert your code here
  $result = publishobject ($site, $location, "index.php", $user);

  // return true if successful
  if ($result[result) == true) return true;
  else return false;
}
```

7 hyperCMS API Function Reference

The documentation of all API functions of the current version is available on our website hypercms.com.
You can view the documentation of your system directly in the browser. Click on the ?-Icon in the template editor to access the help with all hyperCMS tags and API functions.



The help will be opened in a new pop-up window. Click on the "hyperCMS API Reference" tab to view the API Reference. You can open the search using the key combination Ctrl + F.



8 Legal reference / flag

8.1 Questions and suggestions

For advanced questions and suggestions, please contact the support.

hyperCMS Support:
support@hypercms.com
http://www.hypercms.com

8.2 Imprint

Responsible for the content:

hyperCMS
Content Management Solutions GmbH
Rembrandtstr. 35/6
A-1020 Vienna – Austria

office@hypercms.com
http://www.hypercms.com

8.3 Legal information

The present product information is based on the version of the program, which was available at the time the document was composed.

The maker reserves the rights of modifications and corrections of the program.
Errors and misapprehension accepted.