hyperCMS

Content & Digital Asset Management Server

Version 10.0

# Developer Guide

2022-11-02

# Inhalt

# 1 Einleitung

Die folgenden Kapitel behandeln die Funktionsbibliotheken des hyper Content & Digital Asset Management Servers und stellen somit die Dokumentation des API (Application Programming Interface) dar.
Alle Bibliotheken befinden sich innerhalb der hyperCMS Installation im Ordner "function" und können in die jeweiligen Scripts bzw. Templates eingebunden und genutzt werden. Damit lassen sich z.B. auch dynamische Seiten (Applikationen) unter Einsatz des XML-Content-Repository programmieren.
Sollten Sie Ihre Applikation auf einen physisch getrennten Server betreiben, so ist es wichtig, dass die Funktionsbibliotheken auch auf dem Publikationsserver zur Verfügung stehen. In diesem Fall ist es wichtig, dass die entsprechenden Dateien auch am Publikationsserver zur Verfügung stehen.

# 2 hyperCMS XML-Content-Repository

Das XML-Content-Repository beinhaltet alle XML-Content-Container und stellt somit alle Inhalte native XML zur Verfügung. Die Struktur (Schema) innerhalb eines XML-Content-Containers wird auf Basis des verwendeten Templates dynamisch erzeugt und besitzt folgendes Aussehen:

```
<?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>
```

Nach Durchsicht des Content Containers ist eine Struktur zu erkennen, die sich aus den folgenden wesentlichen Grundelementen für die Content-Ablage zusammensetzt:

- hyperCMS spezifische Informationen
- Meta-Informationen
- Text
- Medien (Bilder oder andere Multimedia-Dateien)
- Links
- Komponenten
- Artikel

Der gesamte Inhalt setzt sich aus diesem Grundbausteinen zusammen, deren Informationen wiederum innerhalb von XML-Tags abgelegt werden.
Artikel nehmen so wiederum die Elemente Text, Medien und Links in sich auf. Der gesamte Inhalt einer Seite oder Komponente lässt sich über den zugehörigen Content-Container beziehen.

## 2.1 hyperCMS spezifische Informationen

Die in diesem XML-Knoten erfassten Daten stellen primär für das Management des Containers relevante Informationen dar.

```
<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>
```

**Erklärung:**

| | |
|---|---|
| contentcontainer | Name des Content Containers (einmalig über alle Publikationen) |
| contentxmlschema | Schema des Objektes: Seite = page oder Komponente = comp |
| contentorigin | Objekt (Seite oder Komponente) die zur Generierung des Content Containers führte |
| contentobjects | Alle Objekte die diesen Content Container benutzen |
| contentuser | Objekteigentümer |
| contentdate | Datum der letzten Änderung des Containers |
| contentpublished | Datum der letzten Publizierung eines Objektes basierend auf den Content Container |
| contentstatus | Der Status ist "active" solange ein Objekt das auf den Container basiert existiert. Wurden alle Objekte die auf den Container basieren entfernt wird der Status "deleted" gesetzt. Der Container beinhaltet damit den letzten Informationsstand, kann jedoch nicht mehr genutzt werden. |

## 2.2 Meta-Informationen

Die Standard Meta-Informationen einer HTML-Seite werden in diesem XML-Knoten beschrieben.

```
<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>
```

**Erklärung:**

| | |
|---|---|
| pagetitle | Seitentitel |
| pageauthor | Seitenautor |
| pagedescription | Beschreibung der Inhalte der Seite |
| pagekeywords | Liste der Schlüsselwörter der Seite |
| pagecontenttype | Content-Type (Zeichensatz) der Seite oder Komponente |
| pagelanguage | Sprachkürzel der Seite |
| pagerevisit | Wiederbesuch der Seite durch Suchmaschinen |

## 2.3 Text

Diese XML-Knoten speichern den Text.

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

**Erklärung:**

| | |
|---|---|
| text_id | Textidentifikation |
| textuser | Texteigentümer (letzte Änderung des Textes durch einen Benutzer) |
| textcontent | Inhalt des Textes |

## 2.4 Medien

Dieser XML-Knoten beschreibt eingebunden Medien.

```
<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>
```

Erklärung:

| | |
|---|---|
| media_id | Medienidentifikation |
| mediauser | Medieneigentümer (letzte Änderung des Mediums durch einen Benutzer) |
| mediafile | eingebunden Mediendatei mit Angabe der Publikation |
| mediaobject | Pfadangabe zur Multimediakomponente |
| mediaalttext | Alternativtext des Mediums |
| mediaalign | Ausrichtung des Mediums |
| medawidth | Dargestellte Breite des Mediums |
| mediaheight | Dargestellte Höhe des Mediums |

## 2.5 Links

Dieser XML-Knoten beschreibt die Verlinkung zu Seiten.

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

**Erklärung:**

| | |
|---|---|
| link_id | Linkidentifikation |
| linkuser | Linkeigentümer (letzte Änderung des Links durch einen Benutzer) |
| linkhref | Referenz (Link) zu einer Seite oder Datei |
| linktarget | Ziel des Referenzierung (Name des Frames) |
| linktext | Text der den Link beschreibt/darstellt |

## 2.6 Komponenten

Dieser XML-Knoten beschreibt die Verlinkung zu Komponenten.

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

**Erklärung:**

| | |
|---|---|
| component_id | Komponentenidentifikation |
| componentuser | Komponenteneigentümer (letzte Änderung der Komponentenreferzierung durch einen Benutzer) |
| componentcond | Zugeordnetes Kundenprofil zu der Komponente |
| componentfiles | Referenz (Komponenten-Link) zu einer oder mehreren Komponenten |

## 2.7 Artikel

Dieser XML-Knoten beschreibt die Artikelinformation.

```
<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>
```

**Erklärung:**

| | |
|---|---|
| article_id | Artikelnidentifikation |
| articletitle | Titel des Artikels |
| articeldatefrom | Beginn der Veröffentlichung des Artikels |
| articeldateto | Ende der Veröffentlichung des Artikels |
| articlestatus | Bestimmung der Veröffentlichung des Artikels:<br>active = immer veröffentlicht<br>inactive = nicht veröffenlicht<br>timeswitched = zeitgesteuerte Veröffentlichung |
| articleuser | Artikeleigentümer (letzte Änderung des Artikels durch einen Benutzer) |
| articlecollection | Umfasst alle dem Artikel zugeordneten Inhalte |

# 3 Funktionsbibliotheken

## 3.1 Einbindung einer Bibliothek

Das Einbinden einer Konfiguratzion oder Bibliothek setzt voraus, dass man den absoluten oder relativen Pfad zur Bibliothek kennt. Durch Verwendung der Funktion "require" oder "require_once" und der Angabe des Pfades inklusive der einzubinden Datei werden die enthaltenen Funktionen der Bilbliothek eingebunden. Sobald die Bibliothek eingebunden ist, können deren Funktionen im Script genutzt werden.
Um die hyperCMS-Funktionen nutzen zu können, bedarf es der Einbindung der Datei "hypercms_api.inc.php". Diese Datei beinhaltet alle für die Programmierung benötigten Funktionen.

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

// relative Angabe unter MS Windows oder auch UNIX-Derivaten
require_once ("function/hypercms_api.inc.php");
```

## 3.2 Laden der Konfiguration

### 3.2.1 Content Management Server

Um die Konfiguration von hyperCMS nutzen zu können muss die entsprechende Datei geladen warden. Diese beinhaltet alle wesentlichen Einstellungen des zu behandelnden Mandanten (Site).
Mit Hilfe der Identifikation einer Publikation, z.B. mit der Variable $site kann die Konfiguration einer Publikation geladen werden. Die hyperCMS Hauptkonfigurationsatei befindet sich im Verzeichnis „hypercms/config“ und trägt den Namen "config.inc.php". Die publikationsspezifischen Konfigurationsateien befinden sich im Verzeichnis "data/config". Deren Dateiname setzt sich aus dem Namen der Publikation sowie der Endung ".inc.php" zusammen, Bsp: site.inc.php.

```
// Einbinden der Hauptkonfigurationsatei (auf Pfadangabe ist zu achten):
require_once ("C:/inetpub/wwwroot/hypercms/config.inc.php");

// Einbinden Konfiguration einer Publikation
// Achtung: Bitte verwenden Sie valid_publicationname, um den Namen zu verifizieren, bevor
// Sie die Datei einbinden
if (valid_publicationname ($site))
{
  require_once ($mgmt_config['abs_path_data']."config/".$site.".conf.php");
}
```

Die Config-Dateien können geöffnet und gelesen werden. Jeder Parameter wird darin beschrieben und steht für die Nutzung in Programmen zur Verfügung. Bitte werfen Sie daher einen Blick in die Konfiguration, um mehr über die Parameter und deren Namen zu erfahren.

Es ist auch notwendig eine Sprache zu wählen. Hierfür dient die Variabel $lang. $lang beinhaltet das Sprachkürzel, welche in der Konfiguration "hypercms/config/config.inc.php" eingesehen werden können.

```
// Setzen der Spracheinstellung für Nachrichten von Funktionen, Deutsch (de)
$lang = "de";
```

Da Sie die Funktionen des hyper API benutzen möchten, müssen Sie auch noch dieses einbinden.

```
// Einbinden der Funktionsbibliothek:
require_once ($mgmt_config['abs_path_cms']."/function/hypercms_api.inc.php");
```

Nun können Sie die Funktionen des APIO nutzen, um z.B. einen Content Container einer bestimmtes Objektes über unterschiedliche Methoden zu laden:

```
// Laden der Seite
$pagedata = loadfile ("%page%/MyPublication/home/", "index.php");

// Content Container Name auslesen
$contentcontainer  = filepointer ($pagedata, "content");

// Laden des veröffentlichten Content Container aus dem Content Repository
$containerdata = loadcontainer ($contentcontainer, "published", $user);

// Oder noch einfacher direkt über den Objektpfad
$containerdata = getobjectcontainer ("MyPublication", "%page%/MyPublication/home/",
"index.php", $user);
```

Funktionen laden die Konfiguration einer Publikatzion, sollte diese nicht verfügbar sein. Da viele Funktionen die Einstellungen einer Publikation benötigen, ist es ratsam die Konfiguration immer einzubinden.

### 3.2.2 Publication Server

Beachten Sie, dass die Konfiguration des Publication Servers (Publikationsziel) davon getrennt in einer INI-Datei abgelegt ist. Benötigen Sie die Publikationziel-Einstellungen, so müssen Sie die INI-Datei laden und parsen. Danach stehen Ihnen die Variablen in einem Array zur Verfügung.
Die INI-Datei des Publikationszieles befindet sich im externen Repository im Verzeichnis "repository/config". Der Name der Datei entspricht dem Namen der Publikation mit der Dateierweiterung ".ini".

```
// laden und parsen der INI-Datei mit hilfe von PHP
$publ_config = parse_ini_file ("C:/inetpub/wwwroot/repository/config/Mandant_1.ini");

// Zugreifen auf die Variablen des Publikationszieles
echo "Das ist der Document Root der Seiten der Publikation:".$publ_config[abs_publ_page];
```

## 3.3 Globale Variablen

Viele Funktionen nutzen globale Variablen die in der Konfiguration gespeichert sind und den Funktionen zur Verfügung stehen. Sie sollten daher bei der Wahl der Variablennamen in Ihren eigenen Scripts acht geben, dass Sie nicht die von hyperCMS genutzen globalen Variablen verwenden.
Die folgende Liste zeigt alle globalen Variablen von hyperCMS, die nicht in eigenen Scripts manipuliert/verändert werden dürfen:

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

Viele globale Variablen von hyperCMS sind für die Verwendung in hyperCMS-Scripts und PHP-Scripts nützlich, diese stehen nur dann zur Verfügung, wenn die entsprechende Konfiguration zuvor geladen wurde, oder eine hyperCMS-Script (wird nur während des Publikationsprozesses ausgeführt) in Verwendung ist. Da dies bei der Voransicht als auch beim Publizieren von Seiten und Komponenten passiert, können diese Variablen in hyperCMS-Scripts genutzt werden. Bei dynamischen Applikationen, die bei jedem Aufruf der Seite oder Komponente durch einen Besucher ausgeführt werden, muss die Konfiguration direkt im Template eingebunden werden, sofern Variablen von hyperCMS benötigt werden.

**Content Management Server:**

| | |
|---|---|
| **$lang** | Sprachkürzel lt. config.inc.php |
| **$mgmt_config[**'url_path_cms'**]** | URL des hyperCMS Root Verzeichnis lt. config.inc.php |
| **mgmt_config[**'abs_path_cms'**]** | absoluter Pfad zum hyperCMS Root Verzeichnis lt. config.inc.php |
| **$mgmt_config[**'url_path_page'**]** | URL des Doc Roots der Publikation im Managementsystem |
| **$mgmt_config[**'abs_path_page'**]** | absoluter Pfad zum Doc Roots der Publikation im Managementsystem |
| **$mgmt_config[**'url_path_comp'**]** | URL des Komponenten Root der Publikation im Managementsystem |
| **mgmt_config[**'abs_path_comp'**]** | absoluter Pfad zum Komponenten Roots der Publikation im Managementsystem |

**Publication Server:**
hyperCMS-Scripts können die Variablen ohne weiteres zutun nutzen. Die Werte werden im Array $publ_config gespeichert, sind aber auch optional auch ohne Array nutzbar. Wird das Script/Anwendung bei jedem publikationsseitigen Aufruf ausgeführt, so ist die Konfigurationsdatei gesondert zu laden.

| | |
|---|---|
| **$publ_config[**'url_publ_page'**]** | URL des Doc Roots der Publikation im Publikationssystem |
| **$publ_config[**'abs_publ_page'**]** | absoluter Pfad zum Doc Roots der Publikation im Publikationssystem |
| **$publ_config[**'url_publ_comp'**]** | URL des Komponenten Roots der Publikation im Publikationssystem |
| **$publ_config[**'abs_publ_comp'**]** | absoluter Pfad zum Komponenten Roots der Publikation im Publikationssystem |

Optional (veraltet):

| | |
|---|---|
| **$url_publ_page** | URL des Doc Roots der Publikation im Publikationssystem |
| **$abs_publ_page** | absoluter Pfad zum Doc Roots der Publikation im Publikationssystem |
| **$url_publ_comp** | URL des Komponenten Roots der Publikation im Publikationssystem |
| **$abs_publ_comp** | absoluter Pfad zum Komponenten Roots der Publikation im Publikationssystem |

## 3.4 Vorlagenvariablen

Es gibt auch die Möglichkeit mit hyperCMS-eigenen Vorlagenvariablen zu arbeiten. Diese Variablen stellen eine Besonderheit dar, da sie nicht mit hyperCMS-Script in Verbindung stehen müssen. Sie sind vielmehr Platzhalter für den Wert einer Variable und können in Vorlagen beliebig eingesetzt werden.
Diese neutrale Form der Variablen sollte primär in Templates Verwendung finden, da damit ein technologieneutraler Einsatz stattfinden kann.
Achten Sie bitte auf die Kleinschreibung aller Variablen!

| | |
|---|---|
| **%container%** | steht für den Namen des Content Containers eines Objektes. |
| **%container_id%** | steht für die ID des Content Containers eines Objektes. |
| **%objecthash%** | steht für den Hash eines Objektes |
| **%object_id%** | steht für die ID eines Objektes |
| **%template%** | steht für den Dateinamen der verwendeten Vorlage des Objektes. |
| **%publication%** | steht für die Publikation in dem sich das Objekte befindet. |
| **%url_location%** | steht für die absolute Pfadangabe (URL) der Position an dem sich das aktuelle Objekt befindet. |
| **%abs_location%** | steht für die absolute Pfadangabe im Dateisystem der Position an dem sich das aktuelle Objekt befindet |
| **%object%** | steht für den Namen des Objektes. |
| **%date%** | beschreibt das aktuelle Datum im Format JJJJ-MM-TT. |
| **%view%** | beschreibt den Anzeigemodus:<br>publish … publiziert<br>cmsview … Bearbeitungsansicht im EasyEdit Modus<br>preview … Voransicht<br>formedit … Bearbeitunsgsmodus in Formularansicht<br>formlock … Formularansicht mit gesperrter Bearbeitung<br>formmeta … Metadaten in Formularansicht<br>template … Template-Voransicht |

Für die Einbindung von Mediendateien wird eine Pfadvariable benutzt. Diese Pfadvariable wird beim Publizieren der Seite oder Komponente z.B. durch die URL (Adresse) der Konfiguration des Publikationszieles ersetzt:

| | |
|---|---|
| **%tplmedia%** | steht für die absolute Pfadangabe (URL) des Vorlagen Medien Repository. |
| **%url_media%** | steht für die absolute Pfadangabe (URL) des Content Medien Repository (Alternativ kann auch **%media%** verwendet werden) |
| **%abs_media%** | steht für die absolute Pfadangabe im Dateisystem des Content Medien Repository. |

Auch die publikationsseitigen Wurzelverzeischnisse der Seiten und Komponenten lassen sich abrufen:

| | |
|---|---|
| **%url_page%** | steht für die absolute Pfadangabe (URL) des Seiten-Wurzelverzeichnisses. |
| **%abs_page%** | steht für die absolute Pfadangabe im Dateisystem des Seiten-Wurzelverzeichnisses. |
| **%url_comp%** | steht für die absolute Pfadangabe (URL) des Komponenten-Wurzelverzeichnisses. |
| **%abs_comp%** | steht für die absolute Pfadangabe im Dateisystem des Komponenten-Wurzelverzeichnisses. |
| **%url_rep%** | steht für die absolute Pfadangabe (URL) des externen Repository-Wurzelverzeichnisses. |
| **%abs_rep%** | steht für die absolute Pfadangabe im Dateisystem des externen Repository-Wurzelverzeichnisses. |

**%url_hypercms%** steht für die absolute Pfadangabe (URL) des hyperCMS-Wurzelverzeichnisses.
**%abs_ hypercms%** steht für die absolute Pfadangabe im Dateisystem des hyperCMS-Wurzelverzeichnisses.

Achten Sie darauf „/“ am Ende der Pfadvariable zu ergänzen, wenn die Variable durch einen weiterführenden Pfad ergänzt werden soll.

**Definition des Datumsformats bei Verwendung des format-Attributes im textd-Tag:**

%a 'am' oder 'pm'
%A 'AM' oder 'PM'
%d Tag des Monats, 2 Stellen mit führender Null (01 bis 31)
%D Wochentag als Text in Kurzform, z.B. "Fre"
%F Monat als Text in Langform, z.B. "Januar"
%h Stunde, 12-Stunden Format (01 bis 12)
%H Stunde, 24-Stunden Format (00 bis 23)
%g Stunde, 12-Stunden Format ohne führender Null (1 bis 12)
%G Stunde, 24-Stunden Format ohne führender Null (0 bis 23)
%i Minuten (00 bis 59)
%j Tag des Monats ohne führender Null (1 bis 31)
%l Wochentag als Text in Langform, z.B. "Freitag"
%L 1 falls Schaltjahr, sonst - 0
%m Monat (01 bis 12)
%n Monat ohne führender Null (1 bis 12)
%M Monat als Text in Kurzform, z.B. "Jan"
%s Sekonden (00 bis 59)
%t Anzahl der Tage im Monat (28 bis 31)
%w Wochentag numerisch (0, Sonntag bis 6, Samstag)
%Y Jahr, 4-stellig, z.B. "2007"
%y Jahr, 2-stellig, z.B. "07"
%z Tag des Jahres (1 bis 366)

In Zusammenhang mit der Nutzung des hyperCMS APIs ist es oft ratsam, bei Pfadangaben die Platzhalter %page% und %comp% zu nutzen. Diese Pfadvariablen lassen sich nur managementseitig nutzen, sie stehen für die Pfade zu den Wurzelverzeischnissen von Seiten und Komponenten.
Zu beachten ist, dass die Variable immer gepaart mit dem Publikationsnamen das Wurzelverzeichnis bildet, z.B:

%page%/%publication%/ …. Wurzelverzeichnis der Seiten der aktuellen Publikation

**%page%/Publikationsname/** steht für die absolute Pfadangabe des Seiten-Wurzelverzeichnisses.
**%comp%/Publikationsname/** steht für die absolute Pfadangabe des Komponenten - Wurzelverzeichnisses.

## 3.5 Bibliothek Object Operation

Diese Bibliothek beinhaltet alle Funktionen für die Manipulation von Objekten (Seiten, Komponenten oder Dateien). Bitte benutzen Sie ausschließlich diese Funktionen für den Zugriff auf Objekte, die das System verwaltet.

### 3.5.1 createfolder

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

**Beschreibung:**
Erzeugt einen neuen Ordner.

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

**Input-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $location | absoluter Pfad (Position des neuen Ordners) |
| $foldernew | Name des neuen Ordners |
| $user | Benutzername |

**globale Input-Parameter:**
Die folgenden globalen Input Parameter sind ebenfalls der Funktion zu übergeben:

| | |
|---|---|
| $lang | Spracheinstellung bzw. Sprachkürzel, z.B. "en", "de" |

**Output:**

| | |
|---|---|
| Array | Array $result das folgende Informationen beinhaltet: |
| | |
| $result[result] | True/False (Konnte der neue Ordner angelegt werden) |
| $result[add_onload] | JavaScript Code für das onLoad Event |
| $result[message] | Nachricht über den Erfolg der Aktion bzw. Fehlermeldung |
| $result[folder] | Name des Ordners |

### 3.5.2 deletefolder

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

**Beschreibung:**
Entfernt einen bestehenden Ordner. Der Ordner wird nur dann entfernt, wenn er keine Objekte mehr beinhaltet. Alle Objekte müssen daher zuvor mit deleteobject entfernt werden.

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

**Input-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $location | absoluter Pfad (Position des neuen Ordners) |
| $folder | Name des zu entfernenden Ordners |
| $user | Benutzername |

**globale Input-Parameter:**
Die folgenden globalen Input Parameter sind ebenfalls der Funktion zu übergeben:

| | |
|---|---|
| $lang | Spracheinstellung bzw. Sprachkürzel, z.B. "en", "de" |

**Output:**

| | |
|---|---|
| Array | Array $result das folgende Informationen beinhaltet: |
| | |
| $result[result] | True/False (Konnte der Ordner entfernt werden) |
| $result[add_onload] | JavaScript Code für das onLoad Event |
| $result[message] | Nachricht über den Erfolg der Aktion bzw. Fehlermeldung |
| $result[folder] | Name des bestehenden Ordners bei Misserfolg, ansonst leer |

### 3.5.3 renamefolder

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

**Beschreibung:**
Benennt einen bestehenden Ordner um.

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

**Input-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $location | absoluter Pfad (Position des Ordners) |
| $folder | Alter Name des Ordners |
| $foldernew | Neuer Name des Ordners |
| $user | Benutzername |

**globale Input-Parameter:**
Die folgenden globalen Input Parameter sind ebenfalls der Funktion zu übergeben:

| | |
|---|---|
| $lang | Spracheinstellung bzw. Sprachkürzel, z.B. "en", "de" |

**Output:**

| | |
|---|---|
| Array | Array $result das folgende Informationen beinhaltet: |
| | |
| $result[result] | True/False (Konnte der Ordner umbenannt werden) |
| $result[add_onload] | JavaScript Code für das onLoad Event |
| $result[message] | Nachricht über den Erfolg der Aktion bzw. Fehlermeldung |
| $result[folder] | Name des Ordners |

### 3.5.4 createobject

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

**Beschreibung:**
Erzeugt eine neue Seite oder Komponente auf Basis einer Vorlage. Bitte beachten Sie das die Position ($location) auch die Kategorie des Objektes (Seite/Komponente) bestimmt. Dies bedingt weiters, dass es sich beim Wert des Parameters $template um eine gültige Seiten- bzw. Komponentenvorlage handeln muss.

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

**Input-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $location | absoluter Pfad (Position des Objektes) |
| $object | Name des neuen Objektes (Seite oder Komponente) |
| $template | Name der zu verwendenden Seiten- oder Komponentenvorlage (Name der Vorlage oder Dateiname) |
| $user | Benutzername |

**globale Input-Parameter:**
Die folgenden globalen Input Parameter sind ebenfalls der Funktion zu übergeben:

| | |
|---|---|
| $lang | Spracheinstellung bzw. Sprachkürzel, z.B. "en", "de" |

**Output:**

| | |
|---|---|
| Array | Array $result das folgende Informationen beinhaltet: |
| | |
| $result[result] | True/False (Erfolg der Aktion) |
| $result[add_onload] | JavaScript Code für das onLoad Event |
| $result[message] | Nachricht über den Erfolg der Aktion bzw. Fehlermeldung |
| $result[object] | Dateiname der Seite oder Komponente |
| $result[objectname] | Name der Seite oder Komponente |
| $result[objecttype] | Filetype bzw. File Extension der Datei |

### 3.5.5 deleteobject

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

**Beschreibung:**
Entfernt eine bestehende Seite, Datei oder Komponente.

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

**Input-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $location | absoluter Pfad (Position des Objektes) |
| $object | Name des Objektes |
| $user | Benutzername |

**globale Input-Parameter:**
Die folgenden globalen Input Parameter sind ebenfalls der Funktion zu übergeben:

| | |
|---|---|
| $lang | Spracheinstellung bzw. Sprachkürzel, z.B. "en", "de" |

**Output:**

| | |
|---|---|
| Array | Array $result das folgende Informationen beinhaltet: |
| | |
| $result[result] | True/False (Erfolg der Aktion) |
| $result[add_onload] | JavaScript Code für das onLoad Event |
| $result[message] | Nachricht über den Erfolg der Aktion bzw. Fehlermeldung |
| $result[publication] | Name der Publikation bzw. Mandant in dem das Objekt existiert |
| $result[location] | absoluter Pfad im Filesystem (Position des Objektes) |
| $result[object] | Dateiname der Seite, Datei oder Komponente |
| $result[objectname] | Name der Seite, Datei oder Komponente |
| $result[objecttype] | Filetype bzw. File Extension der Datei |

### 3.5.6 renameobject

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

**Beschreibung:**
Umbenennen eine bestehender Seite, Datei oder Komponente.

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

**Input-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $location | absoluter Pfad (Position des Objektes) |
| $object | Alter Name des Objektes |
| $objectnew | Neuer Name des Objektes (ohne Dateiendung) |
| $user | Benutzername |

**globale Input-Parameter:**
Die folgenden globalen Input Parameter sind ebenfalls der Funktion zu übergeben:

| | |
|---|---|
| $lang | Spracheinstellung bzw. Sprachkürzel, z.B. "en", "de" |

**Output:**

| | |
|---|---|
| Array | Array $result das folgende Informationen beinhaltet: |
| | |
| $result[result] | True/False (Erfolg der Aktion) |
| $result[add_onload] | JavaScript Code für das onLoad Event |
| $result[message] | Nachricht über den Erfolg der Aktion bzw. Fehlermeldung |
| $result[publication] | Name der Publikation bzw. Mandant in dem das Objekt existiert |
| $result[location] | absoluter Pfad im Filesystem (Position des Objektes) |
| $result[object] | Dateiname der Seite, Datei oder Komponente |
| $result[objectname] | Name der Seite, Datei oder Komponente |
| $result[objecttype] | Filetype bzw. File Extension der Datei |

### 3.5.7 cutobject

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

**Beschreibung:**
Ausschneiden eine bestehender Seite, Datei oder Komponente.

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

**Input-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $location | absoluter Pfad (Position des Objektes) |
| $object | Alter Name des Objektes |
| $user | Benutzername |

**globale Input-Parameter:**
Die folgenden globalen Input Parameter sind ebenfalls der Funktion zu übergeben:

| | |
|---|---|
| $lang | Spracheinstellung bzw. Sprachkürzel, z.B. "en", "de" |

**Output:**

| | |
|---|---|
| Array | Array $result das folgende Informationen beinhaltet: |
| | |
| $result[result] | True/False (Erfolg der Aktion) |
| $result[add_onload] | JavaScript Code für das onLoad Event |
| $result[message] | Nachricht über den Erfolg der Aktion bzw. Fehlermeldung |
| $result[object] | Dateiname der Seite, Datei oder Komponente |
| $result[objectname] | Name der Seite, Datei oder Komponente |
| $result[objecttype] | Filetype bzw. File Extension der Datei |
| $result[clipboard] | temporärer Eintrag im Clipboard (kann als globale Variable $clipboard der Funktion pasteobject übergeben werden, diese muss somit keinen Lesezugriff auf die temporäre Datei ausführen) |

### 3.5.8 copyobject

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

**Beschreibung:**
Kopieren eine bestehender Seite, Datei oder Komponente.

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

**Input-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $location | absoluter Pfad (Position des Objektes) |
| $object | Alter Name des Objektes |
| $user | Benutzername |

**globale Input-Parameter:**
Die folgenden globalen Input Parameter sind ebenfalls der Funktion zu übergeben:

| | |
|---|---|
| $lang | Spracheinstellung bzw. Sprachkürzel, z.B. "en", "de" |

**Output:**

| | |
|---|---|
| Array | Array $result das folgende Informationen beinhaltet: |
| | |
| $result[result] | True/False (Erfolg der Aktion) |
| $result[add_onload] | JavaScript Code für das onLoad Event |
| $result[message] | Nachricht über den Erfolg der Aktion bzw. Fehlermeldung |
| $result[object] | Dateiname der Seite, Datei oder Komponente |
| $result[objectname] | Name der Seite, Datei oder Komponente |
| $result[objecttype] | Filetype bzw. File Extension der Datei |
| $result[clipboard] | temporärer Eintrag im Clipboard (kann als globale Variable $clipboard der Funktion pasteobject übergeben werden, diese muss somit keinen Lesezugriff auf die temporäre Datei ausführen) |

### 3.5.9 copyconnectedobject

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

**Beschreibung:**
Kopieren eine bestehender Seite, Datei oder Komponente auf Basis des gleichen Content Containers.

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

**Input-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $location | absoluter Pfad (Position des Objektes) |
| $object | Alter Name des Objektes |
| $user | Benutzername |

**globale Input-Parameter:**
Die folgenden globalen Input Parameter sind ebenfalls der Funktion zu übergeben:

| | |
|---|---|
| $lang | Spracheinstellung bzw. Sprachkürzel, z.B. "en", "de" |

**Output:**

| | |
|---|---|
| Array | Array $result das folgende Informationen beinhaltet: |
| | |
| $result[result] | True/False (Erfolg der Aktion) |
| $result[add_onload] | JavaScript Code für das onLoad Event |
| $result[message] | Nachricht über den Erfolg der Aktion bzw. Fehlermeldung |
| $result[object] | Dateiname der Seite, Datei oder Komponente |
| $result[objectname] | Name der Seite, Datei oder Komponente |
| $result[objecttype] | Filetype bzw. File Extension der Datei |
| $result[clipboard] | temporärer Eintrag im Clipboard (kann als globale Variable $clipboard der Funktion pasteobject übergeben werden, diese muss somit keinen Lesezugriff auf die temporäre Datei ausführen) |

### 3.5.10 pasteobject

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

**Beschreibung:**
Einfügen einer bestehender Seite, Datei oder Komponente.

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

**Input-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $location | absoluter Pfad (Position des Objektes) |
| $user | Benutzername |
| $clipboard | globale Varibale mit dem temporären Eintrag im Clipboard (Damit ist ein Lesezugriff auf die temporäre Datei nicht notwendig.) |

**globale Input-Parameter:**
Die folgenden globalen Input Parameter sind ebenfalls der Funktion zu übergeben:

| | |
|---|---|
| $lang | Spracheinstellung bzw. Sprachkürzel, z.B. "en", "de" |

**Output:**

| | |
|---|---|
| Array | Array $result das folgende Informationen beinhaltet: |
| | |
| $result[result] | True/False (Erfolg der Aktion) |
| $result[add_onload] | JavaScript Code für das onLoad Event |
| $result[message] | Nachricht über den Erfolg der Aktion bzw. Fehlermeldung |
| $result[publication] | Name der Publikation bzw. Mandant in dem das Objekt existiert |
| $result[location] | absoluter Pfad im Filesystem (Position des Objektes) |
| $result[object] | Dateiname der Seite, Datei oder Komponente |
| $result[objectname] | Name der Seite, Datei oder Komponente |
| $result[objecttype] | Filetype bzw. File Extension der Datei |

### 3.5.11 lockobject

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

**Beschreibung:**
Sperren einer oder mehrerer bestehender Seiten oder Komponenten die auf den gleichen Content Container beruhen für die exklusive Nutzung eines Benutzers.

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

**Input-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $location | absoluter Pfad (Position des Objektes) |
| $object | Name des Objektes |
| $user | Benutzername |

**globale Input-Parameter:**
Die folgenden globalen Input Parameter sind ebenfalls der Funktion zu übergeben:

| | |
|---|---|
| $lang | Spracheinstellung bzw. Sprachkürzel, z.B. "en", "de" |

**Output:**

| | |
|---|---|
| Array | Array $result das folgende Informationen beinhaltet: |
| | |
| $result[result] | True/False (Erfolg der Aktion) |
| $result[add_onload] | JavaScript Code für das onLoad Event |
| $result[message] | Nachricht über den Erfolg der Aktion bzw. Fehlermeldung |
| $result[object] | Dateiname der Seite, Datei oder Komponente |
| $result[objectname] | Name der Seite, Datei oder Komponente |
| $result[objecttype] | Filetype bzw. File Extension der Datei |

### 3.5.12 unlockobject

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

**Beschreibung:**
Entsperren einer oder mehrerer bestehender Seiten oder Komponenten die auf den gleichen Content Container beruhen für die gemeinsame Nutzung durch alle Benutzer.

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

**Input-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $location | absoluter Pfad (Position des Objektes) |
| $object | Name des Objektes |
| $user | Benutzername |

**globale Input-Parameter:**
Die folgenden globalen Input Parameter sind ebenfalls der Funktion zu übergeben:

| | |
|---|---|
| $lang | Spracheinstellung bzw. Sprachkürzel, z.B. "en", "de" |

**Output:**

| | |
|---|---|
| Array | Array $result das folgende Informationen beinhaltet: |
| | |
| $result[result] | True/False (Erfolg der Aktion) |
| $result[add_onload] | JavaScript Code für das onLoad Event |
| $result[message] | Nachricht über den Erfolg der Aktion bzw. Fehlermeldung |
| $result[object] | Dateiname der Seite, Datei oder Komponente |
| $result[objectname] | Name der Seite, Datei oder Komponente |
| $result[objecttype] | Filetype bzw. File Extension der Datei |

### 3.5.13 publishobject

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

**Beschreibung:**
Publizieren einer Seite oder Komponente. Alle gebundenen Kopien des Objektes bzw. dessen Content Containers werden ebenfalls publiziert. Gestattet die Berechtigung eines im Einsatz befindlichen Workflows die Publizierung nicht, so wird das Objekt auch nicht publiziert.

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

**Input-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $location | absoluter Pfad (Position des Objektes) |
| $object | Name des Objektes |
| $user | Benutzername |

**globale Input-Parameter:**
Die folgenden globalen Input Parameter sind ebenfalls der Funktion zu übergeben:

| | |
|---|---|
| $lang | Spracheinstellung bzw. Sprachkürzel, z.B. "en", "de" |

**Output:**

| | |
|---|---|
| Array | Array $result das folgende Informationen beinhaltet: |
| | |
| $result[result] | True/False (Erfolg der Aktion) |
| $result[add_onload] | JavaScript Code für das onLoad Event |
| $result[message] | Nachricht über den Erfolg der Aktion bzw. Fehlermeldung |

### 3.5.14 unpublishobject

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

**Beschreibung:**
Entpublizieren einer Seite oder Komponente. Link und Task Management werden automatisch ausgeführt. Alle gebundenen Kopien des Objektes bzw. dessen Content Containers werden ebenfalls entpubliziert.

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

**Input-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $location | absoluter Pfad (Position des Objektes) |
| $object | Name des Objektes |
| $user | Benutzername |

**globale Input-Parameter:**
Die folgenden globalen Input Parameter sind ebenfalls der Funktion zu übergeben:

| | |
|---|---|
| $lang | Spracheinstellung bzw. Sprachkürzel, z.B. "en", "de" |

**Output:**

| | |
|---|---|
| Array | Array $result das folgende Informationen beinhaltet: |
| | |
| $result[result] | True/False (Erfolg der Aktion) |
| $result[add_onload] | JavaScript Code für das onLoad Event |
| $result[message] | Nachricht über den Erfolg der Aktion bzw. Fehlermeldung |

### 3.5.15 getlinkedobject

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

**Beschreibung:**
Diese Funktion extrahiert alle Objekte, die auf das gegebene Objekt zeigen. Dies können Seiten-Links oder auch Komponenten-Links sein. Ist das übergebene Objekt eine Seite, so werden alle Objekte ermittelt die einen Seiten-Link auf das Objekt besitzen. Ist das übergebene Objekt eine Komponente, so werden alle Objekte gefunden die einen Komponenten-Link zu dem Objekt besitzen.

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

**Input-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $location | absoluter Pfad (Position des Objektes) |
| $object | Name des Objektes |
| $cat | optional: Objekt Kategorie [page, comp] |

**Output:**

| | |
|---|---|
| Array | Array $result das folgende Informationen beinhaltet: |
| | |
| $result | False (Aktion fehlgeschlagen) |
| $result[publication] | Name der Publikation bzw. Mandant in dem das Objekt existiert |
| $result[location] | absoluter Pfad im Filesystem (Position des Objektes) |
| $result[object] | Name des Objektes |
| $result[category] | Kategorie des Objektes [page, comp] |

### 3.5.16 getconnectedobject

**Syntax:**
getconnectedobject ($site, $container)

**Beschreibung:**
Diese Funktion ermittelt alle Objekte, die auf dem gleichen Content Container basieren. Der Name des Content Containers eines Objektes kann mittels der Funktion "getfilename" ermittelt werden.

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

**Input-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $container | Name des Content Containers |

**Output:**

| | |
|---|---|
| Array | Array $result das folgende Informationen beinhaltet: |
| | |
| $result | False (Aktion fehlgeschlagen) |
| $result[publication] | Name der Publikation bzw. Mandant in dem das Objekt existiert |
| $result[location] | absoluter Pfad im Filesystem (Position des Objektes) |
| $result[object] | Name des Objektes |
| $result[category] | Kategorie des Objektes [page, comp] |

### 3.5.17 getobjectcontainer

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

**Beschreibung:**
Diese Funktion lädt den Content Container (XML-String) eines bestimmten Objektes. Das Objekt kann eine Seite, Datei, Komponente oder ein Ordner sein.
Die gewünschten Daten können mittels der Funktionen „getcontent“ oder „selectcontent“ aus dem XML-String ermittelt werden.

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

**Input-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $location | Pfad im Filesystem (Position des Objektes) |
| $object | Name des Objektes |
| $user | Benutzername |

**Output:**

| | |
|---|---|
| XML-String | Rückgabe des geladenen Content Containers |
| False | Fehler aufgetreten |

## 3.6 Bearbeiten von Inhalten

Die programmatische Bearbeitung der Inhalte eines Objektes wird anhand des folgenden Beispiels gezeigt. Zu beachten ist die Funktion settext, die dazu verwendet wird Texte eines Objektes zu manipulieren, für nähere Details siehe die Set API Funktionsbibliothek.

```
// 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 Bibliothek File Operation

Die folgenden Funktionen für File-Operationen sollten keinesfalls benutzt werden, um Objekte (Seiten, Komponenten oder Dateien) zu laden oder zu speichern.
Sie können diese Funktionen jedoch zum Laden und Speichern von XML-Content-Container verwenden, sollten Sie dies für die Entwicklung von Erweiterungen oder Anwendungen benötigen.

### 3.7.1 loadfile

**Syntax:**
loadfile ($abs_path, $filename)

**Beschreibung:**
Mit hilde dieser Funktion können Dateien geladen werden. Es müssen der Absolutpfad, als auch der Dateiname selbst als Parameter übergeben werden. Die Funktion wartet bis zu 3 Sekunden lang beim Laden von gesperrten Dateien. Wird der User-Parameter $user gesetzt, so kann die Funktion auch gesperrte Dateien des gegebenen Benutzers lesen.

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

**Input-Parameter:**

| | |
|---|---|
| $abs_path | absoluter Pfad zur gewünschten Datei, %page% und %comp% können in der Pfadangabe verwendet werden |
| $filename | Dateiname |

**Output:**

| | |
|---|---|
| Dateinhalt | Funktion wurde fehlerfrei ausgeführt und liefert den Inhalt der Datei |
| False | Fehler aufgetreten |

### 3.7.2 savefile

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

**Beschreibung:**
Mit savefile werden Dateien gespeichert. Hierfür müssen der Absolutpfad, der gewünschte Dateiname, als auch der Inhalt, die in die Datei geschrieben werden soll als Parameter übergeben werden. Ist die Datei gesperrt, so wird nicht gespeichert und False retourniert.

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

**Input-Parameter:**

| | |
|---|---|
| $abs_path | absoluter Pfad zur gewünschten Datei, %page% und %comp% können in der Pfadangabe verwendet werden |
| $filename | Dateiname |
| $filedata | Inhalt, der in die Datei geschrieben werden soll |

**Output:**

| | |
|---|---|
| True | Funktion wurde fehlerfrei ausgeführt |
| False | Fehler aufgetreten |

### 3.7.3 loadlockfile

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

**Beschreibung:**
Damit können Dateien wie mit loadfile geladen werden, es wird aber zusätzlich ein Sperr-Mechanismus ausgelöst.
Diese Funktion sollte nur dann genutzt werden, wenn Daten manipuliert und wieder gespeichert werden sollen. Damit wird sichergestellt, dass keine anderen Schreibzugriffe eines anderen Users erfolgen können. Beim Speichern muss die Funktion „savelockfile“ benutzt werden, um den Inhalt wieder freizugeben.
Es müssen der Benutzer, der absolute Pfad als auch der Dateiname selbst als Parameter übergeben werden.

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

**Input-Parameter:**

| | |
|---|---|
| $user | Benutzer der die Datei sperrt |
| $abs_path | absoluter Pfad zur gewünschten Datei, %page% und %comp% können in der Pfadangabe verwendet werden |
| $filename | Dateiname |

**Output:**

| | |
|---|---|
| Dateinhalt | Funktion wurde fehlerfrei ausgeführt und liefert den Inhalt der Datei |
| False | Fehler aufgetreten |

### 3.7.4 savelockfile

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

**Beschreibung:**
Mit savelockfile werden Dateien gespeichert und entsperrt, die vorher mit loadlockfile geöffnet wurden. Hierfür müssen der Benutzer, der Absolutpfad, der gewünschte Dateiname, als auch der Inhalt, die in die Datei geschrieben werden soll als Parameter übergeben werden.

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

**Input-Parameter:**

| | |
|---|---|
| $user | Benutzer der die Datei sperrt |
| $abs_path | absoluter Pfad zur gewünschten Datei, %page% und %comp% können in der Pfadangabe verwendet werden |
| $filename | Dateiname |
| $filedata | Inhalt, der in die Datei geschrieben werden soll |

**Output:**

| | |
|---|---|
| True | Funktion wurde fehlerfrei ausgeführt |
| False | Fehler aufgetreten |

### 3.7.5 lockfile

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

**Beschreibung:**
Mit lockfile werden Dateien von einem bestimmten Benutzer gesperrt und stehen für dessen exklusive Nutzung zur Verfügung. Hierfür müssen der Benutzer, der Absolutpfad, der gewünschte Dateiname als Parameter übergeben werden.

Bsp:
lockfile ("Miller", "%page%/myPublication/home/", "index.php");

**Input-Parameter:**

| | |
|---|---|
| $user | Benutzer der die Datei sperrt |
| $abs_path | absoluter Pfad zur gewünschten Datei, %page% und %comp% können in der Pfadangabe verwendet werden |
| $filename | Dateiname |

**Output:**

| | |
|---|---|
| True | Funktion wurde fehlerfrei ausgeführt |
| False | Fehler aufgetreten |

### 3.7.6 unlockfile

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

**Beschreibung:**
Mit unlockfile werden Dateien entsperrt, die vorher mit lockfile gesperrt oder mit loadlockfile geöffnet wurden. Hierfür müssen der Benutzer, der Absolutpfad, der gewünschte Dateiname als Parameter übergeben werden.

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

**Input-Parameter:**

| | |
|---|---|
| $user | Benutzer der die Datei sperrt |
| $abs_path | absoluter Pfad zur gewünschten Datei, %page% und %comp% können in der Pfadangabe verwendet werden |
| $filename | Dateiname |

**Output:**

| | |
|---|---|
| True | Funktion wurde fehlerfrei ausgeführt |
| False | Fehler aufgetreten |

### 3.7.7 deletefile

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

**Beschreibung:**
Mit deletefile können Dateien und (leere) Ordner gelöscht werden. Es wird der Pfad der gewünschten Datei übergeben, der Dateiname, und ein Parameter "Rekursiv", der entweder (0) oder (1) beträgt. Wenn recursive 1 gesetzt wurde, wird der gesamte Inhalt des Ordners behandelt, also auch Unterverzeichnisse und deren Dateien, bei 0 werden nur die Dateien des angesprochenen Ordners (falls leer) entfernt.

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

**Input-Parameter:**

| | |
|---|---|
| $abs_path | absoluter Pfad zur gewünschten Datei, %page% und %comp% können in der Pfadangabe verwendet werden |
| $file | Dateiname |
| $recursive | 0 oder 1, je nachdem ob sich der Vorgang auch auf Unterverzeichnisse auswirken soll |

**Output:**

| | |
|---|---|
| True | Funktion wurde fehlerfrei ausgeführt |
| False | Fehler aufgetreten |

### 3.7.8 appendfile

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

**Beschreibung:**
Mit appendfile können Inhalte an Dateien angefügt werden. Die Funktion arbeitet wie savefile, der Unterschied besteht allerdings darin, dass bereits vorhandene Daten nicht überschrieben, sondern ergänzt werden. Hierfür müssen der Absolutpfad, der gewünschte Dateiname, als auch der Inhalt, die in die Datei geschrieben werden soll als Parameter übergeben werden.

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

**Input-Parameter:**

| | |
|---|---|
| $abs_path | absoluter Pfad zur gewünschten Datei, %page% und %comp% können in der Pfadangabe verwendet werden |
| $filename | Dateiname |
| $filedata | Inhalt, der an die Datei angefügt werden soll |

**Output:**

| | |
|---|---|
| True | Funktion wurde fehlerfrei ausgeführt |
| False | Fehler aufgetreten |

## 3.8 XML Bibliothek

Die folgenden Funktionen bieten Ihnen die Möglichkeit Inhalte aus XML-Content-Container zu lesen und zu schreiben. Sie können optional auch mit anderen Technologien, die mit XML umgehen können, die Inhalte der Container abfragen. Die Bibliothek Edit Content bietet Ihnen jedoch eine sehr einfache als auch performante Methode hierfür.

### 3.8.1 setxmlparameter

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

**Beschreibung:**
Setzt den Wert eines bestimmten Parameters innerhalb der XML-Deklaration (1.Zeile).

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

**Input-Parameter:**

| | |
|---|---|
| $xmldata | XML-String der übergeben und manipuliert werden soll |
| $parameter | Name des Parameter dessen Wert geändert werden soll |
| $value | Wert des Paramaters |

**Output:**

| | |
|---|---|
| XML-String | Rückgabe des manipulierten XML-Strings |
| False | Fehler aufgetreten |

### 3.8.2 getcontent

**Syntax:**
getcontent ($xmldata, $tag)

**Beschreibung:**
Holt den XML-Content aus dem Content Container, der sich innerhalb der Tags $tag befindet. Ein Array mit allen gefundenen Inhalten bzw. Childs wird zurückgegeben und kann in einer Array-Variable gespeichert und weiterverwendet werden.

Bsp:
```
// hole alle text-childs aus Content Container
$text_array = getcontent ($xmldata, "<text>");

// ausgeben aller Text-Childs
foreach ($text_array as $text) echo $text;
```

**Input-Parameter:**

| | |
|---|---|
| $xmldata | XML-String der übergeben und durchsucht werden soll |
| $tag | XML-Tag der die Information bzw. Childs umschliesst |

**Output:**

| | |
|---|---|
| Array | Array mit allen gefunden Inhalten, der erste Wert/Inhalt kann mit Array[0] angesprochen werden |
| False | Fehler aufgetreten |

### 3.8.3 getxmlcontent

**Syntax:**
getxmlcontent ($xmldata, $tag)

**Beschreibung:**
Holt den XML-Content aus dem Content Container, der sich innerhalb der Tags $tag befindet und belässt im Unterschied zu getcontent die Tags im Rückgabewert (Array). Ein gesamter Node (well-formed) wird daher zurückgeliefert.
Ein Array mit allen gefundenen Inhalten bzw. Childs wird zurückgegeben und kann in einer Variable vom Typ Array gespeichert und weiterverwendet werden.

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

**Input-Parameter:**

| | |
|---|---|
| $xmldata | XML-String der übergeben und durchsucht werden soll |
| $tag | XML-Tag der die Information bzw. Childs umschliesst |

**Output:**

| | |
|---|---|
| Array | Array mit allen gefunden Inhalten, der erste Wert/Inhalt kann mit Array[0] angesprochen werden |
| False | Fehler aufgetreten |

### 3.8.4 selectcontent

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

**Beschreibung:**
Holt jenen XML-Content bestimmt durch $parenttag aus dem Content-Container, der innerhalb des Childtags $childtag einen bestimmten Wert $childvalue aufweist.
Ein Array mit allen gefundenen Inhalten wird zurückgegeben und kann in einer Array-Variable gespeichert und weiterverwendet werden.

Bsp:
Auszug aus dem Content Container:
```
…..
<text>
  <text_id>summary</text_id>
  <textuser>editor1</textuser>
  <textcontent>This is my summary!</textcontent>
</text>
…..

// hole alle Text-Childs mit der id=summary
$text_array = selectcontent ($xmldata, "<text>", "<text_id>", "summary");

// extrahiere das Summary aus dem gefundenen Inhalt
foreach ($text_array as $text)
{
  $summary = getcontent ($text, "<textcontent>");
}
```

**Input-Parameter:**

| | |
|---|---|
| $xmldata | XML-String der übergeben und durchsucht werden soll |
| $parenttag | XML-Tag der die Information bzw. das Child beinhaltet |
| $childtag | optional: XML-Tag der die Information umschliesst, die einen gewissen Wert besitzen muss |
| $childvalue | optional: Wert der Bedingung, das WildCard Zeichen * kann am Anfang und/oder am Ende des Ausdruckes verwendet werden und ist Platzhalter für beliebige weitere Zeichen. |

**Output:**

| | |
|---|---|
| Array | Array mit allen gefunden Inhalten, der erste Wert/Inhalt kann mit Array[0] angesprochen werden |
| False | Fehler aufgetreten |

### 3.8.5 selectxmlcontent

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

**Beschreibung:**
Holt jenen XML-Content definiert durch $parenttag aus dem Content-Container, der innerhalb eines Childtags $childtag einen bestimmten Wert $childvalue aufweist. Im Unterschied zu getcontent werden die Parent-Tags im Rückgabewert (Array) belassen.
Ein Array mit allen gefundenen Inhalten wird zurückgegeben und kann in einer Array-Variable gespeichert und weiterverwendet werden.

Bsp:
Auszug aus dem Content Container:

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

// hole alle Text-Childs mit der id=summary
$text_array = selectxmlcontent ($xmldata, "<text>", "<text_id>", "summary");

// extrahiere das Summary aus dem gefundenen Inhalt
foreach ($text_array as $text)
{
  $summary = getcontent ($text, "<textcontent>");
}
```

**Input-Parameter:**

| | |
|---|---|
| $xmldata | XML-String der übergeben und durchsucht werden soll |
| $parenttag | XML-Tag der die Information bzw. das Child beinhaltet |
| $childtag | optional: XML-Tag der die Information umschliesst, die einen gewissen Wert besitzen muss |
| $childvalue | optional: Wert der Bedingung, das WildCard Zeichen * kann am Anfang und/oder am Ende des Ausdruckes verwendet werden und ist Platzhalter für beliebige weitere Zeichen. |

**Output:**

| | |
|---|---|
| Array | Array mit allen gefunden Inhalten, der erste Wert/Inhalt kann mit Array[0] angesprochen werden |
| False | Fehler aufgetreten |

### 3.8.6 deletecontent

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

**Beschreibung:**
Löscht den gesamten XML-Content definiert durch den Tag $tagname. Als Kriterium für die Auswahl der zu löschenden Childs wird das entsprechende XML-Childtag $condtag und die umschlossene Information $condvalue als Bedingung mitgeschickt.

Bsp:
Auszug aus dem Content Container:

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

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

**Input-Parameter:**

| | |
|---|---|
| $xmldata | XML-String der übergeben wird |
| $parenttag | XML-Tag der die Information bzw. Childs umschliesst, die aus dem Content Container entfernt werden sollen |
| $condtag | optional: Name des Parameters (XML-Child) das der Bedingung unterliegt |
| $condvalue | optional: Wert der Bedingung, die erfüllt werden muss |

**Output:**

| | |
|---|---|
| XML-String | Rückgabe des manipulierten XML-Strings |
| False | Fehler aufgetreten |

### 3.8.7 setcontent

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

**Beschreibung:**
Ein XML-String wird übergeben und innerhalb eines bestimmten Parent Nodes ($parenttagname) wird überprüft, ob ein bestimmter Parameter ($condtag) existiert und einen bestimmter Wert ($condvalue) aufweist. Ist die Bedingung erfüllt, wird der Wert des Parameters $tagname durch einen neuen Wert $contentnew ersetzt.

Bsp:
Auszug aus dem Content Container:

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

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

**Input-Parameter:**

| | |
|---|---|
| $xmldata | XML-String der übergeben und modifiziert werden soll |
| $parenttagname | optional: XML-Parenttag |
| $tagname | optional: XML-Childtag, dessen Wert ersetzt werden soll (wenn Bedingung erfüllt) |
| $contentnew | Neuer Wert für den XML-Childtag $tagmame |
| $condtag | optional: Name des Parameters der die Bedingung erfüllen muss |
| $condvalue | optional: Wert des Parameters für die Bedingung |

**Output:**

| | |
|---|---|
| XML-String | Rückgabe des manipulierten XML-Strings |
| False | Fehler aufgetreten |

### 3.8.8 updatecontent

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

**Beschreibung:**
Alle XML-String $xmlnode wird durch einen neuen String $xmlnodenew in $xmldata ersetzt. Diese Methode ist schneller als setcontent, wenn der aktualisierende XML Node bereits aus dem Container extrahiert wurde.

Bsp:
Auszug aus dem Content Container:

```
…..
<text>
  <text_id>bedingung</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-Parameter:**

| | |
|---|---|
| $xmldata | XML-String der übergeben und modifiziert werden soll |
| $xmlnode | zu ersetzender XML-String (Node bzw. Substring von $xmldata) |
| $xmlnodenew | optional: neuer XML-String, wenn leer, so wird der bestehende XML-String entfernt. |

**Output:**

| | |
|---|---|
| XML-String | Rückgabe des manipulierten XML-Strings |
| False | Fehler aufgetreten |

### 3.8.9 insertcontent

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

**Beschreibung:**
Fügt einen XML-String (Child Node) vor dem Ende des übergebenen XML-Parenttags ein. Der modifizierte XML-String wird zurückgegeben.

Bsp:
Auszug aus dem Content Container:

```
…..
  <articletextlist>
    <text>
      <text_id>art1:summary</text_id>
      <textuser>editor1</textuser>
      <textcontent>This is my summary!</textcontent>
    </text>
---------- hier wird ein Child Node eingefügt ------------
    <text>
      <text_id>art1:longtext</text_id>
      <textuser>editor1</textuser>
      <textcontent>This is my summary!</textcontent>
    </text>
-------------------------------------------------------------
  </articletextlist>
…..

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

**Input-Parameter:**

| | |
|---|---|
| $xmldata | XML-String der übergeben und modifiziert werden soll |
| $insertxmldata | XML-String der eingesetzt wird |
| $tagname | optional: XML-Parenttag an dessen Ende eingesetzt werden soll |

**Output:**

| | |
|---|---|
| XML-String | Rückgabe des manipulierten XML-Strings |
| False | Fehler aufgetreten |

### 3.8.10 addcontent

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

**Beschreibung:**
Innerhalb eines Parent Nodes wird ein Child Node hinzugefügt, sofern ein Wert im darüberliegenden Grandparent Node die Bedingung erfüllt. Im Child Node kann auf Wunsch gleichzeitig ein Wert gesetzt werden. Der modifizierte XML-String wird zurückgegeben.

Bsp:
Auszug aus dem 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>
---------- hier wird ein Child Node eingefügt ------------
    <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-Parameter:**

| | |
|---|---|
| $xmldata | XML-String der übergeben und modifiziert werden soll |
| $sub_xmldata | XML-String der eingebettet werden soll |
| $grandtagname | Enthält den XML-Childtag, in dem $sub_xmldata eingebettet werden soll |
| $condtag | optional: Name des Parameters der überprüft werden soll |
| $condvalue | optional: Wert des Parameters der überprüft werden soll |
| $parenttagname | optional: XML-Childtag, in dem $sub_xmldata eingebettet werden soll |
| $tagname | optional: Childtag des eingebetteten XML-String |
| $contentnew | optional: Content für den Tag $tagname |

**Output:**

| | |
|---|---|
| XML-String | Rückgabe des manipulierten XML-Strings |
| False | Fehler aufgetreten |

## 3.9 Bibliothek Meta Data Generator

Diese Funktionsbibliothek ermöglicht Ihnen Keyword-Listen, die Description aus einem Inhalt zu erzeugen. Dies kann zur automatischen Erzeugung bzw. Befüllung von Metadaten verwendet werden.
Es können Meta Daten aus Multimedia-Dateien ausgelesen und im Container eines Objektes gespeichert werden.

### 3.9.1 getmetakeywords

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

**Beschreibung:**
Der Funktion wird der Inhalt übergeben. Damit werden alle Keywords aus dem Text ermittelt und als Keyword-Liste zurückgegeben.

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

**Input-Parameter:**

| | |
|---|---|
| $text | Content als String |
| $language | optional: Sprache [en, de], Standard ist "en" |
| $charset | optional: Character Set, Standard ist "UTF-8" |

**Output:**

| | |
|---|---|
| Keywords | Komma-getrennte Liste aller Keywords |
| False | Fehler aufgetreten |

### 3.9.2 getmetadescription

**Syntax:**
getdescription ($text, $charset)

**Beschreibung:**
Dieser Funktion wird der Inhalt übergeben. Daraufhin wird eine Kurzbeschreibung aus dem Text ermittelt und zurückgegeben.

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

**Input-Parameter:**

| | |
|---|---|
| $text | Content als String |
| $charset | optional: Character Set, Standard ist "UTF-8" |

**Output:**

| | |
|---|---|
| Keywords | Kurzbeschreibung des Inhaltes |
| False | Fehler aufgetreten |

## 3.10 Bibliothek Notifications

Diese Funktionsbibliothek versendet automatisierte Nachrichten an einen Benutzer anhand von Grenzwerten eines bestimmten Feldes.
Der Benutzer erhält eine vorformatierte Nachricht mit Information (Links) zu allen Objekten, die in den Suchbereich (Datumsober und -untergrenze) fallen.

### 3.10.1 licensenotification

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

**Beschreibung:**
Der Funktion ermittelt alle Objekte aufgrund des vorgegebenen Suchbereiches (Lokation und Datumsgrenzwerte) und versendet eine E-Mail an einen bestimmten Benutzer mit den Links zu allen betroffenen Objekten.

Bsp:

```
// 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-Parameter:**

| | |
|---|---|
| $site | Name der Publikation |
| $cat | Objekt Kategorie [page, comp] |
| $folderpath | Pfad für die Defintion des Suchbereiches |
| $text_id | Text ID des Feldes auf das die Suche angewendet werden soll |
| $date_begin | Startdatum für die Suche (YYYY-MM-DD) |
| $date_end | Endedatum für die Suche (YYYY-MM-DD) |
| $user | Benutzername |

**Output:**

| | |
|---|---|
| True | Mail wurde erfolgreich gesendet |
| False | Fehler aufgetreten |

# 4 Komponenten und Applikationen

Wenn Anwendungen in Komponenten integriert werden und Variablen aus einer Seite an eine Komponente übergeben werden müssen, so ist auf folgendes zu achten:
Die Komponenten müssen über das Dateisystem eingebunden werden (nicht via HTTP).
Alle Variablen die an die Komponente zu übergeben sind, sind in der Komponente als global zu definieren.

Bsp:
Eine Seite übergibt eine Variable an eine Komponente.

Hier der Code der Seite:

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

Der Code der Komponente muss wie folgt aussehen:

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

Im Beispiel wird die Variable $test bzw. dessen Wert "This is just a test!" von der Komponente übernommen und in der Präsentation angezeigt.

# 5 Database Connectivity

Die Database Connectivity des hyper Content & Digital Asset Management Servers erlaubt die Anbindung von diversen Datenbanken zur Speicherung und Entnahme von Inhalten. Damit können z.B. relationale Datenbanken als externes Content Repository genutzt werden.
Zu diesem Zweck ist je Template der entsprechende hyperCMS-Tag für die Database Connectivity einzufügen, der auf ein DB-Connect File verweißt.
In diesem File werden Funktionen hinterlegt, die hyperCMS aufruft, sofern das Template auf die Funktionsdatei zeigt.
Die Inhalte werden aus der Datenbank gelesen und dem Redakteur angezeigt. Verändert der Redakteur die Inhalte, so können diese auch wieder in die Datenbank geschrieben werden. Es können für Lese- und Schreibzugriffe auch verschiedene Datenbanken aufgerufen werden. Die Funktionen im DB-Connect File bieten nur die Hülle bzw. standardisierte Schnittstelle zu hyperCMS, die durch den Programmierer befüllt werden muss.

Das Thema der Datenbankintegration ist komplex und individuell zu behandeln, da auch bereits bestehende Datenbanken und deren Informationen integriert werden können. hyperCMS gibt kein ER-Modell vor bzw. legt sich auf keine speziellen Datenbank-Produkte fest. Generell kann gesagt werden, dass alle Möglichkeiten von PHP ausgeschöpft werden können, um sich zu diversen Datenquellen zu verbinden.

Neben den notwendigen Parameter für Queries auf relationale Datenbanken wird auch der gesamte Content Conatiner als XML-String übergeben. Damit könnten Dokumente bzw. Inhalte aus den Content Repository auch als Node in XML-Datenbanken abgelegt werden.

Sie selbst bestimmen, wohin Sie Ihre Daten speichern bzw. woher Sie diese holen. Mit PHP besitzen eine mächtige Sprache, die Ihnen Zugriff auf alle gängigen Datenbanken bietet.

Mehr Information zu den Funktionen von PHP finden Sie unter: http://www.php.net

## 5.1 Erstellen einer Database Connectivity

Möchten Sie eine Database Connectivity erstellen, so erstellen Sie eine Kopie des Files db_connect_default.php, dieses finden Sie in dem gewählten Root-Verzeichnis für die Ablage der Management Daten unter dem folgenden weiterführenden Pfad: /data/db_connect/
Die Kopie des Files nennen Sie z.B. nach der Datenbank, die Sie anbinden möchten.

Danach öffnen Sie die Datei und erhalten Einsicht in die Funktionen. Im Source Code finden Sie auch eine Beschreibung der Funktionen und der übergebenen Parameter als auch des Outputs.

Exemplarisch soll hier ein Lesezugriff auf eine MySQL Datenbank für einen Text-Inhalt dargestellt werden. Wir gehen davon aus, dass in einem Table "TextContent" die Inhalte mit dem Primary Key "container_id" und "text_id", dem Text-Inhalt "Text" sowie dem Text-Typ "Type" vorliegen. Der User sowie die Artikel ID wird nicht gesondert gespeichert, dies ist für die Eindeutigkeit des Inhalts auch nicht notwendig, denn die ID des Content Containers als auch die ID des Elements reichen als Primärschlüssel aus.

```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

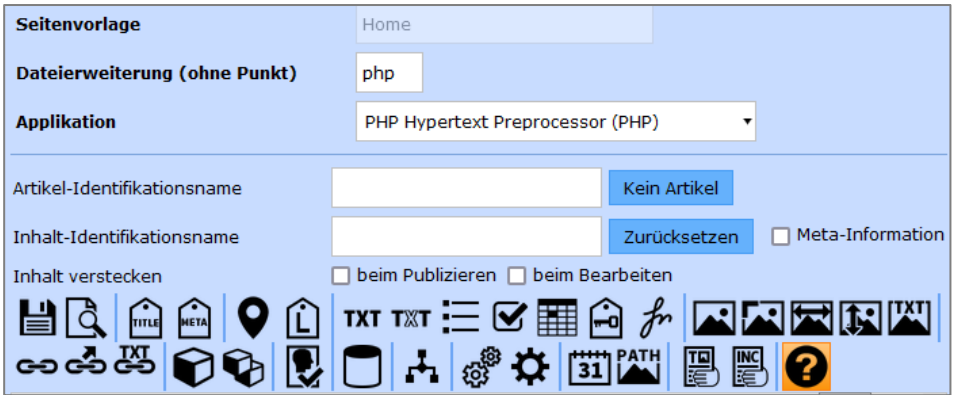
Der hyper Content & Digital Asset Management Server beinhaltet ein Event System, das eine automatisierte Ausführung von Aktionen passierend auf Ereignissen im System ermöglicht. Damit lassen sich z.B. manuelle Vorgänge automatisieren.
Events werden meist durch den Benutzer durch Wahl einer Aktion gestartet, z.B. das Publizieren einer Seite. Ist der entsprechende Event aktiviert, so wird nach erfolgreicher Ausführung des Publikationsprozesses der Seite das Event "onpublishobject" aufgerufen. Die darin definierten Funktionen werden sodann ausgeführt.
Die Events des Event Systems können in der Datei "hypercms_eventsys.inc.php" definiert werden. Diese befindet sich im internen Repository im Ordner "eventsystem". In dieser Datei befinden sich auch weitere wichtige Hinweise, die bei der Ausführung von Events zu beachten sind.
Das Event System ist innerhalb des gesamten Management Systems über alle Publikationen gültig. Das System ist Bestandteil des hyperCMS APIs und wird somit bei jedem Aufruf einer Funktion des APIs ausgeführt.
Events lassen sich in der Datei "hypercms_eventsys.inc.php" aktivieren als auch deaktivieren, sodass der Einsatz der darin definierten Events leicht gesteuert werden kann.
Bei allen Events wird zwischen PRE- und POST-Events unterschieden. Das PRE-Event wird vor der eigentlichen Ausführung der aufgerufenen Aktion gestartet, während das POST-Event nach der erfolgreichen Ausführung der Aktion aufgerufen wird.

Bsp:
Beim Publizieren eines Objektes soll automatisch auch die Seite "index.php" die sich an der gleichen Position befindet publiziert werden, da diese z.B. ein über ein hyperCMS Script generiertes Verzeichnis aller Objekte des gleichen Ordners beinhaltet.

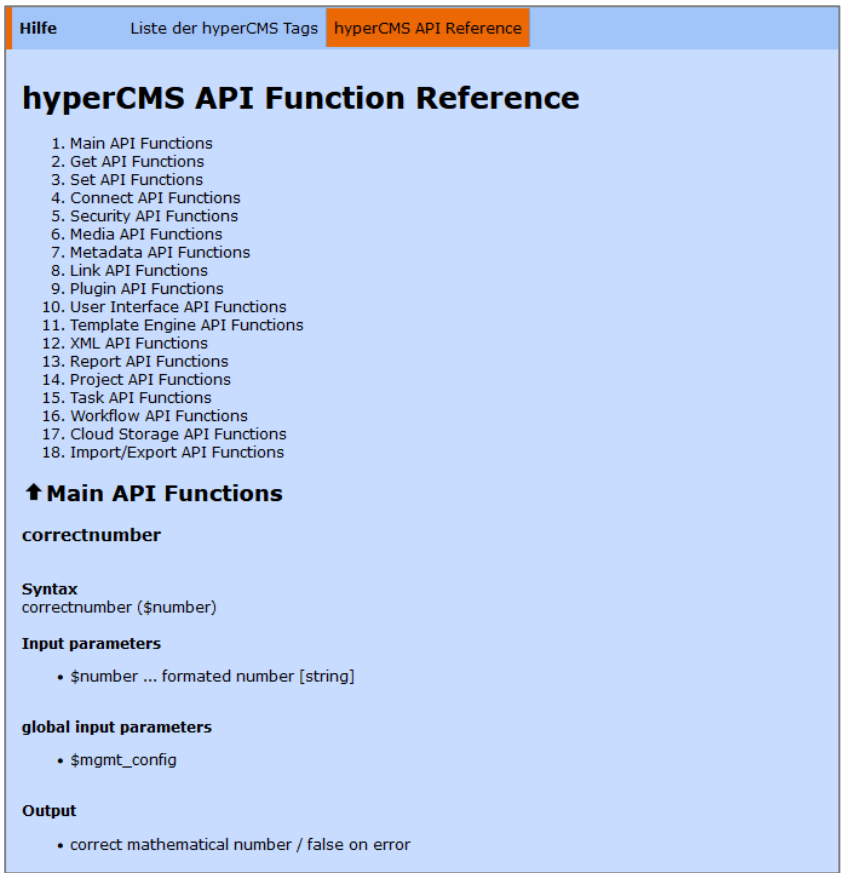
```
// ------------------------------------ on publish object POST event ------------------------------------
function onpublishobject_post ($site, $cat, $location, $object, $user)
{
  // load configuration
  include_once ("config.inc.php");

  // 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 Dokumentation aller hyperCMS API-Funktionen

Die vollständige Dokumentation aller API-Funktionen ist auf unserer Website hypercms.com in der aktuellen Version verfügbar.
Sie können die Dokumentation ihres Systems als Hilfe im Browser anzeigen. Nutzen Sie hierzu das ?-Icon im Vorlageneditor um die Referenz aller hyperCMS Tags und API Funktionen einzusehen.




Die Hilfe öffnet sich danach in einem neuen Popup-Fenster. Klicken Sie auf den Reiter „hyperCMs API Reference“ um die API Funktionen einzusehen. Mit Hilfe der Tastenkombination Strg + F können Sie die Suche aufrufen.



# 8 Rechtliche Hinweise / Impressum

## 8.1 Fragen und Anregungen

Sollten Sie weitergehende Fragen oder Anregungen zum Produkt haben, so wenden Sie sich bitte an den Support.

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

## 8.2 Impressum

Verantwortlich für den Inhalt:

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

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

## 8.3 Rechtliche Hinweise

Vorliegendes Benutzerhandbuch basiert auf der zum Zeitpunkt der Verfassung des Dokumentes verfügbaren Programmversion.

Der Hersteller behält sich Programmänderungen und –Verbesserungen vor.

Fehler und Irrtümer vorbehalten.