Infopark CMS Fiona

Tcl Interface Referenz

Infopark CMS Fiona

Tcl Interface Referenz

Inhalt

1 Vorbemerkungen ... 9

2 Mit der Tcl-Schnittstelle arbeiten ... 10

2.1 Die Tcl-Kommandoschnittstelle aufrufen ... 10

2.2 Tastaturbefehle ... 11

2.3 Eigene Tcl-Prozeduren verwenden ... 12

2.4 Onlinehilfe aufrufen ... 12

2.5 Das Encoding anpassen ... 13

2.6 Der abgesicherte Tcl-Interpreter ... 13

3 Über die Befehlsbeschreibungen ... 15

3.1 Struktur der Befehlsbeschreibungen ... 15

3.2 Datentypen und Notation ... 15

3.3 Datumsformat ... 16

3.4 Anforderungen an Bezeichner ... 16

4 Die Tcl-Befehle des Content Managers und der Template Engine ... 17

4.1 attribute (Versionsfelder) ... 17

4.1.1 Feldparameter ... 17

4.1.2 Eingabefeldparameter ... 18

4.1.3 Feldbefehle ... 19

4.2 attributeGroup (Feldergruppen) ... 26

4.2.1 AttributeGroup-Parameter ... 26

4.2.2 AttributeGroup-Befehle ... 26

4.3 channel (News-Listen) ... 32

4.3.1 Channel-Parameter ... 32

4.3.2 Channel-Befehle ... 32

4.4 content (Versionen) ... 36

4.4.1 Versionsfelder ... 36

4.4.2 Versionsbefehle ... 41

4.5 group (Benutzergruppen) ... 48

4.5.1 Benutzergruppen-Parameter ... 48

4.5.2 Benutzergruppenbefehle ... 49

4.6 incrExport (Inkrementeller Export) ... 55

4.6.1 IncrExport-Parameter ... 56

4.6.2 IncrExport-Befehle 57
4.7 job (Jobs) 59
4.7.1 Job-Parameter 60
4.7.2 Job-Befehle 62
4.8 licenseManager (Lizenzinformationen) 68
4.8.1 LicenseManager-Parameter 68
4.8.2 LicenseManager-Befehle 69
4.9 link (Verweise) 70
4.9.1 Link-Parameter 70
4.9.2 Link-Befehle 72
4.10 linkChecker (Link-Überprüfung) 75
4.10.1 Link-Checker-Parameter 75
4.10.2 Link-Checker-Befehle 75
4.11 logEntry (Protokoll-Einträge) 77
4.11.1 LogEntry-Parameter 77
4.11.2 Log-Typen 78
4.11.3 LogEntry-Befehle 78
4.12 obj (Dateien) 79
4.12.1 Dateiparameter 79
4.12.2 Dateibefehle 84
4.13 objClass (Dateivorlagen) 114
4.13.1 Dateivorlagenparameter 114
4.13.2 Dateivorlagenbefehle 118
4.14 reminder (Wiedervorlage) 123
4.14.1 reminder define 123
4.14.2 reminder delete 124
4.14.3 reminder where 124
4.15 statistics (Statistische Informationen) 125
4.15.1 Statistikparameter 125
4.15.2 Statistikbefehle 125
4.16 systemConfig (Systemkonfiguration) 127
4.16.1 Das Property-List-Format 127
4.16.2 Systemkonfigurationsbefehle 128
4.17 task (Aufgaben) 136
4.17.1 Task-Parameter 136

4.17.2 Task-Befehle 137
4.18 user (Benutzer) 140
4.18.1 Benutzerparameter 140
4.18.2 Benutzerbefehle 141
4.19 userAttribute (Benutzerfelder) 149
4.19.1 Benutzerfeldparameter 149
4.19.2 Benutzerfeldbefehle 149
4.20 userConfig (persönliche Einstellungen) 154
4.20.1 userConfig 154
4.21 userConfigForUser (persönliche Einstellungen anderer Benutzer) 155
4.21.1 userConfigForUser 155
4.22 workflow (Workflows) 156
4.22.1 Workflow-Parameter 156
4.22.2 Workflow-Befehle 156
4.23 Callback-Funktionen 160
4.23.1 exportFillCallback 160
4.23.2 exportSwitchCallback 161
4.23.3 exportSyncCallback 161
4.23.4 importCallback 161
4.24 Andere administrative Anweisungen 163
4.24.1 app closeDbConnection 163
4.24.2 app export 163
4.24.3 app get 164
4.24.4 app makeDbConnection 165
4.24.5 app publish 165
4.24.6 clearUsermanCache 166
4.24.7 decodeData 166
4.24.8 decodeFile 167
4.24.9 decodeToString 167
4.24.10 encodeData 167
4.24.11 encodeFile 168
4.24.12 filterTags 168
4.24.13 getRegisteredCommands 169
4.24.14 indexAllObjects 169
4.24.15 listTasks 169

4.24.16 listTasksOfGroup 170
4.24.17 listTasksOfUser 170
4.24.18 loadFile 171
4.24.19 loadTextFile 171
4.24.20 logMessage 172
4.24.21 logWarning 172
4.24.22 logout 173
4.24.23 news where 173
4.24.24 now 174
4.24.25 stream downloadBase64 174
4.24.26 stream downloadFile 175
4.24.27 stream uploadBase64 175
4.24.28 stream uploadFile 176
4.24.29 sudo 176
4.24.30 today 177
4.24.31 valueForLocalizerKey 177
4.24.32 whoami 178
4.24.33 writeFile 178
4.24.34 writeTextFile 179
4.25 Zusätzliche Tcl-Prozeduren 179
4.25.1 contentService 179
4.25.2 cp 180
4.25.3 findObjectId 181
4.25.4 grep 181
4.25.5 incrNpsDate 182
4.25.6 interactiveLoadSubtree 182
4.25.7 l 182
4.25.8 listObjectsWithoutSuperLinks 183
4.25.9 listSubtree 184
4.25.10 loadSubtree 184
4.25.11 ls 185
4.25.12 mkpubs 185
4.25.13 modifyPermissions 186
4.25.14 objWherePath 187
4.25.15 performWithEditedContentOfObjects 187

4.25.16 performWithObjects . . . 188
4.25.17 removeBlobAndFreeLinks . . . 188
4.25.18 removeSubtree . . . 189
4.25.19 renameObjClass . . . 190
4.25.20 rm . . . 190
4.25.21 showLinksOfObjects . . . 191
4.25.22 showPermissions . . . 192
4.25.23 streamFromServer . . . 192
4.25.24 streamToServer . . . 193

5 Mit Tcl Standardaufgaben lösen . . . 194
5.1 Vorlagen kopieren . . . 194
5.2 Teilhierarchien kopieren . . . 195
5.3 Alte Versionen und Protokolleinträge löschen . . . 196
5.4 Dateien finden . . . 197
5.5 Dateien verschieben . . . 198
5.6 Tcl-Skripte in mehreren Callback-Funktionen verwenden . . . 198
5.7 Datumsformate umwandeln . . . 199

6 Funktionsweise der Template Engine . . . 201
6.1 Import und Export . . . 201
6.2 Die Verzeichnishierarchien . . . 203
6.3 Abhängigkeiten . . . 204
6.4 Den Exportvorgang beschleunigen . . . 206

7 Fehlermeldungen . . . 208

1 Vorbemerkungen

Dieses Dokument beschreibt den Befehlsumfang der Tcl-Kommandoschnittstellen des Content Management Servers und der Template Engine. Das Dokument wendet sich an CMS- und Systemadministratoren, die mit der Skriptsprache Tcl und der Funktionalität von Infopark CMS Fiona vertraut sind.

Die Tcl-Syntax und der Funktionsumfang des Content Managers und der Template Engine werden hier deshalb nicht erläutert. Eine ausführliche Beschreibung von Tcl finden Sie in dem Buch „Tcl and the Tk Toolkit" (John K. Ousterhout, Addison-Wesley Publishing Company, 1994) sowie auf vielen Websites, unter anderem auf:

- Tcl-Ressourcen
- Tcl Developer Xchange

Die englischen CMS-spezifischen oder Internet-spezifischen Begriffe und Akronyme wie "Body", "Content" oder "Link" sind in diesem Dokument nicht ins Deutsche übersetzt, da auch die (Tcl-eigenen und die CMS-spezifischen) Tcl-Kommandos englische Wörter oder von englischen Wörtern abgeleitete Kunstwörter sind.

2

2 Mit der Tcl-Schnittstelle arbeiten

Der Content Management Serverund die Template Engine verfügen jeweils über eine Tcl-Kommandoschnittstelle, über die sich administrative Tätigkeiten (und beim Content Manager auch redaktionelle Arbeiten) schnell und effizient erledigen lassen.

2.1 Die Tcl-Kommandoschnittstelle aufrufen

Auf die Tcl-Kommandoschnittstelle einer CMS-Fiona-Applikation (CM, TE oder SES) können Sie mit Hilfe einer Tcl-Shell zugreifen. Eine Tcl-Shell kann auf jedem Computer installiert werden, nicht nur auf dem Server, auf dem CMS-Fiona läuft.

Bei einer CMS-Fiona-Standardinstallation steht Ihnen ein Skript zur Verfügung, das die mitgelieferte serverseitige Tcl-Shell aufruft und die Kommandos ausführt, die erforderlich sind, um sich mit dem Tcl-Server zu verbinden.

Voraussetzung für die erfolgreiche Verbindung mit dem Tcl-Server ist, dass die jeweilige Applikation läuft. Voreingestellt können Sie sich am Content Manager mit den Anmeldedaten anmelden, die nach der Installation von CMS-Fiona auf der Wilkommensseite standen. Die Template Engine und der Search Server erfordern keine Anmeldung.

Gehen Sie folgendermaßen vor, um die Shell zu starten:

1. Melden Sie sich bei dem Rechner an, falls Sie noch nicht eingeloggt sind.
2. Wechseln Sie in das Verzeichnis, in dem die CMS-Applikation installiert ist.
3. Führen Sie die Skript-Datei `client` aus.
4. Geben Sie den Befehl
   `connect` *`host port`*
   ein, wobei *`host`* der Rechnername und *`port`* die Portnummer ist. Beide Angaben sind installationsabhängig.
5. Bei Anmeldung am Content Manager werden Sie aufgefordert, Ihren Anmeldenamen und Ihr Passwort einzugeben. Geben Sie diese Daten bitte ein.

Bei korrekter Eingabe aller Daten sind Sie nun mit dem Server verbunden und können Tcl-Befehle eingeben oder Tcl-Skripte aufrufen.

Die komplette Aufrufsyntax des Tcl-Clients lautet:

```
client [-f scriptPath ] [host [port [login [password ]]]]
```

Die Argumente bedeuten:

- *scriptPath*: Der Pfad eines Tcl-Skripts. Dieses Skript wird mit dem Befehl `source` eingelesen und ausgeführt, nachdem die Verbindung zum Server hergestellt wurde.
- *host*: Der Name des Tcl-Servers.
- *port*: Der Port, unter dem der Tcl-Server zu erreichen ist.
- *login*: Der Anmeldename des Benutzers.
- *password*: Das Passwort des Benutzers.

2.2 Tastaturbefehle

In die Tcl-Shell sind die GNU-Libraries `readline` und `history` eingebunden. Damit sind beispielsweise folgende Tastenkombinationen nutzbar (EMACS-Notation):

Cursorposition

`<CTRL>-a`	Springt zum Anfang der Zeile
`<CTRL>-e`	Springt zum Ende der Zeile
`<META>-b`	Springt zum vorangehenden Wortanfang
`<META>-f`	Springt zum nächsten Wortanfang

Zeile bearbeiten

`<CTRL>-k`	Löscht die aktuelle Zeile
`<CTRL>-h` `<backspace>`	Löscht den vorangehenden Buchstaben
`<META>-<CTRL>-h` `<META>-<backspace>`	Löscht das vorangehende Wort
`<CTRL>-t`	Vertauscht die beiden vorangehenden Buchstaben
`<META>-t`	Vertauscht das aktuelle mit dem vorangehenden Wort
`<META>-r`	Zeile aus der History wiederherstellen
`<CTRL>-_`	Letzte Änderung zurücknehmen

Kommando-History

`<CTRL>-p` `<CURSOR_UP>`	Vorhergehender Befehl
`<CTRL>-n` `<CURSOR_DOWN>`	Nächster Befehl
`<CTRL>-r`	Inkrementelle Suche rückwärts
`<CTRL>-s`	Inkrementelle Suche vorwärts (funktioniert auf vielen Terminals nicht, da <CTRL>-s als X-OFF interpretiert wird)

`<META>-<`	Erster Befehl der History
`<META>->`	Letzter Befehl der History

Weitere Befehle

`<CTRL>-l`	Fensterinhalt löschen
`<META><number><command>`	Wiederholt `<command>` `<number>` -mal.

2.3 Eigene Tcl-Prozeduren verwenden

Eigene Tcl-Prozeduren, die Ihnen persönlich nach jedem Start der Tcl-Shell zur Verfügung stehen sollen, können Sie in einer beliebigen Datei deklarieren und mit dem Tcl-Befehl `source` einlesen.

Bei der Initialisierung werden Tcl-Scripte aus den folgenden Verzeichnissen eingelesen:

```
share/script/common/clientCmds
share/script/app/clientCmds
share/script/common/serverCmds
share/script/app/serverCmds

instance/default/script/common/clientCmds
instance/default/script/app/clientCmds
instance/default/script/common/serverCmds
instance/default/script/app/serverCmds
```

In den obigen Pfaden steht *app* für das applikationsspezifische Skriptverzeichnis *cm* oder *ses*. Beachten Sie bitte, dass die Template Engine beim Start die Skripte des Content Management Servers einliest. Skripte in den Verzeichnissen `common/clientCmds` und `common/serverCmds` werden beim Start aller Applikationen eingelesen.

Skripte in den Verzeichnissen, die in der ersten Gruppe aufgeführt sind, sind allen Instanzen zugeordnet und werden beim Start der entsprechenden CMS-Applikation eingelesen, unabhängig von der Instanz.

Ihre instanzenspezifischen Skripte können Sie in den Verzeichnissen der zweiten Gruppe ablegen. Diese Skripte werden nach den Skripten aus der ersten Gruppe eingelesen, so dass sich Prozeduren, die aus der ersten Gruppe stammen, bei Bedarf durch instanzenspezifische Prozeduren redefinieren lassen.

Tcl-Skripte erleichtern Ihnen periodisch wiederkehrende oder umfangreiche Arbeiten erheblich. Bitte prüfen Sie Ihre Skripte mit Testdaten, bevor Sie mit Produktionsdaten arbeiten. In den meisten Fällen ist es nicht problemlos möglich, per Skript durchgeführte Änderungen rückgängig zu machen.

2.4 Onlinehilfe aufrufen

Um von der Tcl-Shell aus Hilfe zu den Befehlsgruppen und zu den weiteren implementierten Tcl-Funktionen zu erhalten, geben Sie am Prompt der Tcl-Shell

```
help command
```

ein, wobei *command* für eines der Schlüsselwörter steht, die ausgegeben werden, wenn Sie nur `help` eingeben.

2.5 Das Encoding anpassen

Unter Windows kann der Tcl-Client die im Terminal verwendete Zeichensatz-Kodierung nicht zuverlässig ermitteln. Er nimmt daher an, dass die unter Windows übliche Kodierung, cp1252, aktiv ist. Ist dies nicht der Fall oder müssen Daten in einer anderen, umfangreicheren Kodierung (wie UTF-8) übertragen werden, so muss die Kodierung des Terminals korrekt eingestellt werden. Dies geschieht mit den folgenden Befehlen, hier am Beispiel von UTF-8:

```
CM>fconfigure stdout -encoding utf-8
```

```
CM>fconfigure stdin -encoding utf-8
```

```
CM>fconfigure stderr -encoding utf-8
```

Zusätzlich kann es erforderlich sein, die Kodierung der Daten, die aus Dateien gelesen oder in Dateien geschrieben werden, anzupassen, damit beispielsweise mit `source` eingelesene Skripte korrekt interpretiert werden. Verwenden Sie für diese Anpassung den folgenden Befehl:

```
CM>encoding system utf-8
```

2.6 Der abgesicherte Tcl-Interpreter

Tcl-Skripte werden in einem Interpreter ausgeführt. Die Skriptsprache verfügt über einen sicheren Interpreter, in dem jeglicher Zugriff auf das System gesperrt ist. Damit wird die Gefahr einer Kompromittierung des Server-Systems ausgeschlossen.

Skripte, die über das GUI oder die XML-Schnittstelle eingepflegt werden können, werden stets im sicheren Interpreter ausgeführt. Dies betrifft die folgenden Checks und Funktionen:

- Wertzuweisungsfunktion (`callback`) und Wertanzeigefunktion (`displayValueCallback`) bei Feldern.
- Contentzuweisungsfunktion (`recordSetCallback`). Hier steht (zusätzlich) das Kommando `open`, beschränkt auf die übergebenen Blob-Dateien, zur Verfügung.
- Workflowzuweisungsfunktion (`workflowModification`)
- Vollständigkeitscheck (`completionCheck`)

Umgekehrt gilt, dass sämtliche Routinen, die Schreibzugriff auf Dateien benötigen, nicht im sicheren Interpreter, sondern im Standard-Interpreter ausgeführt werden:

- Die Linkfunktion (`linkCallback`)
- Die Post-Action-Funktion (`notificationCmd`)
- SystemExecute-Prozeduren
- Formatierungsprozeduren für Feldwerte und Links (`dynamicLinkFormatter`)
- `generateThumbnail`

- Benutzermanager-Funktionen (*usermanAPI*)
- Alle beim Server-Start gelesenen Tcl-Dateien

Die zu diesen Systemaufrufen gehörenden Prozeduren sind teilweise auch im sicheren Interpreter verfügbar. Für Prozeduren, die durch kundenspezifische Befehle aufgerufen werden, ist dies zwingend, für Benutzermanager-Funktionen sinnvoll.

Tcl-Prozeduren können mit dem Befehl `safeInterp alias` *`serverProc clientProc`* im sicheren Interpreter registriert werden.

3

3 Über die Befehlsbeschreibungen

3.1 Struktur der Befehlsbeschreibungen

Die Tcl-Befehlsbeschreibungen in diesem Handbuch sind nach Befehlsgruppen (beispielsweise `obj` oder `attribute`) geordnet. Die Beschreibungen der Befehle einer Gruppe werden durch eine tabellarische Aufstellung der Felder oder Parameter dieser Gruppe ergänzt. Zu jedem Feld oder Parameter ist der Name und der Typ seines Wertes angegeben. Ferner wird seine Bedeutung erläutert, und es ist angegeben, wie auf das Feld oder den Parameter zugegriffen werden kann (lesend, schreibend). Ferner ist angegeben, ob der Wert des Feldes oder Parameters gegebenenfalls bei der Erzeugung einer neuen Instanz (einer Datei, einer Vorlage usw.) gesetzt werden kann.

Die Befehlsbeschreibungen selbst enthalten die Funktionsdeklaration mit anschließender Erläuterung der Aufgabe, der Parameter und des Rückgabewertes. Ferner werden die Rechte genannt, die ein Benutzer haben muss, um den Befehl ausführen zu können.

3.2 Datentypen und Notation

Die Datentypen der Feldwerte und -parameter sowie der Rückgabewerte der Funktionsaufrufe können sein:

- string: Zeichenkette
- stringlist: Listen und Felder von strings
- number: Ganzzahl
- datetime: Datum und Uhrzeit
- bool: Logische Werte (`no | yes` oder `false | true` oder `0 | 1`)
- void: nichts (kein Rückgabewert, kein Parameter)

In Tcl-Funktionsdeklarationen werden die folgenden Symbole verwendet:

Symbol	Verwendung und Beispiel
( )	Fasst Elemente logisch zusammen: `obj (withId id) | (withPath path) get parameter`
[ ]	Das geklammerte Element ist optional: `logout [login]`
{ }	Das geklammerte Element muss wenigstens einmal vorkommen: `obj withId id mget {parameter}`

| Der vertikale Strich bedeutet, dass entweder das links oder das rechts davon stehende Element angegeben werden muss:
`obj (withId id) | (withPath path) get parameter`

3.3 Datumsformat

Intern werden alle Datums- und Zeitstempelangaben in kanonischer Form als 14stelliger String (von links beginnend: Jahr vierstellig, Monat zweistellig, Tag zweistellig, Stunde zweistellig, Minute zweistellig, Sekunde zweistellig) in GMT gespeichert.

Zur Konvertierung von Datums- und Zeitstempelangaben von der kanonischen in eine konventionelle Form siehe Der SystemConfig-Befehl. Bei der Umwandlung eines Datums oder einer Zeitangabe werden immer die Benutzer-Voreinstellungen für die Zeitzone und das Ausgabeformat berücksichtigt, wenn Sie statt des SystemConfig-Befehls den UserConfig-Befehl verwenden (siehe Der UserConfig-Befehl).

3.4 Anforderungen an Bezeichner

Sämtliche Bezeichner, die im Content Manager (für Feldnamen, Logins, Gruppennamen, Dateinamen usw.) verwendet werden, dürfen weder Leerzeichen noch Sonderzeichen enthalten. Als Sonderzeichen gelten alle Zeichen außer a-z, A-Z, 0-9 und der Unterstrich. Die Namen von CMS-Dateien dürfen auch das Dollar-Zeichen und den Bindestrich enthalten. Beim Import werden nicht erlaubte Zeichen in CMS-Dateinamen in Unterstriche umgewandelt.

4

4 Die Tcl-Befehle des Content Managers und der Template Engine

Die Tcl-Befehle des Content Management Servers und der Template Engine sind entsprechend ihrer Anwendungsbereiche in mehrere Gruppen unterteilt.

4.1 attribute (Versionsfelder)

4.1.1 Feldparameter

Auf die Feldparameter können Sie mit den Feldbefehlen zugreifen (Beispiele sind bei den einzelnen Befehlsbeschreibungen zu finden). Welche Art des Zugriffs auf einen Feldparameter möglich ist, ist in den Spalten rechts neben der Erklärung ausgewiesen. Die Spalte `create` gibt an, ob ein Parameter bei der Erzeugung eines Feldes angegeben werden kann. `descr` zeigt an, ob ein Feldparameter in der Beschreibung eines Feldes (siehe den Feldbefehl `description`) aufgeführt ist.

Parameter	Typ	Erklärung	get	set	create	descr
callback	string	Tcl-Script, das aufgerufen wird, nachdem dem Feld ein Wert zugewiesen wurde	•	•	•	•
displayTitle	string	Der in der HTML-Benutzerschnittstelle angezeigte Titel des Feldes	•			
displayValueCallback	string	Tcl-Script, das aufgerufen wird, um den Anzeigewert eines Versionsfeldes zu berechnen (siehe auch den `displayValue` -Parameter im Abschnitt Versionsfelder)	•	•	•	•
editField	stringlist	Definition des bei Wertzuweisung zu verwendenden Eingabefeldes.	•			•
editFieldSpec	stringlist	vollständige Spezifikation des Eingabefeldes - enthält auch Feldnamen und Aufzählungswerte	•			
getKeys	stringlist	Liste der mit `get` abfragbaren Parameter	•			
helpText	string	Hilfstext zum Feld	•	•	•	•

helpText. *language*	string	Hilfstext zum Feld in der betreffenden Sprache (aus der Localizer-Sektion in der Systemkonfiguration)	•	•	•	•
isSearchableInCM	bool	Gibt an, ob die Werte des Feldes bei laufender Suchmaschine im Content Manager durchsucht werden können.	•	•	•	•
isSearchableInTE	bool	Gibt an, ob die Werte des Feldes bei laufender Suchmaschine auf dem Live-Server durchsucht werden können.	•	•	•	•
maxSize	integer	Maximalanzahl der Links in der Liste (ab Version 6.7.0)	•	•	•	•
minSize	integer	Mindestanzahl der Links in der Liste (ab Version 6.7.0)	•	•	•	•
name	string	Name des Feldes	•		•	•
setKeys	stringlist	Liste der mit `set` setzbaren Parameter	•			
title	string	Titel des Feldes in der benutzerspezifischen Sprache	•	•	•	•
title. *language*	string	Titel des Feldes in der betreffenden Sprache (aus der Localizer-Sektion der Systemkonfiguration)	•	•	•	•
type	string	Typ des Feldes (zulässig: `string` (voreingestellt), `text, date, enum, multienum, html, signature, linklist`, ab Version 6.6 auch `markdown`)	•		•	•
validEditFieldKeys	stringlist	Liste der möglichen Parameternamen in der Eingabefeld-Definition (abhängig vom Eingabefeld-Typ)	•			
validEditFieldTypes	stringlist	Liste der möglichen Eingabefeld-Typen (abhängig vom Feldtyp)	•			
values	stringlist	Aufzählungswerte (nur bei *enum*- und *multienum*- Feldern)	•	•	•	•
wantedTags	stringlist	Liste der im Feldwert erlaubten HTML-Tags (nur bei Feldern vom Typ `html` und `markdown`) (der leere Wert bedeutet: alle).	•	•	•	•

4.1.2 Eingabefeldparameter

Eingabefeld-Parameter	Typ	Erklärung	get	set
editorUrl	string	Bis vor NPS 6.0: Bei Feldern vom Typ HTML die URL, mit der der Zusatzeditor aufgerufen werden kann (als Parameter verfügbar nur beim Eingabefeldtyp `custom`)	•	•

length	number	Die für ein Feld am besten geeignete Breite (als Parameter verfügbar nur bei `textfield`, `passwordfield`, `textarea`)	•	•
maxlength	number	Die maximale Länge des in ein Feld eingebbaren Textes (als Parameter verfügbar nur bei `textfield`, `passwordfield`)	•	•
nilAllowed	bool	Gibt bei Eingabefeldern vom Typ `multiselect` und `popup` an, ob neben den Aufzählungswerten des Feldes auch der leere Wert ausgewählt werden kann. Dieser wird im Eingabefeld als langer Strich angezeigt.	•	•
objClasses	list	Eine Liste von Vorlagen, auf die die Linkzielauswahl eingeschränkt werden soll (als Parameter verfügbar bei `linklistfield` ab Version 6.7.0).	•	•
rows	number	Die Anzahl der darzustellenden Zeilen des Feldes (als Parameter verfügbar nur bei `textarea`, `multiselect`).	•	•
startPub	string	Der Pfad des Startordners für die Linkzielauswahl (als Parameter verfügbar bei `linklistfield` ab Version 6.7.0).	•	•
type	string	Der Typ des Eingabefeldes. Folgende Typen sind verfügbar: `textfield`, `passwordfield`, `textarea`, `multiselect`, `popup`, `radio`, `custom`, `html`, `checkbox`, `hidden`, `external`, `linklistfield`, `wizard`. Nicht alle Typen sind bei allen Feldtypen verfügbar.	•	•
wizard	string	Der zu verwendende Assistent (als Parameter nur verfügbar, wenn `type` gleich `wizard` ist). Der Name des Assistenten entspricht seinem Tcl-Namensraum.	•	•

Der Eingabefeldtyp `linklistfield` ist erst ab Version 6.5.0 verfügbar. Ab dieser Version können Linklisten auch mit einem Assistenten bearbeitet werden.

4.1.3 Feldbefehle

attribute create

Verfügbar für: Content Management Server

Aufgabe: Erzeugt eines neues zusätzliches (kundenspezifisches) Feld mit den angegebenen Werten.

Syntax:

`attribute create {`*`parameter value`*`}`

Funktionsparameter:

- *`parameter`* spezifiziert den Namen des Feldarameters, dessen Wert beim Anlegen des Feldes gesetzt werden soll. Der Feldname muss angegeben werden.
- *`value`* ist der zu setzende Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: Name des Feldes (string).

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel:

```
CM>attribute create name texttype type multienum
texttype
```

attribute list

Verfügbar für: Content Management Server

Aufgabe: Listet die Namen sämtlicher Felder auf.

Syntax:

```
attribute list
```

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die Liste der Feldnamen (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>attribute list
author revision toc_list_type
```

attribute types

Verfügbar für: Content Management Server

Aufgabe: Listet die zulässigen Feldtypen auf.

Syntax:

```
attribute types
```

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die Liste der Feldtypen (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>attribute types
date enum html multienum signature string text markdown
```

attribute where

Verfügbar für: Content Management Server

Aufgabe: Sucht nach Feldern mit bestimmten Parameterwerten, beispielsweise nach Feldern, deren Typ (`type`) `string` ist oder deren Name (`name`) eine gegebene Zeichenkette enthält. Wenn mehr als ein Parameter/Wert-Paar angegeben ist, müssen bei einem Feld sämtliche Suchkriterien erfüllt sein, damit das Feld in das Suchergebnis aufgenommen wird.

Funktionsparameter: (Wenn keine Parameter angegeben werden, gibt der Befehl alle Felder aus.)

Syntax:

`attribute where {parameter value}`

- *parameter* spezifiziert den Namen des Parameters, nach dessen Wert gefragt werden soll oder legt die Anzahl der Suchergebnisse fest. Folgende Parameter können angegeben werden:
  - `maxResults` gibt an, dass *value* eine Zahl ist, die die maximale Anzahl Suchergebnisse festlegt. Ist diese Zahl kleiner oder gleich null, ist die Anzahl der Suchergebnisse nicht beschränkt.
  - `name` gibt an, dass die Namen der Felder auf das Vorkommen von *value* untersucht werden sollen.
  - `type` gibt an, dass *value* ein einzelner Feldtyp oder eine Liste von Feldtypen ist. Nur Felder, deren Typ dem angegebenen Typ oder einem Typ in der Liste entsprechen, werden ins Suchergebnis aufgenommen.
- *value* spezifiziert den Suchbegriff für den betreffenden Parameter.

Rückgabewert bei Erfolg: die Liste der Namen der Felder, auf die die Suchkriterien zutreffen (stringlist)

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>attribute where name dat
datenblatt datenblatt1 datenblatt2
```

attribute attrRef addEnumValues

Verfügbar für: Content Management Server

Aufgabe: fügt zu einem `enum`- oder `multienum`-Feld die angegebenen Aufzählungswerte hinzu, falls diese nicht bereits existieren.

Syntax:

`attribute withName attrName addEnumValues {enumValue}`

Funktionsparameter:

- *enumValue* (string) gibt einen neuen Aufzählungswert an, der zu den bereits existierenden Aufzählungswerten des Feldes hinzugefügt werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel:

```
CM>attribute withName texttype addEnumValues Roman Novelle
Roman Novelle
```

attribute attrRef delete

Verfügbar für: Content Management Server

Aufgabe: löscht das angegebene Feld.

Syntax:

`attribute withName *attrName* delete`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel:

```
CM>attribute withName texttype delete
```

attribute attrRef description

Verfügbar für: Content Management Server

Aufgabe: Liefert eine String-Repräsentation der Daten des Feldes mit dem angegebenen Namen.

Zusatzinformationen: Das Format der String-Repräsentation ist das Property-List-Format eines Dictionarys.

Syntax:

`attribute *attrRef* description`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die String-Repräsentation des Feldes (string).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>attribute withName datenblatt description
{
  editField = {
    length = 65;
    type = textfield;
  };
  name = datenblatt;
  title = datenblatt;
  type = string;
}
```

attribute attrRef editField get

Verfügbar für: Content Management Server

Aufgabe: Liefert den Wert des angegebenen Parameters der Eingabefeld-Spezifikation des Feldes *attrName*.

Syntax:

```
attribute withName attrName editField get parameter
```

Funktionsparameter:

- *parameter* spezifiziert den Namen des Parameters der Eingabefeld-Spezifikation, dessen Wert gesucht wird.

Rückgabewert bei Erfolg: der Wert des angegebenen Eingabefeld-Parameters (string).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>attribute withname datenblatt editfield get type
textfield
```

attribute attrRef editField mget

Verfügbar für: Content Management Server

Aufgabe: Liefert die Werte der angegebenen Parameter der Eingabefeld-Spezifikation des Feldes *attrName*.

Syntax:

```
attribute withName attrName editField mget {parameter}
```

Funktionsparameter:

- *parameter* spezifiziert den Namen des Parameters der Eingabefeld-Spezifikation, dessen Wert gesucht wird.

Rückgabewert bei Erfolg: die Werte der angegebenen Eingabefeld-Parameter (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>attribute withname datenblatt editfield mget type length
textfield 65
```

attribute attrRef editField set

Verfügbar für: Content Management Server

Aufgabe: Setzt die spezifizierten Eingabefeld-Parameter des Feldes *attrName* auf die angegebenen Werte.

Syntax:

`attribute withName attrName editField set {parameter value}`

Funktionsparameter:

- *parameter* (string) spezifiziert den Eingabefeld-Parameter des Feldes, dessen Wert gesetzt werden soll.
- *value* (string) der zu setzende Wert des betreffenden Eingabefeld-Parameters.

Rückgabewert bei Erfolg: 1

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel:

```
CM>attribute withName datenblatt editField set type textfield
```

attribute attrRef get

Verfügbar für: Content Management Server

Aufgabe: Liefert den Wert des angegebenen Feldparameters des Feldes *attrName*.

Syntax:

`attribute withName attrName get parameter`

Funktionsparameter:

- *parameter* spezifiziert den Namen des Parameters, dessen Wert gesucht wird.

Rückgabewert bei Erfolg: der Wert des angegebenen Parameters (string).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>attribute withName datenblatt get editField
type textfield length 65
```

attribute attrRef mget

Verfügbar für: Content Management Server

Aufgabe: Liefert die Werte der angegebenen Parameter des Feldes *attrName*.

Syntax:

`attribute withName attrName mget {parameter}`

Funktionsparameter:

- *parameter* spezifiziert den Namen des Parameters, dessen Wert gesucht wird.

Rückgabewert bei Erfolg: die Liste der Werte der betreffenden Parameter (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>attribute withName datenblatt mget type editField
string {type textfield length 65}
```

attribute attrRef removeEnumValues

Verfügbar für: Content Management Server

Aufgabe: löscht bei einem Aufzählungs- oder Mehrfachaufzählungsfeld (`enum`/`multienum`) die angegebenen Aufzählungswerte.

Syntax:

`attribute withName` *`attrName`* `removeEnumValues {`*`enumValue`*`}`

Funktionsparameter:

- *`enumValue`* gibt einen Aufzählungswert an, der aus den definierten Aufzählungswerten des Feldes gelöscht werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel:

```
CM>attribute withName texttype removeEnumValues Roman
```

attribute attrRef set

Verfügbar für: Content Management Server

Aufgabe: Setzt Parameter eines Feldes auf die angegebenen Werte.

Syntax:

`attribute withName` *`attrName`* `set {`*`parameter value`*`}`

Funktionsparameter:

- *`parameter`* spezifiziert den Parameter des Feldes, dessen Wert gesetzt werden soll.
- *`value`* ist der zu setzende Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel:

```
CM>attribute withName texttype set title Texttyp
```

4.2 attributeGroup (Feldergruppen)

4.2.1 AttributeGroup-Parameter

Parameter	Typ	Erklärung	get	set	create	descr
attributes	stringlist	Die Namen der Felder, die die Feldergruppe enthält. Die Namen werden in der Reihenfolge zurückgegeben, in der die Felder der Gruppe zugeordnet sind.	•			•
displayTitle	string	Der in der HTML-Benutzerschnittstelle angezeigte Titel der Feldergruppe	•			
getKeys	stringlist	Liste der mit `get` abfragbaren Parameter	•			
identifier	string	Name der Dateivorlage und Name der anzulegenden Feldergruppe, getrennt durch einen Punkt.			•	
index	number	Der Index der Gruppe in der Feldgruppenliste der Dateivorlage. Die erste Gruppe hat den Index 0.	•		•	•
isDefaultGroup	bool	Liefert *1*, wenn die Gruppe die Basisgruppe ist, ansonsten *0*.	•			
isEmpty	bool	Liefert *1*, wenn die Gruppe keine Felder enthält, ansonsten *0*.	•			
localizedTitle	string	Der lokalisierte Titel in der Sprache, die der Benutzer eingestellt hat. Ist dieser Titel nicht belegt, so wird der Name der Gruppe zurückgegeben.	•			
name	string	Name der Feldergruppe. Der Name der Basisgruppe (`baseGroup`) kann nicht geändert werden.	•	•		•
objClass	string	Der Name der Dateivorlage, zu der die Gruppe gehört.	•			
setKeys	stringlist	Liste der mit `set` setzbaren Parameter.	•			
title	string	Titel der Feldgruppe in der benutzerspezifischen Sprache	•	•	•	•
title. *language*	string	Titel der Feldergruppe in der betreffenden Sprache	•	•	•	•

4.2.2 AttributeGroup-Befehle

In den AttributeGroup-Befehlen werden die Feldergruppen über einen aus zwei Teilen bestehenden Bezeichner referenziert. Als ersten Teil geben Sie bitte den Namen der Dateivorlage an, die die Gruppe enthält. Als Trennzeichen zwischen dem ersten und dem zweiten Teil wird der Punkt verwendet. Als zweiten Teil geben Sie den Namen der Gruppe an. Beispiel: `myObjectClass.myAttributeGroup`.

attributeGroup create

Verfügbar für: Content Management Server

Aufgabe: Legt eine Feldergruppe an.

Syntax:

`attributeGroup create {`*`parameter value`*`}`

Funktionsparameter:

- *`parameter`* spezifiziert den Namen eines `attributeGroup`-Parameters, dessen Wert bei der Erzeugung der Gruppe gesetzt werden soll. Der Parameter *`identifier`* muss angegeben werden.
- *`value`* ist der zu setzende Wert des betreffenden Parameters. Als Wert des Parameters *`identifier`* muss der Bezeichner der neuen Feldergruppe angegeben werden. Er wird zusammengesetzt aus dem Dateivorlagennamen, einem Punkt und dem Namen der neuen Gruppe.

Rückgabewert bei Erfolg: Der Name der neuen Gruppe.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel: In der Dateivorlage `newsPub` eine Feldergruppe mit dem Namen `headlines` und dem Titel `Schlagzeilen` anlegen:

```
CM>attributeGroup create identifier newsPub.headlines \
title Schlagzeilen
headlines
```

attributeGroup attrGroupRef addAttribute

Verfügbar für: Content Management Server

Aufgabe: Mit diesem Befehl wird ein Feld in eine Feldergruppe aufgenommen.

Zusatzinformationen: Das Feld muss der Dateivorlage zugewiesen sein, zu der die Gruppe gehört. Der Basisgruppe können keine Felder zugewiesen werden. Nur Felder, die sich in der Basisgruppe befinden, können einer anderen Gruppe zugewiesen werden.

Syntax:

`attributeGroup withIdentifier groupId addAttribute {`*`parameter value`*`}`

Funktionsparameter:

- *`parameter`* spezifiziert den Namen eines bei der Zuweisung zu berücksichtigenden Parameters. Folgende Parameter können angegeben werden.
  - `attribute` gibt an, dass *`value`* den Namen des Feldes enthält. Dieser Parameter ist obligatorisch.
  - `index` gibt an, dass *`value`* die Stelle spezifiziert, an der das Feld in die Felderliste der Feldergruppe eingefügt werden soll. Ist dieser Parameter nicht angegeben, so wird das Feld an die Felderliste angehängt. Das erste Feld einer Feldergruppe hat den Index 0.
- *`value`* gibt den Wert des betreffenden Parameters an.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel: In der Dateivorlage `newsPublication` zur Feldergruppe `headlines` das Feld `main` an erster Stelle hinzufügen:

```
CM>attributeGroup withIdentifier newsPub.headlines \
addAttribute attribute main index 0
```

attributeGroup attrGroupRef addAttributes

Verfügbar für: Content Management Server

Aufgabe: Mit diesem Befehl können mehrere Felder in eine Feldergruppe aufgenommen werden.

Zusatzinformationen: Jedes der aufzunehmenden Felder muss der Dateivorlage zugewiesen sein, zu der die Feldergruppe gehört. Der Basisgruppe können keine Felder zugewiesen werden. Nur Felder, die sich in der Basisgruppe befinden, können einer anderen Gruppe zugewiesen werden. Die Felder werden in der angegebenen Reihenfolge an die Felderliste der Feldergruppe angehängt.

Syntax:

`attributeGroup withIdentifier groupId addAttributes {`*`attribute`*`}`

Funktionsparameter:

- *`attribute`* spezifiziert den Namen eines Feldes, das der Gruppe zugewiesen werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel: In der Dateivorlage `newsPub` zur Feldergruppe `headlines` die Felder `head1` und `head2` hinzufügen:

```
CM>attributeGroup withIdentifier newsPub.headlines \
addAttributes head1 head2
```

attributeGroup attrGroupRef delete

Verfügbar für: Content Management Server

Aufgabe: Dieser Befehl löscht eine Feldergruppe.

Zusatzinformationen: Die in der Gruppe enthaltenen Felder werden automatisch zur Basisgruppe hinzugefügt. Die Basisgruppe kann nicht gelöscht werden.

Syntax:

`attributeGroup withIdentifier` *`groupId`* `delete`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel: In der Dateivorlage `newsPub` die Feldergruppe `related` löschen:

```
CM>attributeGroup withIdentifier newsPub.related delete
```

attributeGroup attrGroupRef description

Verfügbar für: Content Management Server

Aufgabe: Liefert eine String-Repräsentation der Daten einer Feldergruppe. Die Repräsentation wird im Property-List-Format eines Dictionarys formatiert.

Syntax:

`attributeGroup withIdentifier` *`groupId`* `description`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die Stringrepräsentation der Feldergruppe (string).

Erforderliche Rechte: keine.

Beispiel:

```
CM>attributeGroup withIdentifier newsPub.headlines description
{
  name = headlines;
  title = Schlagzeilen;
  title.de = "";
  title.en = "";
  index = 1;
  attributes = (head1, head2);
}
```

attributeGroup attrGroupRef get

Verfügbar für: Content Management Server

Aufgabe: Mit diesem Befehl kann ein attributeGroup-Parameter ausgelesen werden.

Syntax:

`attributeGroup withIdentifier` *`groupId`* `get` *`parameter`*

Funktionsparameter:

- *`parameter`* spezifiziert den Namen des `attributeGroup`-Parameters, der ausgelesen werden soll.

Rückgabewert bei Erfolg: der Wert des auszulesenden Parameters.

Erforderliche Rechte: keine.

Beispiel: Die Felder ermitteln, die der Feldergruppe `newsPub.headlines` zugewiesen sind:

```
CM>attributeGroup withIdentifier newsPub.headlines get attributes
head1 head2
```

attributeGroup withIdentifier groupId mget {parameter }

Verfügbar für: Content Management Server

Aufgabe: Mit diesem Befehl können mehrere `attributeGroup`-Parameter ausgelesen werden.

Funktionsparameter:

- *`parameter`* spezifiziert den Namen eines `attributeGroup`-Parameters, der ausgelesen werden soll.

Rückgabewert bei Erfolg: die Werte der auszulesenden Parameter.

Erforderliche Rechte: keine.

Beispiel: Den Index der Feldergruppe `newsPub.headlines` und die ihr zugewiesenen Felder ermitteln:

```
CM>attributeGroup withIdentifier newsPub.headlines mget index attributes
```

```
1 {head1 head2}
```

attributeGroup attrGroupRef moveAttribute

Verfügbar für: Content Management Server

Aufgabe: Der Befehl verschiebt ein Feld einer Feldergruppe an eine andere Stelle.

Syntax:

`attributeGroup withIdentifier` *`groupId`* `moveAttribute {`*`parameter value`*`}`

Funktionsparameter:

- *`parameter`* spezifiziert den Namen eines bei der Ausführung des Befehls zu berücksichtigenden Parameters. Folgende Parameter müssen angegeben werden.
  - `attribute` gibt an, dass *`value`* den Namen des zu verschiebenden Feldes enthält.
  - `index` gibt an, dass *`value`* die Stelle spezifiziert, an der das Feld in die Felderliste der Gruppe eingefügt werden soll. Das erste Feld einer Gruppe hat den Index 0.
- *`value`* gibt den Wert des betreffenden Parameters an.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel: Das Feld `main` in der Feldergruppe `newsPub.headlines` an die zweite Stelle verschieben:

```
CM>attributeGroup withIdentifier newsPub.headlines moveAttribute attribute main index 1
```

attributeGroup attrGroupRef moveToIndex

Verfügbar für: Content Management Server

Aufgabe: Der Befehl verschiebt eine Feldergruppe an eine andere Stelle in der Feldergruppenliste einer Dateivorlage.

Syntax:

`attributeGroup withIdentifier` *`groupId`* `moveToIndex` *`index`*

Funktionsparameter:

- *`index`* spezifiziert die Stelle, an die die Feldergruppe verschoben werden soll. Die erste Gruppe hat den Index 0.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel: Die Feldergruppe `newsPub.headlines` an die dritte Stelle in der Feldergruppenliste der Dateivorlage verschieben:

```
CM>attributeGroup withIdentifier newsPub.headlines moveToIndex 2
```

attributeGroup attrGroupRef removeAttribute

Verfügbar für: Content Management Server

Aufgabe: Der Befehl löscht ein Feld aus einer Feldergruppe.

Zusatzinformationen: Aus der Basisgruppe können keine Felder gelöscht werden. Das zu löschende Feld wird automatisch zur Basisgruppe hinzugefügt.

Syntax:

`attributeGroup withIdentifier` *`groupId`* `removeAttribute` *`attribute`*

Funktionsparameter:

- *`attribute`* spezifiziert den Namen des zu löschenden Feldes.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel: Aus der Feldergruppe `newsPub.headlines` das Attribut `head2` löschen:

```
CM>attributeGroup withIdentifier newsPub.headlines removeAttribute head2
```

attributeGroup attrGroupRef set

Verfügbar für: Content Management Server

Aufgabe: Mit diesem Befehl können `attributeGroup`-Parameter gesetzt werden.

Syntax:

`attributeGroup withIdentifier` *`groupId`* `set {`*`parameter value`*`}`

Funktionsparameter:

- *parameter* spezifiziert den Namen des attributeGroup-Parameters, dessen Wert gesetzt werden soll.
- *value* spezifiziert den Wert, auf den der betreffende attributeGroup-Parameter gesetzt werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht permissionGlobalRTCEdit haben.

Beispiel: Den englischen Titel der Feldergruppe newsPub.headlines setzen:

```
CM>attributeGroup withIdentifier newsPub.headlines \
set title.en Headlines
```

4.3 channel (News-Listen)

4.3.1 Channel-Parameter

Parameter	Typ	Erklärung	get	set	create	descr
name	string	Name des Channels.	•	•	•	•
title. *language*	string	Titel des Channels in der betreffenden Sprache. Die Sprachkürzel sind im Systemkonfigurationseintrag *localizers* definiert.	•	•	•	•

4.3.2 Channel-Befehle

Mit Channels hat man die Möglichkeit, Inhalte in Dateiversionen thematisch zu kategorisieren. Bei Einsatz des Portal Managers können Website-Besucher Channels abonnieren, um so auf die von ihnen bevorzugten Dokumente zuzugreifen.

Die Channels werden im Content Management Server verwaltet. Mit Hilfe des Versionsfeldes channels können Versionen beliebig vielen der definierten Channels zugewiesen werden.

channel create

Verfügbar für: Content Management Server

Aufgabe: Der Befehl legt einen Channel an.

Syntax:

channel create {*parameter value*}

Funktionsparameter:

- *parameter* spezifiziert den Namen eines Channel-Parameters, dessen Wert bei der Erzeugung des Channels gesetzt werden soll. Der Parameter `name` muss angegeben werden.
- *value* ist der zu setzende Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: Der Name des neuen Channels.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel:

```
CM>channel create name weather.europe title.de "Europawetter"
weather.europe
```

channel list

Verfügbar für: Content Management Server

Aufgabe: Der Befehl gibt die Liste der Namen sämtlicher Channels aus.

Syntax:

`channel list`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>channel list
weather.europe weather.europe.france weather.europe.germany
```

channel where

Verfügbar für: Content Management Server

Aufgabe: ermöglicht die Suche nach Channels, bei denen der Wert der angegebenen Parameter die jeweils spezifizierte Zeichenkette enthält.

Syntax:

`channel where {`*parameter value*`}`

Funktionsparameter: (Wenn keine Parameter angegeben werden, gibt der Befehl alle Channels aus.)

- *parameter* spezifiziert den Namen des Parameters, nach dessen Wert gefragt werden soll oder legt die Anzahl der Suchergebnisse fest. Folgende Parameter sind erlaubt:
  - `maxResults` gibt an, dass *value* eine Zahl enthält, die die maximale Anzahl Suchergebnisse festlegt. Ist diese Zahl kleiner oder gleich null, ist die Anzahl der Suchergebnisse nicht beschränkt.
  - `name` gibt an, dass die Namen der Channels auf Übereinstimmung mit *value* untersucht werden sollen.

- `namePrefix` gibt an, dass die Namen vom Anfang her mit *`value`* verglichen werden sollen.
- *`value`* enthält den Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: die Liste der Namen der passenden Channels (stringlist)

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>channel where namePrefix weather
weather weather.europe weather.europe.france weather.europe.germany
```

channel channelRef delete

Verfügbar für: Content Management Server

Aufgabe: Dieser Befehl löscht den angegebenen Channel.

Syntax:

`channel withName` *`channelName`* `delete`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel:

```
CM>channel withName weather.europe.france delete
```

channel channelRef description

Verfügbar für: Content Management Server

Aufgabe: Liefert eine String-Repräsentation der Daten eines Channels. Die Repräsentation wird im Property-List-Format eines Dictionarys formatiert.

Syntax:

`channel withName` *`channelName`* `description`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die Stringrepräsentation des Channels (string).

Erforderliche Rechte: keine.

Beispiel:

```
CM>channel withName weather.europe description
{
  name = weather.europe;
  title.de = "Europawetter";
```

```
  title.en = "European Weather";
}
```

channel channelRef get

Verfügbar für: Content Management Server

Aufgabe: Mit diesem Befehl kann ein Channel-Parameter ausgelesen werden.

Syntax:

`channel withName` *`channelName`* `get` *`parameter`*

Funktionsparameter:

- *`parameter`* spezifiziert den Namen des Channel-Parameters, der ausgelesen werden soll.

Rückgabewert bei Erfolg: der Wert des auszulesenden Parameters.

Erforderliche Rechte: keine.

Beispiel:

```
CM>channel withName weather.europe get title.en
European Weather
```

channel channelRef mget

Verfügbar für: Content Management Server

Aufgabe: Mit diesem Befehl können mehrere Channel-Parameter ausgelesen werden.

Syntax:

`channel withName` *`channelName`* `mget {`*`parameter`*`}`

Funktionsparameter:

- *`parameter`* spezifiziert den Namen eines Channel-Parameters, der ausgelesen werden soll.

Rückgabewert bei Erfolg: die Werte der auszulesenden Parameter.

Erforderliche Rechte: keine.

Beispiel:

```
CM>channel withName weather.europe get title.en title.de
European Weather Europawetter
```

channel channelRef set

Verfügbar für: Content Management Server

Aufgabe: Mit diesem Befehl können Channel-Parameter gesetzt werden.

Syntax:

`channel withName` *`channelName`* `set {`*`parameter value`*`}`

Funktionsparameter:

- *`parameter`* spezifiziert den Namen des Channel-Parameters, dessen Wert gesetzt werden soll.
- *`value`* spezifiziert den Wert, auf den der betreffende Channel-Parameter gesetzt werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel:

```
CM>channel withName weather.europe set title.en "European Weather"
```

4.4 content (Versionen)

4.4.1 Versionsfelder

Die folgende Aufstellung der Versionsfelder gilt für den Content Management Server. In der Template Engine sind nur die Felder verfügbar, deren Name keine weitere Komponente (wie `.plain`) enthält. Ferner können in der Template Engine die Feldwerte nur gelesen, nicht jedoch gesetzt werden.

Feld	Typ	Erklärung	get	set	descr

anchors	stringlist	Liste der Ankernamen in der Version (nur bei Dateien vom Typ Ordner und Dokument).	•		
blob blob.plain	string	Nur bei Layouts, Ordnern und Dokumenten: String-Repräsentation des Inhalts der Version.	•	•	
blob.base64	string	Der base64-kodierte Inhalt zum Content.	•	•	
blob.stream	string	Gibt bei lesendem Zugriff ein Streaming Ticket zurück, mit dem man den Hauptinhalt (Blob) über das Streaming-Interface abholen kann. Bei schreibendem Zugriff wird das Streaming Ticket angegeben, das man erhalten hat, als der Blob über das Streaming-Interface hochgeladen wurde.	•	•	
blobLength	number	Liefert bei Dateien vom Typ `image` und *generic* die Länge der Binärdaten, bei Dateien der anderen Typen die Größe des Hauptinhalts in Bytes.	•		
body	string	Nur bei Layouts, Ordnern und Dokumenten: Vorverarbeitete Version des HTML-Codes der Version, die für interne Zwecke verwendet wird.	•		
channels	stringlist	Die Liste der Channels, denen die Version zugeordnet ist.	•	•	•
contentType	string	Der Dateityp der Version. Für Versionen von Dateien des Typs `document` und `publication` gilt: Nur wenn der zur Dateiendung gehörende MIME-Typ *text/html* ist, wird der Hauptinhalt der Version vom Content Manager verarbeitet, werden also beispielsweise darin enthaltene Links mit der Linkverwaltung abgeglichen. Andernfalls wird der Hauptinhalt als reiner Text behandelt. Siehe auch den Systemkonfigurationseintrag `mimeTypes`.	•	•	•

displayTitle	string	Der in der HTML-Benutzerschnittstelle angezeigte Titel der Version (entspricht dem Wert von `title`).	•		
editor	string	Der Anmeldename des aktuellen Bearbeiters der Version.	•		•
exportBlob	string	Nur bei Layouts, Ordnern und Dokumenten: String-Repräsentation des Hauptinhalts (des Blobs). Je nach Datei, zu der die Version gehört, wird der Hauptinhalt mittels der zuständigen Layouts in das endgültige Export-Format gebracht (für Ordner und Dokumente). Welche Layouts zuständig sind, hängt auch von der Benutzereinstellung `preferEditedTemplates` ab.	•		
exportBlob.plain	string	wie `exportBlob`	•		
exportBlob.base64	string	wie `exportBlob`, jedoch base64-kodiert.	•		
exportBlob.stream	string	wie `exportBlob`, jedoch mit Hilfe des Streaming-Interfaces übertragen (siehe Beschreibung zu `blob.stream`).	•		
exportFiles	stringlist	Eine Liste, die für jede beim Export der Version erzeugte Datei ein Wertepaar enthält. Der erste Wert des Paares ist der Dateiname (ohne Pfad), der zweite Wert ist der zu dieser Datei gehörende Inhalt.	•		
externalAttrNames	stringlist	Die Liste aller zusätzlichen (kundenspezifischen) Felder, die der Version zugeordnet sind. Die Liste wird über die Dateivorlage der Datei ermittelt, dem die Version zugeordnet ist.	•		
frameNames	stringlist	Die Liste der Namen aller beim Export der Version entstehenden Frames.	•		
freeLinks	stringlist	Die Liste der IDs aller freien Links in der Version.	•		
getKeys	stringlist	Liste der mit `get` abfragbaren Versionsfelder.	•		
hasNewsItem	bool	Gibt an, ob die Version News-Einträge hat, d.h ob es sich um eine freigegebene Version handelt, in deren Vorlage `canCreateNewsItems` 1 ist und die Channels hat.	•		
hasThumbnail	bool	Gibt an, ob die Version ein Thumbnail hat.	•		
height	number	Die Höhe des Bildes bei Versionen, die zu Dateien vom Typ `image` gehören, sofern das Bild eines der unterstützen Formate (GIF, JPG, PNG) hat, andernfalls 0.	•		
isActive (ab Version 6.5.0)	bool	Gibt an, ob die Version zeitlich gültig ist.	•		•
isCommitted	bool	Gibt an, ob es sich um eine eingereichte Version handelt.	•		

isComplete	bool	Gibt an, ob die Version vollständig ist.	•		
isEdited	bool	Gibt an, ob es sich um eine Arbeitsversion handelt..	•		
isReleased	bool	Gibt an, ob es sich um eine freigegebene Version handelt.	•		
lastChanged	string	das Datum der letzten Änderung der Version.	•		
linkListAttributes	stringlist	Die Liste der Felder vom Typ `linklist`, die der Version zugeordnet sind.	•		
mimeType	string	Der MIME-Typ der Version, der sich aus der Zuordnung von Content-Typen zu MIME-Typen in der Systemkonfiguration (Eintrag `mimeTypes`) ergibt.	•		
nextEditGroup	string	Name der nächsten Gruppe im Bearbeitungsworkflow.	•		
nextSignGroup	string	Name der nächsten Gruppe im Unterschriftsworkflow	•		
objectId	string	ID der zur Version gehörenden Datei.	•		
reasonsForIncompleteState	stringlist	Die Liste der Gründe, aus denen eine Arbeitsversion unvollständig ist.	•		
setKeys	stringlist	Liste der mit `set` setzbaren Versionsfelder.	•		
signatureAttrNames	stringlist	Die Liste der Unterschriftsfelder, die der Version zugeordnet sind. Die Liste wird aus dem Workflow der zur Version gehörenden Datei ermittelt.	•		
sortKey1	string	Erste Sortierschlüsselkomponente (nur bei Ordnern setzbar)	•	•	
sortKey2	string	Zweite Sortierschlüsselkomponente (nur bei Ordnern setzbar)	•	•	
sortKey3	string	Dritte Sortierschlüsselkomponente (nur bei Ordnern setzbar)	•	•	
sortKeyLength1	number	Anzahl der signifikanten Zeichen der ersten Sortierschlüsselkomponente (nur bei Ordnern setzbar)	•	•	
sortKeyLength2	number	Anzahl der signifikanten Zeichen der zweiten Sortierschlüsselkomponente (nur bei Ordnern setzbar)	•	•	
sortKeyLength3	number	Anzahl der signifikanten Zeichen der dritten Sortierschlüsselkomponente (nur bei Ordnern setzbar)	•	•	
sortOrder	string	Sortierrichtung (nur bei Ordnern)	•	•	
sortType1	string	Sortiermodus der ersten Sortierschlüsselkomponente (nur bei Ordnern)	•	•	
sortType2	string		•	•	

		Sortiermodus der zweiten Sortierschlüsselkomponente (nur bei Ordnern)			
sortType3	string	Sortiermodus der dritten Sortierschlüsselkomponente (nur bei Ordnern)	•	•	
subLinks	stringlist	die Liste der IDs der in der Version vorkommenden Links	•		
textLinks	stringlist	Die Liste der IDs aller Text- und Einfügungslinks in der Version.	•		
thumbnail	string	Ein base-64-kodiertes Vorschaubild im JPEG-Format. Nur verfügbar bei Dateien vom Typ `image` und `generic`. Dieses Feld kann nur im GUI (Content Navigator, Wizards), nicht jedoch beim Export verwendet werden.	•		
title	string	Der Titel der Version.	•	•	•
validFrom	string	Das Datum des Beginns der Gültigkeit der Version im 14-stelligen kononischen Format (wird bei Layouts ignoriert). Wenn `validUntil` gesetzt ist, muss dessen Wert größer als `validFrom` sein.	•	•	
validSortKeys	stringlist	Die Liste aller gültigen Werte für die Felder `sortKey`*n*.	•		
validSortOrders	stringlist	Die Liste aller gültigen Werte für das Feld `sortOrder`.	•		
validSortTypes	stringlist	Die Liste aller gültigen Werte für die Felder `sortTypen`.	•		
validUntil	string	Das Datum des Endes der Gültigkeit der Version im 14-stelligen kononischen Format (wird bei Layouts ignoriert)	•	•	
width	number	Die Breite des Bildes bei Versionen, die zu Dateien vom Typ `image` gehören, sofern das Bild eines der unterstützen Formate (GIF, JPG, PNG) hat, andernfalls 0.	•		
workFlowComment	string	Der Kommentar zu der letzten Workflow-Aktion, die auf diese Version angewendet wurde.	•		
xmlBlob	string	String-Repräsentation des Hauptinhalts (Blobs) der Version als XML-Dokument (nicht verfügbar bei Layouts)	•	•	
Zusätzliches Feld		Der Wert des zusätzlichen (kundenspezifischen) Feldes.	•	•	•

Zu den Versionsfeldern gehören neben den oben aufgeführten vorgegebenen auch die zusätzlichen (kundenspezifischen) Versionsfelder. Die Werte dieser Felder können ebenfalls ausgelesen und im Content Management Server auch gesetzt werden. Ferner können zusätzliche Felder jeweils einen Anzeigewert haben, der durch die Wertanzeigefunktion des Feldes berechnet wird, wenn dem Feld ein

Wert zugewiesen wird. Anzeigewerte können mit Befehlen nach dem folgendem Muster ausgelesen werden:

```
content withId 65429 get customattribute.displayValue
```

Zur Wertanzeigefunktion siehe auch Feldparameter.

4.4.2 Versionsbefehle

Die Versionsbefehle ermöglichen es, auf CMS-Datei-Versionen direkt zuzugreifen, d.h. über deren ID und nicht über die folgenden Datei-Subbefehle:

- `obj (withId objId) | (withPath path) | root editedContent versionSubCommand`
- `obj (withId objId) | (withPath path) | root releasedContent versionSubCommand`

content contentRef addLinkTo

Verfügbar für: Content Management Server

Aufgabe: Erzeugt einen freien Link und fügt ihn zum angegebenen Feld der Arbeitsversion hinzu.

Syntax:

`content withId` *`contentId`* `addLinkTo attribute` *`attrName`* `destinationUrl` *`urlString`*

Funktionsparameter:

- *`attrName`* ist der Name des Feldes, zu dem der freie Link hinzugefügt werden soll. Das betreffende Feld muss vom Typ `linklist` sein.
- *`target`* ist der Zielrahmen.
- *`title`* ist der Titel des Links.
- *`destinationUrl`* ist das Linkziel.

Rückgabewert bei Erfolg: die ID des erzeugten Links (string).

Erforderliche Rechte: Der Benutzer muss das Recht `permissionWrite` für die Datei haben, zu der die Version gehört sowie der Bearbeiter der Version sein.

Beispiel: Einen freien Link zum Feld `imageLinks` in der Version mit der ID 25687 hinzufügen; der Link verweist auf die Datei `location` im gleichen Ordner:

```
CM>content withId 25687 addLinkTo \
   attribute imageLinks destinationUrl location
9983433
```

Beispiel: Einen freien Link zum Feld `imageLinks` in der Arbeitsversion der Datei `newslist` hinzufügen; der Link verweist auf die Datei `location` im gleichen Ordner:

```
CM>obj withPath /newslist editedContent addLinkTo \
   attribute imageLinks destinationUrl location
9983432
```

content contentRef debugExport

Verfügbar für: Content Management Server (bis Version 6.0.x, ab Version 6.5.0 siehe obj *objRef* debugExport)

Aufgabe: Der Befehl simuliert den Export der Version mit der angegebenen ID zu dem Zweck, fehlerhaften Code in Layouts ausfindig zu machen. Aus der Ausgabe ist ersichtlich, welche NPSOBJ-Tags aus welchen Quell-Layouts in welcher Reihenfolge ausgewertet werden und welche Fehler bei der Auswertung aufgetreten sind.

Zusatzinformationen:

- Der Befehl liefert Angaben zu den folgenden Fehlern: Nicht geschlossene Tags, ungleiche Anzahl öffnender und schließender Tags, nicht erlaubte Felder in NPSOBJ-Tags, nicht erlaubte Werte von NPSOBJ-Tag-Attributen, lesenden Zugriff auf nicht definierte Namen.
- Der Inhalt von `systemExecute`-Anweisungen wird nicht auf Fehler untersucht und nicht formatiert. Die Ausgabe solcher Anweisungen dagegen wird formatiert.

Syntax:

```
content withId contentId debugExport
  [templateName templateName]
  [detailed (yes | no)]
  [quoteHtml (yes | no)]
  [errorPrefix errorPrefix]
  [errorSuffix errorSuffix]
  [htmlPrefix htmlPrefix]
  [htmlSuffix htmlSuffix]
  [infoPrefix infoPrefix]
  [infoSuffix infoSuffix]
  [preferEditedTemplates (yes | no)]
  [allowEditedContents (yes | no)]
```

Funktionsparameter:

- *`templateName`*: Gibt an, mit welcher initialen Layoutdatei der Exporttest durchgeführt werden soll. Ist `templateName` nicht angegeben, wird die benutzerspezifische Standardlayoutdatei verwendet (voreingestellt `mastertemplate`).
- *`detailed`*: Gibt an, ob sich die Ausgabe nur auf Fehlermeldungen beschränken (`no`) oder ausführlich sein soll (`yes`). Der voreingestellte Wert ist `no`.
- *`quoteHtml`*: Gibt an, ob die Zeichen <, > und & als HTML-Entities ausgegeben werden sollen (`yes`) oder nicht (`no`). Die Präfix- und Suffix-Parameter sind hiervon nicht betroffen. Der voreingestellte Wert ist `no`.
- *`errorPrefix`*: Gibt eine Zeichenkette an, die vor jeder Fehlermeldung ausgegeben werden soll. Der voreingestellte Wert ist `"***"`.
- *`errorSuffix`*: Gibt eine Zeichenkette an, die nach jeder Fehlermeldung ausgegeben werden soll (voreingestellt leer).
- *`htmlPrefix`*: Gibt eine Zeichenkette an, die vor HTML-Text ausgegeben werden soll. Der voreingestellte Wert ist "<pre>".
- *`htmlSuffix`*: Gibt eine Zeichenkette an, die nach HTML-Text ausgegeben werden soll. Der voreingestellte Wert ist "</pre>".
- *`infoPrefix`*: Gibt eine Zeichenkette an, die vor jeder NPSOBJ-Information ausgegeben werden soll (voreingestellt leer).

- *infoSuffix*: Gibt eine Zeichenkette an, die nach jeder NPSOBJ-Information ausgegeben werden soll (voreingestellt leer).
- *preferEditedTemplates*: Gibt an, ob die Arbeitsversionen der Layouts gegenüber den freigegebenen Versionen bevorzugt verwendet werden sollen. Der voreingestellte Wert entspricht dem gleichnamigen Wert aus den Benutzereinstellungen, der auch für die Vorschau verwendet wird.
- *allowEditedContents*: Gibt an, ob auf Arbeitsversionen zurückgegriffen werden darf, wenn freigegebene Versionen nicht verfügbar sind. Der voreingestellte Wert entspricht dem Status der Version, für den der Befehl aufgerufen wird. Dieser Parameter bezieht sich nicht auf diese Version, sondern auf die darin referenzierten Versionen.

Rückgabewert bei Erfolg: der formatierte Exportbericht (string).

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalExport` oder `permissionRead` für die zur Version gehörende Datei haben.

Beispiel: (Ausgabe siehe unten)

```
CM>content withId 25687 debugExport detailed yes
```

Beispiel zur Verwendung eines entsprechenden `obj`-Befehls: (Ausgabe stark gekürzt)

```
CM>obj withId 2792 editedContent debugExport detailed yes
<HTML>
    <HEAD>
  ...
NPSOBJ insertvalue=var: title
The title
END NPSOBJ insertvalue=var
</TITLE>
    </HEAD>
NPSOBJ insertvalue=var: body
<a ...>Linked text</a>

*** FEHLER [140008] Ein NPSOBJ-Tag enthielt kein Kommandoattribut.
...
```

content contentRef delete

Verfügbar für: Content Management Server

Aufgabe: Löscht eine archivierte Version. Möchten Sie dagegen eine Arbeitsversion löschen, so verwenden Sie bitte den Befehl `obj revert`.

Syntax:

`content withId` *contentId* `delete`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionRoot` für die Datei haben, zu der die Version gehört.

Beispiel: Alle archivierten Versionen einer Datei löschen:

```
CM>foreach i [obj withPath /news get archivedContentIds] \
   {content withId $i delete}
```

content contentRef description

Verfügbar für: Content Management Server

Aufgabe: Liefert eine String-Repräsentation der Daten der Version mit der angegebenen ID.

Zusatzinformationen: die Repräsentation wird im Property-List-Format eines Dictionarys formatiert.

Syntax:

`content` *`contentRef`* `description`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die Stringrepräsentation der Version (string).

Erforderliche Rechte: Der Benutzer muss das Recht `permissionRead` für die Datei haben, zu der die Version gehört.

Beispiel: (Ausgabe siehe unten)

```
CM>content withId 25687 description
```

Beispiel unter Verwendung des `obj`-Befehls:

```
CM>obj withPath /news editedContent description
{
  contentType = html;
  editor = katrin;
  sortKeyLength1 = 50;
  sortKeyLength2 = 50;
  sortKeyLength3 = 50;
  sortOrder = ascending;
  sortType1 = alphaNumeric;
  sortType2 = alphaNumeric;
  sortType3 = alphaNumeric;
  title = untitled;
  validFrom = 20101216230000;
}
```

content contentRef generateThumbnail

Verfügbar für: Content Management Server

Aufgabe: Erzeugt das Thumbnail (Minaturbild) zur Arbeitsversion eines Bildes mit der angegebenen ID und speichert es im Versionsfeld `thumbnail`. Wenn Bilder in das CMS importiert werden (manuell oder während ein Dump wiederhergestellt wird), werden Thumbnails automatisch generiert. Daher wird diese Funktion nur benötigt, wenn die Thumbnail-Erzeugung beispielsweise wegen inkompatibler Bilddaten fehlgeschlagen ist und die Funktion angepasst wurde.

Syntax:

`content withId` *`contentId`* `generateThumbnail`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss mindestens eines der beiden Rechte `permissionWrite` oder `permissionRoot` für die Datei haben, zu der die Version gehört.

Beispiel:

```
CM>content withId 43219 generateThumbnail
```

content contentRef get

Verfügbar für: Content Management Server

Aufgabe: Liefert den Wert des angegebenen Versionsfeldes zurück.

Syntax:

`content withId` *`contentId`* `get` *`contentAttr`*

Funktionsparameter:

- *`contentAttr`*: der Name des Versionsfeldes, nach dessen Wert gefragt wird.

Rückgabewert bei Erfolg: der Wert des Versionsfeldes (string).

Erforderliche Rechte:

- der Benutzer muss das Recht `permissionRead` für die Datei haben, zu der die Version gehört.
- Um Wert des Feldes `exportBlob` abzufragen, genügt auch das Recht `permissionGlobalExport` .

Beispiel:

```
CM>content withId 25687 get title
Newsliste
```

Auch mittels `obj`-Befehl ist es möglich, Felder einer Version abzufragen:

```
CM>obj withPath /news editedContent get title
Newsliste
```

content contentRef getExportBlobForFrame

Verfügbar für: Content Management Server

Aufgabe: Liefert zur Version mit der angegebenen ID den Inhalt der Datei, die der Content Manager erzeugt, wenn es den Frame mit dem angegebenen Namen exportiert.

Syntax:

`content withId` *`contentId`* `getExportBlobForFrame` *`frameName`*

Funktionsparameter:

- *frameName*: der Name des Frames, nach dessen Export-Blob gefragt wird.

Rückgabewert bei Erfolg: der Export-Blob.

Erforderliche Rechte: Der Benutzer muss das Recht permissionGlobalExport oder permissionRead für die zur Version gehörende Datei haben.

Beispiel:

```
CM>content withId 25687 getExportBlobForFrame leftFrame
(Der Export-Blob des Frames).
```

Beispiel für einen entsprechenden obj-Befehl:

```
CM>obj withId 24235 editedContent getExportBlobForFrame leftFrame
(Der Export-Blob des Frames).
```

content contentRef load

Verfügbar für: Content Management Server

Aufgabe: Lädt den Hauptinhalt einer Version und setzt optional ihre Dateinamenserweiterung.

Zusatzinformationen:

- Einer der Parameter blob, blob.plain, blob.base64 und blob.stream muss angegeben werden.

Syntax:

content withId *contentId* load {*contentAttr value*}

Funktionsparameter:

- *contentAttr* bezeichnet einen für das Laden des Inhalts notwendigen Parameter. Folgende Namen sind hier erlaubt:
  - blob, blob.plain, blob.base64 oder blob.stream gibt an, dass *value* den zu importierenden Inhalt in der richtigen Kodierung enthält bzw. dass *value* ein Streaming-Ticket enthält, unter dem der zu importierende Inhalt hochgeladen wurde. Werden mehrere dieser Parameter angegeben, so ist undefiniert, welcher ausgewertet wird.
  - charset: der Zeichensatz des Inhalts bei Dateien, die keine Bilder sind. Wird er nicht angegeben, so wird der Zeichensatz aus der Benutzereinstellung charset verwendet. Der Content Manager konvertiert den Inhalt nach UTF-8. Die Liste der verfügbaren Zeichensätze kann mit dem Tcl-Befehl encoding names ermittelt werden.
  - contentType gibt an, dass *value* die Dateiendung bezeichnet.
- *value* gibt den Wert des betreffenden Parameters an.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht permissionWrite für die Datei haben, zu der die Version gehört. Er muss der Bearbeiter der Datei sein.

Beispiel: Eine Datei in den Hauptinhalt einer Version laden:

```
CM>content withId 25687 load blob [loadFile \
/Users/nps/Upload/news.html] contentType html
```

Beispiel für einen entsprechenden `obj`-Befehl:

```
CM>obj withId 242235 editedContent load blob [loadFile \
/Users/nps/Upload/news.html] contentType html
```

content contentRef mget

Verfügbar für: Content Management Server

Aufgabe: Liefert die Werte der angegebenen Versionsfelder.

Syntax:

`content withId` *`contentId`* `mget {`*`contentAttr`*`}`

Funktionsparameter:

- *contentAttr* : der Name eines abzufragenden Versionsfeldes (siehe Versionsfelder).

Rückgabewert bei Erfolg: die Liste der Werte der abgefragten Versionsfelder (stringlist).

Erforderliche Rechte:

- Der Benutzer muss das Recht *permissionRead* für die Datei haben, zu der die Version gehört.
- Um den Wert des Feldes *exportBlob* abzufragen, genügt auch das Recht *permissionGlobalExport*.

Beispiel:

```
CM>content withId 25687 mget lastChanged validFrom
20101221174909 20101216230000
```

Beispiel für einen entsprechenden `obj`-Befehl:

```
CM>obj withId 242235 editedContent mget lastChanged validFrom
20101221174909 20101216230000
```

content contentRef resolveRefs

Verfügbar für: Content Management Server

Aufgabe: Löst die im Hauptinhalt und in HTML-Feldern der betreffenden Version enthaltenen URLs auf und ordnet ihnen entsprechende Links zu.

Syntax:

`content withId` *`contentId`* `resolveRefs`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss Bearbeiter der Version sein und das Recht `permissionWrite` für die Datei haben, zu der die Version gehört.

Beispiel:

```
CM>content withId 25687 resolveRefs
```

Beispiel für einen entsprechenden `obj`-Befehl:

```
CM>obj withId 242235 editedContent resolveRefs
```

content contentRef set

Verfügbar für: Content Management Server

Aufgabe: Setzt die angegebenen Versionsfelder auf die entsprechenden Werte.

Syntax:

`content withId` *`contentId`* `set {`*`contentAttr value`*`}`

Funktionsparameter:

- *`contentAttr`* bezeichnet das Versionsfeld, dessen Wert gesetzt werden soll.
- *`value`* ist der Wert des zu setzenden Feldes.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss Bearbeiter der Version sein und das Recht `permissionWrite` für die Datei haben, zu der die Version gehört.

Beispiel:

```
CM>content withId 25687 set title {News-Liste}
```

Beispiel für einen entsprechenden `obj`-Befehl:

```
CM>obj withId 242235 editedContent set title {News-Liste}
```

4.5 group (Benutzergruppen)

4.5.1 Benutzergruppen-Parameter

Parameter	Typ	Erklärung	get	set	create	descr

displayTitle	string	Der in der HTML-Benutzerschnittstelle angezeigte Titel der Gruppe. Dieser Titel ist eine Kombination aus dem Namen und dem vollen Namen.	•			
getKeys	stringlist	Liste der mit `get` abfragbaren Parameter	•			
globalPermissions	stringlist	Liste der globalen Rechte	•	•	•	•
name	string	Name der Gruppe	•		•	•
owner	string	Verwalter der Gruppe	•	•	•	•
realName	string	Voller Name der Gruppe	•	•	•	•
setKeys	stringlist	Liste der mit `set` abfragbaren Parameter	•			
users	stringlist	Liste der Benutzer, die Mitglied der Gruppe sind	•	•	•	•

Verwalter einer Benutzergruppe

Im Parameter *owner* einer Gruppe ist der Name des Verwalters der Gruppe (ein Benutzer- oder Gruppenname) gespeichert. Der Verwalter darf die Parameter und Rechte einer Gruppe ändern. Die Wirkungsweise des Gruppen-Parameters *owner* ist analog zu der des User-Parameters *owner* (siehe Verwalter eines Benutzers).

4.5.2 Benutzergruppenbefehle

group create

Verfügbar für: Content Management Server

Aufgabe: Legt eine neue Benutzergruppe mit den angegebenen Parameterwerten an.

Syntax:

`group create {`*`parameter value`*`}`

Zusatzinformationen:

- Bei der Erzeugung der Gruppe wird der `owner` auf den angemeldeten Benutzer gesetzt, falls `owner` beim Aufruf von `create` nicht als Parameter angegeben wurde.
- Der Name der Gruppe kann nach der Erzeugung nicht geändert werden.

Funktionsparameter:

- *`parameter`* spezifiziert den Namen des Gruppenparameters, dessen Wert bei der Erzeugung der Gruppe gesetzt werden soll. Der Parameter `name` muss angegeben werden.
- *`value`* ist der zu setzende Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: der Name der neuen Gruppe (string).

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalUserEdit` haben.

Beispiel:

```
CM>group create name editors realName Editoren
editors
```

group list

Verfügbar für: Content Management Server

Aufgabe: Listet die Namen sämtlicher verwalteter Benutzergruppen auf.

Syntax:

`group list`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: Liste der Namen der Gruppen (stringlist).

Erforderliche Rechte: keine.

Beispiel:

```
CM>group list
admins editors news_editors online_editors
```

group where

Verfügbar für: Content Management Server

Aufgabe: Listet die Namen sämtlicher verwalteter Gruppen auf, in deren Parameter `name` oder `realName` die angegebene Zeichenkette vorkommt.

Syntax:

`group where {`*`parameter value`*`}`

Funktionsparameter:

- *`parameter`* spezifiziert den Namen des Parameters, nach dessen Wert gefragt werden soll oder legt die Anzahl der Suchergebnisse fest. Folgende Parameter sind erlaubt:
  - `maxResults` gibt an, dass *`value`* eine Zahl enthält, die die maximale Anzahl Suchergebnisse festlegt. Ist diese Zahl kleiner oder gleich null, ist die Anzahl der Suchergebnisse nicht beschränkt.
  - `groupText` gibt an, dass *`value`* die Zeichenkette ist, nach der in den Group-Parametern `name` und `realName` gesucht werden soll.
- *`value`*: Der Wert des jeweiligen Parameters.

Rückgabewert bei Erfolg: Liste der Gruppennamen (stringlist).

Erforderliche Rechte: keine.

Beispiel:

```
CM>group where groupText editors maxResults 20
editors news_editors online_editors
```

group groupRef addUsers

Verfügbar für: Content Management Server

Aufgabe: Nimmt jeden der angegebenen Benutzer in die Gruppe mit dem Namen `name` auf, falls er nicht schon Mitglied ist.

Syntax:

`group withName` *`name`* `addUsers {`*`login`*`}`

Funktionsparameter:

- *`login`*: Login eines Benutzers.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte:

- Der eingeloggte Benutzer muss Superuser sein, oder
- der eingeloggte Benutzer muss das Recht `permissionGlobalUserEdit` haben und direkter oder indirekter `owner` der Gruppe `name` sein.

Beispiel:

```
CM>group withName editors addUsers katrin volker
```

group groupRef delete

Verfügbar für: Content Management Server

Aufgabe: Löscht die Gruppe mit dem Namen `name` .

Syntax:

`group withName` *`name`* `delete`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte:

- Der eingeloggte Benutzer muss Superuser sein, oder
- der eingeloggte Benutzer muss das Recht `permissionGlobalUserEdit` haben und direkter oder indirekter `owner` der Gruppe `name` sein.

Beispiel:

```
CM>group withName unwanted delete
```

group groupRef description

Verfügbar für: Content Management Server

Aufgabe: Liefert eine String-Repräsentation der Daten der Gruppe mit dem Namen `name`.

Zusatzinformationen: die String-Repräsentation wird im Property-List-Format eines Dictionarys formatiert.

Syntax:

`group withName` *`name`* `description`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: String-Repräsentation der Gruppen-Daten (string).

Erforderliche Rechte: keine.

Beispiel:

```
CM>group withName editors description
{
  globalPermissions = (
  );
  name = editors;
  owner = root;
  realName = Editoren;
  users = (
    katrin,
    volker
  );
}
```

group groupRef get

Verfügbar für: Content Management Server

Aufgabe: Liefert den Wert des angegebenen Gruppenparameters der Gruppe mit dem Namen *name*.

Syntax:

`group withName` *`name`* `get` *`paramName`*

Funktionsparameter:

- *paramName* spezifiziert den Namen des Parameters, dessen Wert gesucht wird.

Rückgabewert bei Erfolg: Wert des angegebenen Parameters (string).

Erforderliche Rechte: keine.

Beispiel:

```
CM>group withName editors get realName
Editors
```

group groupRef grantGlobalPermissions

Verfügbar für: Content Management Server

Aufgabe: Erteilt der Gruppe mit dem Namen `name` die angegebenen Rechte.

Syntax:

`group withName` *`name`* `grantGlobalPermissions {`*`permission`*`}`

Funktionsparameter:

- *`permission`* ist die Bezeichnung eines globalen Rechts.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte:

- Der eingeloggte Benutzer muss Superuser sein, oder
- der eingeloggte Benutzer muss das Recht `permissionGlobalUserEdit` haben und direkter oder indirekter `owner` der Gruppe `name` sein.

Beispiel:

```
CM>group withName admins grantGlobalPermissions permissionGlobalRTCEdit
```

group groupRef hasGlobalPermission

Verfügbar für: Content Management Server

Aufgabe: Prüft, ob die Benutzergruppe mit dem Namen `name` das angegebene Recht hat.

Zusatzinformationen: Eine Gruppe hat das Recht *`permission`* genau dann, wenn ihr dieses Recht erteilt wurde. Es reicht nicht aus, dass ein Mitglied der Gruppe das Recht hat.

Syntax:

`group withName` *`name`* `hasGlobalPermission` *`permission`*

Funktionsparameter:

- *`permission`* bezeichnet ein globales Recht.

Rückgabewert bei Erfolg: 1, wenn die Gruppe das Recht hat, andernfalls 0.

Erforderliche Rechte: keine.

Beispiel:

```
CM>group withName admins hasGlobalPermission permissionGlobalRTCEdit
1
```

group groupRef mget

Verfügbar für: Content Management Server

Aufgabe: Liefert die Werte der angegebenen Parameter der Gruppe mit dem Namen `name`.

Syntax:

`group withName` *`name`* `mget {`*`paramName`*`}`

Funktionsparameter:

- *`paramName`* spezifiziert den Namen eines Gruppen-Parameters, dessen Wert gesucht wird.

Rückgabewert bei Erfolg: Liste der Parameterwerte (stringlist)

Erforderliche Rechte: keine.

Beispiel:

```
CM>group withName editors mget realName users
Editoren {katrin volker}
```

group groupRef removeUsers

Verfügbar für: Content Management Server

Aufgabe: Entfernt die Benutzer mit den angegebenen Logins aus der Gruppe mit dem Namen `name`.

Zusatzinformationen: Ein Benutzer kann nicht aus seiner Standardgruppe entfernt werden.

Syntax:

`group withName` *`name`* `removeUsers {`*`login`*`}`

Funktionsparameter:

- *`login`* Login eines Benutzers, der aus der Gruppe entfernt werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte:

- Der eingeloggte Benutzer muss Superuser sein, oder
- der eingeloggte Benutzer muss das Recht `permissionGlobalUserEdit` haben und direkter oder indirekter `owner` der Gruppe `name` sein.

Beispiel:

```
CM>group withName admins removeUsers stefan
```

group groupRef revokeGlobalPermissions

Verfügbar für: Content Management Server

Aufgabe: Entzieht der Gruppe mit dem Namen `name` die angegebenen Rechte.

Syntax:

`group withName name revokeGlobalPermissions {permission}`

Funktionsparameter:

- *permission* ist die Bezeichnung eines globalen Rechts.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte:

- Der eingeloggte Benutzer muss Superuser sein, oder
- der eingeloggte Benutzer muss das Recht `permissionGlobalUserEdit` haben und direkter oder indirekter `owner` der Gruppe `name` sein.

Beispiel:

```
CM>group withName editors revokeGlobalPermissions permissionGlobalRTCEdit
```

group groupRef set

Verfügbar für: Content Management Server

Aufgabe: Setzt in der Gruppe mit dem Namen `name` die angegebenen Parameter auf die spezifizierten Werte.

Syntax:

`group withName name set {paramName value}`

Funktionsparameter:

- *paramName* spezifiziert den Namen des Parameters, dessen Wert gesetzt werden soll.
- *value* ist der zu setzende Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte:

- Der eingeloggte Benutzer muss Superuser sein, oder
- der eingeloggte Benutzer muss das Recht `permissionGlobalUserEdit` haben und direkter oder indirekter `owner` der Gruppe `name` sein.

Beispiel:

```
CM>group withName editors set realName Editoren
```

4.6 incrExport (Inkrementeller Export)

Der inkrementelle Export ist ein Verfahren, mit dem ein Webserver mit dem Content Manager automatisch synchronisiert werden kann, so dass die Besucher Ihrer Website die gegenwärtig verfügbaren Inhalte abrufen können. Zu diesem Zweck ist es erforderlich, auf der Seite des Webservers

die Template Engine einzusetzen, es sei denn, die Inhalte werden mit dem Rails Connector für Infopark CMS Fiona ausgeliefert.

Der Content Management Server übergibt der Template Engine über das XML-Interface laufend aktualisierte Dateiversionen und weist sie in regelmäßigen, konfigurierbaren Abständen an, diese zu exportieren und live zu schalten. Aktualisierte Daten wie auch Anweisungen werden in Form so genannter Update-Records vom Content Management Server zur Template Engine übertragen.

Die für den inkrementellen Export zuständigen Konfigurationswerte werden mit dem Systemkonfigurationseintrag `export` festgelegt. Insbesondere lässt sich mit dem Wert `incrementalUpdate.isActive` steuern, welche Aktion der Content Manager ausführt, wenn Contents oder andere Daten wie Channels aktualisiert werden. Hat `isActive` den Wert `true`, so werden sämtliche Änderungen zur Template Engine übertragen. Hat `isActive` den Wert `false`, so werden Änderungen nicht protokolliert.

Um die `incrExport` -Befehle über die Tcl-Schnittstelle verwenden zu können, muss man das Recht `permissionGlobalExport` haben oder ein Superuser sein.

4.6.1 IncrExport-Parameter

Parameter	Typ	Erklärung	get
updateRecordCount	string	Liefert die Anzahl der noch nicht zur Template Engine übertragenen Update-Records.	•
updateRecords	stringlist	liefert im Modus (`mode`) `active` eine Liste mit Update-Records. Jeder Update-Record ist selbst eine Liste, die aus zwei Elementen besteht, `updateRecordId` und `updateType` . Im Modus `suspended` liefert `updateRecords` die leere Liste und im Modus `off` den Fehlertext *incremental Export is turned off* . Die zurück gegebenen Update-Records werden nicht aus der Liste der Update-Records gestrichen. Dies muss explizit mit dem Befehl `incrExport removeUpdateRecords` ausgelöst werden.	•
mode	string	Liefert den Status des inkrementellen Exports, `on` oder `off`. Der Modus kann mit dem Systemkonfigurationseintrag `export.incrementalUpdate.isActive` eingestellt werden.	•
getKeys	stringlist	Liste der mit `get` abfragbaren `incrExport`-Parameter	•

Es gibt die folgenden Update-Record-Typen:

Update-Record-Typ	Bedeutung	Übertragung
1	Datei wurde angelegt	CM -> TE
2	Datei wurde geändert	CM -> TE
3	Datei wurde gelöscht	CM -> TE
4	Version wurde geändert	CM -> TE

5	Version wurde gelöscht	CM -> TE
6	Reset wurde begonnen	CM -> TE
7	Reset wurde beendet	CM -> TE
8	Mit der Veröffentlichung beginnen	CM -> TE
9	Channel wurde gelöscht	CM -> TE -> PM
10	Datei wurde verschoben oder geändert	CM -> TE

4.6.2 IncrExport-Befehle

incrExport get

Verfügbar für: Content Management Server

Aufgabe: Liefert den Wert des angegebenen incrExport-Parameters.

Syntax:

`incrExport get` *`paramName`*

Funktionsparameter:

- *`paramName`* spezifiziert den Namen des Parameters, dessen Wert zurückgegeben werden soll.

Rückgabewert bei Erfolg: Der Wert des angegebenen Parameters (string).

Erforderliche Rechte: Der Benutzer muss Superuser sein oder das Recht `permissionGlobalExport` haben.

Beispiel:

```
CM>incrExport get mode
suspended
```

incrExport getUpdateData

Verfügbar für: Content Management Server

Aufgabe: Liefert alle für den Export notwendigen Daten der Datei, die zum Update-Record mit der ID `id` gehört. Die Daten werden im XML-Format so geliefert, dass sie von der Template Engine verarbeitet werden können.

Syntax:

`incrExport getUpdateData updateRecordId` *`id`*

Funktionsparameter:

- *`id`* spezifiziert die ID des Update-Records, der auf die Datei verweist, deren Daten geliefert werden sollen.

Rückgabewert bei Erfolg: das entsprechende XML-Dokument, das die Daten enthält (string). Das Wurzelelement des Dokuments ist `updateData`.

Erforderliche Rechte: Der Benutzer muss Superuser sein oder das Recht `permissionGlobalExport` haben.

Beispiel:

```
CM>incrExport getUpdateData updateRecordId 4711
(die Update-Daten)
```

incrExport mget

Verfügbar für: Content Management Server

Aufgabe: Liefert die Werte der angegebenen incrExport-Parameter.

Syntax:

`incrExport mget {`*`paramName`*`}`

Funktionsparameter:

- *`paramName`* spezifiziert den Namen eines Parameters, dessen Wert zurückgegeben werden soll.

Rückgabewert bei Erfolg: Die Werte der angegebenen Parameter (string).

Erforderliche Rechte: Der Benutzer muss Superuser sein oder das Recht `permissionGlobalExport` haben.

Beispiel:

```
CM>incrExport mget mode updateRecordCount
active 481
```

incrExport removeUpdateRecords

Verfügbar für: Content Management Server

Aufgabe: Löscht die Update-Records mit den angegebenen IDs sowie alle Detail-Daten wie die zu den Update-Records gehörenden XML-Dateien.

Syntax:

`incrExport removeUpdateRecords ({`*`id`*`}) | all`

Funktionsparameter:

- *`id`* spezifiziert die ID eines Update-Records. Ist statt einer oder mehrerer IDs die Zeichenkette `all` angegeben, so werden alle Update-Records gelöscht.

Rückgabewert bei Erfolg: Die Anzahl der gelöschten Update-Records (number).

Erforderliche Rechte: Der Benutzer muss Superuser sein oder das Recht `permissionGlobalExport` haben.

Beispiel:

```
CM>incrExport removeUpdateRecords 3461 3462 5471
3
```

incrExport reset

Verfügbar für: Content Management Server

Aufgabe: Löscht die interne Tabelle der Update-Records und erzeugt sie neu. Die Zeit, die der Content Manager benötigt, um den Befehl abzuarbeiten, ist proportional zur Anzahl der Dateien.

Zusatzinformationen: **Während dieser Befehl ausgeführt wird, dürfen keine Update-Records erzeugt werden. Der Befehl `incrExport reset` darf nur im Single-Modus verwendet werden und nur ausgeführt werden, wenn mit Sicherheit keine schreibenden Zugriffe auf den Datenbestand stattfinden, da andernfalls die exportierten Daten unvollständig sind oder zerstört werden. Stellen Sie daher sicher, dass während der Ausführung des Befehls der CM nicht als Server läuft und keine anderen CMs im Single-Modus gestartet werden oder laufen.**

Syntax:

```
incrExport reset
```

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss Superuser sein oder das Recht `permissionGlobalExport` haben.

Beispiel:

```
CM>incrExport reset
```

4.7 job (Jobs)

Mit dem *Job*-Befehl lassen sich kundenspezifische Tcl-Skripte einmalig oder wiederholt ausführen.

Der Content Management Server und die Template Engine unterscheiden zwischen System-Jobs und Benutzer-Jobs. System-Jobs sind vordefinierte Routinen, etwa zur regelmäßigen Übertragung von Update-Records zur Template Engine im Rahmen des inkrementellen Live-Server-Updates. Sie können weder gelöscht noch geändert werden. Benutzer-Jobs dagegen können bedarfsweise angelegt, modifiziert und auch wieder gelöscht werden. Beide Job-Kategorien teilen sich einen Namensraum, d. h. zwei Jobs können auch dann nicht den gleichen Namen haben, wenn sie zu unterschiedlichen Kategorien gehören.

Die Jobs einer Kategorie werden zum Ausführungszeitpunkt in die Warteschlange dieser Kategorie aufgenommen, sofern sie nicht bereits aufgenommen wurden. Dadurch ist sicher gestellt, dass ein häufiger auszuführender Job nicht mehrmals in die Warteschlange gestellt wird, wenn ein anderer oder auch dieser Job gerade abgearbeitet wird.

Alle Jobs einer Kategorie werden sequenziell abgearbeitet, da sie sich eine Warteschlange teilen. Dies hat den Vorteil, dass mehrere Jobs auf einmal angestoßen werden können und sich dennoch nicht überschneiden (Initiierung und Ausführung sind asynchron). Die Jobs unterschiedlicher Kategorien werden jedoch parallel abgearbeitet.

Bitte beachten Sie, dass Jobs nur ausgeführt werden, wenn die betreffende CMS-Applikation (CM oder TE) als Server läuft. Wird die Applikation mit dem Kommandozeilenargument `-single` aufgerufen und läuft sie nicht gleichzeitig als Server, so werden Jobs zwar in die Warteschlange gestellt, jedoch erst dann ausgeführt, nachdem der Server wieder gestartet wurde.

Die Ausführung von Jobs wird protokolliert. Je Job legt das System maximal so viele Protokolleinträge an, wie es der Wert von `jobMaxLogLength` im Systemkonfigurationseintrag `tuning` festgelegt.

4.7.1 Job-Parameter

Parameter	Typ	Erklärung	get	set create	descr
category	string	Die Job-Kategorie (`user` oder `system`)	•		•
comment	string	Beschreibung des Jobs	•	•	
displayTitle	string	Der in der HTML-Benutzerschnittstelle angezeigte Titel des Jobs	•		
execLogin	string	Login, unter dem das Skript ausgeführt wird	•	•	•
execPerm	string	Das globale Recht, das ein Benutzer benötigt, um den Job auszuführen, d. h. in die Warteschlange zu stellen. Ist kein Recht angegeben, so dürfen alle Benutzer den Job ausführen.	•	•	
getKeys	stringlist	Liste der mit `get` abfragbaren Job-Parameter	•		
id	number	Die ID des Jobs	•		•
isActive	bool	Gibt an, ob ein Job aktiv (1) oder nicht aktiv (0) ist. Nur aktive Jobs können ausgeführt werden	•	•	•

lastExecEnd	datetime	Zeitpunkt, an dem die letzte Ausführung des Jobs beendet wurde.	•		
lastExecResult	string	Ergebnis der letzten Ausführung des Skripts	•		
lastExecStart	datetime	Zeitpunkt, an dem die letzte Ausführung des Jobs begonnen wurde.	•		
lastLogEntry	stringlist	Der letzte (aktuelle) Protokolleintrag des Jobs. Das Format ist beim Subbefehl `getLogEntry` beschrieben.	•		
lastOutput	string	Die letzte vom Job produzierte Ausgabe.	•		
log	stringlist	Die Liste der IDs der letzten Protokolleinträge des Jobs. Die maximale Anzahl der Einträge ist im Systemkonfigurationseintrag *jobMaxLogLength* definiert.	•		
logEntries	stringlist	Die letzten Protokolleinträge des Jobs, von denen jeder wiederum eine Liste ist. Das Format ist beim Subbefehl `getLogEntry` beschrieben.	•		
name	string	Der Name des Jobs. Namen von Jobs der Kategorie `user` dürfen nicht mit *system* oder einem Unterstrich beginnen.	•	•	•
nextExecStart	datetime	Zeitpunkt, an dem der Job das nächste Mal ausgeführt wird (leer, wenn er gerade läuft oder es keinen nächsten Termin gibt).	•		•
queuePos	number	Wenn der Wert größer 0 ist, gibt er die Position des Jobs in der Warteschlange an, ist er 0, so wird der Job gerade ausgeführt, andernfalls ist der Job nicht in der Warteschlange	•		•
schedule	stringlist	Der Ausführungsplan (Beschreibung siehe unten).	•	•	
script	string	Das Tcl-Skript, das abgearbeitet werden soll, wenn der Job ausgeführt wird (nicht bei Jobs der Kategorie `system`).	•	•	
setKeys	stringlist	Liste der mit `set` setzbaren Job-Parameter	•		
title	string	Der Titel des Jobs	•	•	•

Der Ausführungsplan

Jedem Job ist ein Ausführungsplan (engl. *schedule* = Terminplan) zugeordnet. In diesem Plan sind die (gegebenenfalls wiederkehrenden) Ausführungszeitpunkte des Jobs festgelegt. Jeder Ausführungszeitpunkt besteht aus Angaben über die Jahre (`years`), Monate (`months`), Tage (`days`) oder Wochentage (`weekdays`), die Stunden (`hours`) und die Minuten (`minutes`). Die Angaben müssen nicht vollständig sein, d. h. Bestandteile können weggelassen werden. Der folgende Job wird im Juli 2010 täglich um 6.30 und um 22.30 Uhr ausgeführt:

```
job withName twiceADayInJuly2010 set schedule {
  {minutes 30 hours {6 22} months 7 years 2010}
}
```

Der Ausführungsplan ist eine Liste von Einträgen, von denen jeder wiederum eine Liste ist. Schließlich werden innerhalb eines Eintrags dessen Bestandteile mit den oben aufgeführten Bezeichnern und den dazu gehörenden Werten spezifiziert.

Für die Logik der Wiederholung ist es entscheidend, dass ein fehlender Bestandteil eines Eintrags so interpretiert wird, als wären alle möglichen Werte für diesen Bestandteil angegeben. Führt man beispielsweise nicht die Tage oder Wochentage auf, in denen der Job ausgeführt werden soll, so wird er täglich ausgeführt. Ein leerer Eintrag für einen Ausführungszeitpunkt bewirkt also, dass der Job jede Minute ausgeführt wird, und zwar so lange, bis der Eintrag gelöscht oder geändert wird. Ein einmaliger Ausführungszeitpunkt (ohne Wiederholung) lässt sich definieren, indem für jeden Bestandteil außer `weekdays` genau ein Wert angegeben wird. `days` und `weekdays` werden mit *oder* verknüpft, und das Ergebnis wird mit allen anderen Kategorien mit *und* verknüpft.

Die Stunden werden als Zahlen von 0 bis 23, die Wochentage mit den Zahlen 1 bis 7 angegeben, wobei 1 für Montag, 2 für Dienstag usw. steht.

Alle Einträge in Ausführungsplänen beziehen sich auf die Server-Zeit mit ihrer spezifischen Zeitzone. Überschneiden sich die Ausführungszeiten in Einträgen, so hat der in der Liste weiter vorn stehende Eintrag Vorrang.

4.7.2 Job-Befehle

job create

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Erzeugt einen neuen Job mit den angegebenen Parametern.

Syntax:

`job create {`*`parameter value`*`}`

Funktionsparameter:

- *`parameter`* spezifiziert den Namen eines Parameters, der bei `create` angegeben werden kann. Wird der Parameter `execLogin` nicht angegeben, so wird als dessen Wert das Login des aktuellen Benutzers eingetragen. Der Parameter `name` muss angegeben werden.
- *`value`* ist der Wert des mit *`parameter`* angegebenen Parameters.

Rückgabewert bei Erfolg: Der Name des neuen Jobs.

Erforderliche Rechte (nur CM): Der Benutzer muss Superuser sein.

Zusatzinformationen: Es ist empfehlenswert, mindestens ein globales Recht (beispielsweise `permissionGlobalJobExec`) zu definieren und neuen Jobs als erforderliches Ausführungsrecht zuzuweisen. Andernfalls können die Jobs von allen Benutzern ausgeführt werden.

Beispiel:

```
CM>job create name myJob title "Rootpub exportieren" \
execPerm permissionGlobalJobExec \
script "obj root exportSubtree filePrefix /tmp"
myJob
```

job list

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Liefert die Namen aller Jobs.

Syntax:

```
job list
```

Funktionsparameter: keine.

Rückgabewert bei Erfolg: Die Liste der Namen aller Jobs (stringlist).

Erforderliche Rechte (nur CM): keine.

Beispiel:

```
CM>job list
testJob adminExportFailureNotification systemSwitch systemTransferUpdates
```

job listQueue

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Liefert die IDs aller Jobs in der Ausführungswarteschlange der angegebenen Job-Kategorie, aufsteigend sortiert nach dem Job-Parameter `queuePos`.

Syntax:

`job listQueue [`*`category`*`]`

Funktionsparameter:

- *`category`* spezifiziert den Namen einer Job-Kategorie (`user` oder `system`). Voreinstellung: `user`.

Rückgabewert bei Erfolg: Die Liste der Namen der Jobs in der Warteschlange (stringlist).

Erforderliche Rechte (nur CM): keine.

Beispiel:

```
CM>job listQueue system
systemSwitch systemTransferUpdates
```

job where

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Sucht nach Jobs, bei denen alle Bedingungen, die als Parameter-Werte-Paare angegeben wurden, erfüllt sind.

Syntax:

```
job where {parameter value}
```

Funktionsparameter:

- *parameter* ist einer der folgenden Bezeichner:
  - maxResults gibt an, dass *value* eine Zahl enthält, die die maximale Anzahl Suchergebnisse festlegt. Ist diese Zahl kleiner oder gleich null, ist die Anzahl der Suchergebnisse nicht beschränkt.
  - category: Die Jobs, bei denen der Parameter category dem Wert *value* entspricht, werden zurückgegeben.
  - comment: Die Jobs, bei denen der Parameter comment den Wert *value* enthält, werden zurückgegeben.
  - execLogin: Liefert die Jobs, bei denen das Skript unter dem Login *value* ausgeführt wird.
  - isActive: Liefert die Jobs, bei denen der Parameter isActive den Wert *value* enthält.
  - isQueued: Ist *value* ungleich 0, so werden alle Jobs geliefert, bei denen queuePos größer oder gleich null ist. Andernfalls werden die Jobs geliefert, bei denen queuePos kleiner null ist.
  - name: Die Jobs, bei denen der Parameter name dem Wert *value* entspricht, werden zurückgegeben.
  - title: Die Jobs, bei denen der Parameter title den Wert *value* enthält, werden zurückgegeben.
- *value* enthält den Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: die Liste der Namen der passenden Job-Einträge (stringlist).

Erforderliche Rechte (nur CM): keine Einschränkungen.

Beispiel:

```
CM>job where isActive 1 execLogin root
rootJob1 rootJob2
```

job jobRef cancel

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Nimmt den angegebenen Job aus der Warteschlange.

Zusatzinformationen: Beim angegebenen Job wird queuePos auf -1 gesetzt. Der gerade ausgeführte Job (queuePos = 0) kann nicht angehalten werden.

Syntax:

job (withName *jobName*) | (withId *jobId*) cancel

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss Superuser sein, das als execLogin spezifizierte Login oder das als Wert des Parameters execPerm angegebene Recht haben.

Beispiel:

```
CM>job withId 3099 cancel
```

job jobRef delete

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Löscht den angegebenen Job aus der Jobliste.

Syntax:

`job (withName *jobName*) | (withId *jobId*) delete`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss Superuser sein.

Beispiel:

```
CM>job withId 2023 delete
```

job jobRef description

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Liefert eine String-Repräsentation des angegebenen Jobs.

Zusatzinformationen: Das Format der String-Repräsentation ist das Property-List-Format eines Dictionarys.

Syntax:

`job (withName *jobName*) | (withId *jobId*) description`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: Repräsentation des Jobs (string).

Erforderliche Rechte (nur CM): keine Einschränkungen.

Beispiel:

```
CM>job withId 3245 description
{
  id = 3245;
  name = someJob;
  category = user;
  queuePos = -1;
  title = "Rootpub exportieren";
  execLogin = root;
  isActive = 1;
}
```

job jobRef exec

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Stellt einen Job in die Ausführungswarteschlange.

Zusatzinformationen: Befindet sich der Job bereits in der Ausführungswarteschlange, so wird er nicht ein zweites Mal in die Warteschlange gestellt.

Syntax:

`job (withName jobName) | (withId jobId) exec`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: Der neue Wert für `queuePos`. Befindet sich der Job bereits in der Warteschlange, so wird der aktuelle Wert von `queuePos` zurückgegeben.

Erforderliche Rechte (nur CM): Der Benutzer muss Superuser sein, das als `execLogin` spezifizierte Login oder das als Wert des Parameters `execPerm` angegebene Recht haben.

Beispiel:

```
CM>job withId 2023 exec
5
```

job jobRef get

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Liefert den Wert des Job-Parameters *paramName*.

Syntax:

`job (withName jobName) | (withId jobId) get paramName`

Funktionsparameter:

- *paramName* spezifiziert den Namen des Job-Parameters, dessen Wert gesucht wird.

Rückgabewert bei Erfolg: Der Wert des angegebenen Job-Parameters.

Erforderliche Rechte (nur CM): keine Einschränkungen

Beispiel:

```
CM>job withId 2245 get title
Rootpub exportieren
```

job jobRef getLogEntry

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Der Befehl gibt einen Protokolleintrag des angegebenen Jobs aus.

Syntax:

`job (withName jobName) | (withId jobId) getLogEntry entryId`

Funktionsparameter:

- *entryId* spezifiziert die ID eines zum betreffenden Job gehörenden Protokolleintrags. Die Liste der Protokoll-IDs liefert der Job-Parameter `log`.

Rückgabewert bei Erfolg: eine Liste mit Name-Wert-Paaren (stringlist).

Erforderliche Rechte (nur CM): keine Einschränkungen.

Beispiel:

```
CM>job withName myJob getLogEntry 3461
entryId 3461 execStart 20021219111651 execEnd 20021219111735 execResult {The Job Result}
```

job jobRef getOutput entryId

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Der Befehl gibt die protokollierte Ausgabe des angegebenen Jobs zurück.

Syntax:

`job (withName` *jobName*`) | (withId` *jobId*`) getOutput` *entryId*

Funktionsparameter:

- *entryId* spezifiziert die ID eines zum betreffenden Job gehörenden Protokolleintrags. Die Liste der Protokoll-IDs liefert der Job-Parameter `log`.

Rückgabewert bei Erfolg: die vom Job erzeugte Ausgabe (string).

Erforderliche Rechte (nur CM): keine Einschränkungen.

Beispiel:

```
CM>job withName myJob getOutput 3461
Output generated by the job.
```

job jobRef mget

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Liefert die Werte eines oder mehrerer Job-Parameter, die jeweils als *paramName* spezifiziert werden.

Syntax:

`job (withName` *jobName*`) | (withId` *jobId*`) mget {`*paramName*`}`

Funktionsparameter:

- *paramName* spezifiziert den Namen eines Job-Parameters, dessen Wert gesucht wird.

Rückgabewert bei Erfolg: Die Werte der angegebenen Job-Parameter.

Erforderliche Rechte (nur CM): keine Einschränkungen

Beispiel:

```
CM>job withId 2245 mget execLogin execPerm
root permissionGlobalJobExec
```

job jobRef set

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Setzt die Werte der angegebenen Job-Parameter auf den jeweils spezifizierten Wert.

Syntax:

`job (withName` *`jobName`*`) | (withId` *`jobId`*`) set {`*`paramName value`*`}`

Funktionsparameter:

- *`paramName`* spezifiziert den Namen eines Job-Parameters, dessen Wert gesetzt werden soll.
- *`value`* spezifiziert den Wert des Job-Parameters, der als *`paramName`* angegeben wurde.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss Superuser sein.

Beispiel:

```
CM>job withId 2245 set execLogin mustermann isActive 1
```

4.8 licenseManager (Lizenzinformationen)

4.8.1 LicenseManager-Parameter

Parameter	Typ	Erklärung	get
clients	stringlist	(Bis Version 6.6.1.) Die Instanzen, bei denen Benutzer angemeldet sind sowie deren Benutzer. Das Ergebnis ist eine Liste, die paarweise einen Instanzennamen sowie die Liste der Benutzer enthält (siehe das Beispiel unten).	•
extensions	stringlist	(Bis Version 6.6.1.) Die Liste der lizenzierten Erweiterungen wie beispielsweise der *Content Service*.	•
getKeys	stringlist	Die Liste der LicenseManager-Parameter, die mit `get` abgefragt werden können.	•
idLimit	number	Liefert bei ID-beschränkten Lizenzen die größtmögliche ID.	•
license	string	Gibt die Lizenzierungsdaten als XML-Element zurück.	•
licenseExpirationDate	string	Gibt das Datum zurück, an dem die Lizenz ausläuft.	•
licenseType	string	Gibt je nach Art der installierten Lizenz *demo* oder *full* zurück.	•

loginCount	string	Gibt die Anzahl der Benutzer zurück, die gegenwärtig am System angemeldet sind.	•
logins	stringlist	Gibt eine Liste von Dictionaries zurück. Jedes Dictionary bezieht sich auf einen angemeldeten Benutzer und enthält zwei Werte, *login* und *timeStamp*. *login* gibt das verwendete Interface, die Session und das Login an (beispielsweise `T(1069544507) root` wobei T=tcl, X=XML, J=Jobs). Der Wert von `timeStamp` spezifiziert Datum und Uhrzeit des letzten Zugriffs auf das System durch den Benutzer.	•
maxConcurrentClients	string	(Bis Version 6.6.1.) Die maximale Anzahl Instanzen, bei denen sich Benutzer gleichzeitig anmelden können. Diese Zahl bezieht sich nicht auf einen einzelnen Benutzer, sondern auf alle Benutzer zusammen.	•
maxConcurrentUsers	string	Die maximale Anzahl der Benutzer, die gleichzeitig am System angemeldet sein dürfen.	•
remainingIds	number	Liefert bei ID-beschränkten Lizenzen die Anzahl der verbleibenden IDs.	•

4.8.2 LicenseManager-Befehle

licenseManager checkLicense

Verfügbar für: Content Management Server, Template Engine (bis Version 6.6.1)

Aufgabe: Das Kommando prüft, ob für die angegebene Applikation in der angegebenen Version eine Lizenz vorhanden ist.

Syntax:

`licenseManager checkLicense` *`appName version`*

Funktionsparameter:

- `appName` spezifiziert den Namen der Applikation, deren Vorhandensein in der Lizenzdatei `license.xml` geprüft werden soll. Der Name muss genau so angegeben werden wie in der Lizenzdatei.
- `version` Die Versionszeichenkette, die mit der Versionsnummer in der Lizenzdatei verglichen wird.

Rückgabewert bei Erfolg: Ist die Applikation `appName` in der Lizenzdatei aufgeführt und beginnt die Versionsnummer in der Lizenzdatei mit `version`, so wird der leere Wert zurückgegeben. Andernfalls wird eine entsprechende Fehlermeldung ausgegeben.

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>licenseManager checkLicense CM 5
```

licenseManager get

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Liefert den Wert des `licenseManager`-Parameters *`paramName`*.

Syntax:

`licenseManager get` *`paramName`*

Funktionsparameter:

- *`paramName`* spezifiziert den Namen des `licenseManager`-Parameters, dessen Wert gesucht wird.

Rückgabewert bei Erfolg: Der Wert des angegebenen `licenseManager`-Parameters.

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>licenseManager get loginCount
15
CM>licenseManager get clients
default {root designer} myinstance {editor designer chiefeditor}
```

4.9 link (Verweise)

4.9.1 Link-Parameter

Parameter	Typ	Erklärung	Anwend.	get	set	descr
attributeName	string	Name des Versionsfeldes, in dem der Link steht (nur bei freien Links).	CM, TE	•	•	•
canHaveAnchor	bool	Gibt an, ob der Link ein Einsprungziel in der Zieldatei haben kann	CM, TE	•		
canHaveTarget	bool	Gibt an, ob der Link ein Frame-Target haben kann, in dem die Zieldatei dargestellt wird.	CM, TE	•		
destination	string	Datei-ID der Zieldatei (bei internen Links)	CM, TE	•		•
destinationUrl	string	die URL, die der Link repräsentiert	CM, TE	•	•	•

displayTitle	string	Der in der HTML-Benutzerschnittstelle angezeigte Titel des Links. Hat der Link keinen Titel, so wird bei internen Links der Titel der Zieldatei geliefert.	CM, TE	•		•
expectedPath	string	Pfad zur referenzierten Datei in der Ordnerhierarchie (bei internen Links)	CM, TE	•		•
getKeys	stringlist	Liste der mit `get` abfragbaren Linkparameter	CM, TE	•		
id	string	die ID des Links	CM, TE	•		•
isComplete	bool	gibt an, ob der Link aufgelöst (resolved) ist	CM, TE	•		•
isContextLink	bool	Gibt an, ob der Link ein Kontextlink ist, d. h. in einer NPSOBJ-context-Anweisung vorkommt.	CM, TE	•		•
isDynamicLink	bool	Gibt an, ob der Link ein dynamischer Link ist, d. h. in einer NPSOBJ-insertvalue-dynamiclink-Anweisung vorkommt.	CM, TE	•		•
isExternalLink	bool	Gibt an, ob der Link ein externer Link ist	CM, TE	•		•
isFreeLink	bool	Gibt an, ob der Link ein freier Link, d. h. einem linklist-Feld zugeordnet ist.	CM, TE	•		•
isIncludeLink	bool	Gibt an, ob der Haupttext des Linkziels in die Quelldatei eingefügt werden soll (`<NPSOBJ includetext="...">`).	CM, TE	•		•
isInlineReferenceLink	bool	Gibt an, ob es sich um einen Inline Reference Link handelt	CM, TE	•		•
isLinkFromCommittedContent	bool	Gibt an, ob sich der Link in der eingereichten Version befindet.	CM	•		
isLinkFromEditedContent	bool	Gibt an, ob sich der Link in der Arbeitsversion befindet.	CM	•		
isLinkFromReleasedContent	bool	Gibt an, ob sich der Link in der freigegebenen Version befindet.	CM	•		
isWritable	bool	Gibt an, ob der eingeloggte Benutzer den Link bearbeiten kann. Dies trifft zu, wenn sich der Link in der Arbeitsversion befindet und der Benutzer das Datei-Schreibrecht hat.	CM	•		•

position (ab Version 6.6.1)	integer	Wenn der Link in einer Linkliste enthalten ist, fungiert `position` als Sortierwert, der die Position des Links in der Linkliste bestimmt. Der Wert kann frei vergeben werden, daher müssen zur Sortierung der Linkliste die Positionen aller darin enthaltenen Links untersucht und ggf. geändert werden.	CM, TE	•	•	
setKeys	stringlist	Liste der mit `set` setzbaren Linkparameter	CM	•		
source	string	die ID der Datei, die den Link enthält	CM, TE	•		•
sourceContent	string	Die ID der Dateiversion, die den Link enthält.	CM	•		•
sourcetagAttribute	string	Der Name des HTML-Tag-Attributs, in dem der Link definiert ist.	CM, TE	•		•
sourcetagName	string	Der Name des HTML-Tags, in dem der Link definiert ist.	CM, TE	•		•
target	string	Das Ziel-Frame oder -Fenster	CM, TE	•	•	•
title	string	Der Titel des Links	CM, TE	•	•	•

4.9.2 Link-Befehle

link list

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Der Befehl gibt eine Liste der IDs sämtlicher Links aus.

Syntax:

```
link list
```

Funktionsparameter: keine.

Rückgabewert bei Erfolg: Liste der IDs der Links (stringlist).

Erforderliche Rechte (nur CM): Der Benutzer muss Superuser sein.

Beispiel:

```
CM>link list
23585 32914 42739 81203
```

link linkRef delete

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Den Link *linkId* löschen, wenn es ein freier Link ist, der sich in einer Arbeitsversion befindet.

Syntax:

`link withId linkId delete`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss der Bearbeiter der Datei sein, in dessen Arbeitsversion sich der Link befindet. Dies setzt voraus, dass er für die Datei das Recht `permissionWrite` hat.

Beispiel:

```
CM>link withId 4273.12.9 delete
```

link linkRef description

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Liefert eine String-Repräsentation der Daten des Links *linkId*.

Zusatzinformationen: Das Format der String-Repräsentation ist das Property-List-Format eines Dictionarys.

Syntax:

`link withId linkId description`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: Repräsentation des Links (string).

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionRead` für die Datei haben, in deren Version der Link definiert ist.

Beispiel:

```
CM>link withId 4273.12.9 description
{
  destination = 6518;
  destinationUrl = /index.html;
  displayTitle = untitled;
  id = 4273.12.9;
  isComplete = 1;
  isExternalLink = 0;
  isIncludeLink = 0;
  isInlineReferenceLink = 0;
  isFreeLink = 1;
  isWritable = 1;
  source = 2012;
  sourceContent = 2012.6;
```

```
  target = "_new";
}
```

link linkRef get

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Liefert zu *linkId* den Wert des Linkparameters *paramName*.

Syntax:

`link linkRef get`

Funktionsparameter:

- *paramName* spezifiziert den Namen des Linkparameters, dessen Wert gesucht wird.

Rückgabewert bei Erfolg: Wert des angegebenen Linkparameters (string).

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionRead` für die Datei haben, in der der Link definiert ist.

Beispiel:

```
CM>link withId 4273.22.9 get destinationUrl
/index.html
```

link linkRef mget

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Liefert zum Link *linkId* die Werte der angegebenen Linkparameter.

Syntax:

`link withId linkId mget {paramName}`

Funktionsparameter:

- *paramName* spezifiziert die Namen der Linkparameter, deren jeweiliger Wert gesucht wird.

Rückgabewert bei Erfolg: Liste der Feldwerte (stringlist).

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionRead` für die Datei haben, in der der Link definiert ist.

Beispiel:

```
CM>link withId 4273.22.9 mget destinationUrl source
/index.html 2012
```

link linkRef set

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Setzt bei dem Link *linkid* die spezifizierten Linkparameter auf die angegebenen Werte. Der Link muss sich in einer Arbeitsversion befinden.

Syntax:

`link withId linkid set {paramName value}`

Funktionsparameter:

- *paramName* spezifiziert den Namen des Linkparameters, dessen Wert gesetzt werden soll.
- *value* ist der zu setzende Wert des betreffenden Linkparameters.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss der Bearbeiter der Datei sein, in deren Arbeitsversion sich der Link befindet. Dies setzt voraus, dass er für die Datei das Recht `permissionWrite` hat.

Beispiel:

```
CM>link withId 2084.9.12 set title {Zur Hauptseite}
```

4.10 linkChecker (Link-Überprüfung)

Mit dem `linkChecker`-Befehl kann die Erreichbarkeit externer Links im CMS geprüft werden.

4.10.1 Link-Checker-Parameter

Der Linkchecker-Befehl und seine Parameter sind ab Version 6.5.0 verfügbar.

Parameter	Typ	Erklärung	get
status	string	Der aktuelle Zustand des Link-Checkers: `running` (aktiv) oder `idle` (inaktiv)	•
count	number	Die Anzahl der zu überprüfenden externen Linkziele.	•
checkedCount	number	Die Anzahl der bereits überprüften externen Linkziele seit dem letzten Start.	•
startedAt	string	Der Zeitpunkt des letzten Starts.	•
unreachableUrls	stringlist	Die Liste der Linkziele, die nicht erreichbar sind. Die Liste enthält externe URLs und Pfade deaktivierter Dateien.	•

4.10.2 Link-Checker-Befehle

Der `linkchecker`-Befehl und seine Parameter sind ab Version 6.5.0 verfügbar.

linkChecker get

Verfügbar für: Content Management Server (ab Version 6.5.0)

Aufgabe: Liefert den Wert eines linkChecker-Parameters.

Syntax:

`linkChecker get parameter`

Funktionsparameter:

- *parameter*: der Name des Parameters, nach dessen Wert gefragt wird.

Rückgabewert bei Erfolg: der Wert des Parameters.

Erforderliche Rechte: keine

Beispiel:

```
CM>linkChecker get status
idle
```

linkChecker mget

Verfügbar für: Content Management Server (ab Version 6.5.0)

Aufgabe: Liefert die Werte eines oder mehrerer linkChecker-Parameter.

Syntax:

`linkChecker mget {parameter}`

Funktionsparameter:

- *parameter*: der Name eines Parameters, nach dessen Wert gefragt wird.

Rückgabewert bei Erfolg: die Werte der angegebenen Parameter (stringlist).

Erforderliche Rechte: keine

Beispiel:

```
CM>linkChecker mget status startedAt
running 20110601181500
```

linkChecker start

Verfügbar für: Content Management Server (ab Version 6.5.0)

Aufgabe: Das Kommando startet die Prüfung aller externen Links auf Erreichbarkeit. Der Befehl wird im Hintergrund ausgeführt und kehrt daher sofort zurück.

Syntax:

`linkChecker start`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: keine

Beispiel:

```
CM>linkChecker start
```

linkChecker stop

Verfügbar für: Content Management Server (ab Version 6.5.0)

Aufgabe: Das Kommando beendet die Überprüfung aller externen Links auf Erreichbarkeit.

Syntax:

```
linkChecker stop
```

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: keine.

Beispiel:

```
CM>linkChecker stop
```

4.11 logEntry (Protokoll-Einträge)

4.11.1 LogEntry-Parameter

Parameter	Typ	Erklärung	where	deleteWhere
fromDate	string	Das Anfangsdatum des Zeitraums, in dem Log-Einträge gesucht werden.	•	•
logText	string	Der im Kommentar eines Log-Eintrags enthaltene Text.	•	•
logTypes	stringlist	Die gesuchten Log-Typen.	•	•
objectId	string	Die ID der Datei, zu dem Log-Einträge gesucht werden.	•	•
receiver	string	Der Name der Gruppe, die eine Datei durch eine Workflow-Aktion erhalten hat.	•	•
untilDate	string	Das Enddatum des Zeitraums, in dem Log-Einträge gesucht werden.	•	•
userLogin	string	Der Benutzer, zu dem Log-Einträge gesucht werden.	•	•

4.11.2 Log-Typen

Log-Typ	Auslösende Aktion
action_comment	Ein Kommentar wurde zu einer Datei hinzugefügt.
action_create_subobj	Erzeugung einer Datei. Als objectId wird im Eintrag die Datei-ID des Ordners festgehalten, in der die Datei erzeugt wurde.
action_deactivate	Deaktivierung einer Datei.
action_delete_subobj	Löschen einer Datei.
action_admin_obj	Verschieben einer Datei oder Änderung von Dateifeldern.
action_edit_obj	Workflow-Aktion *bearbeiten*.
action_give_obj	Workflow-Aktion *geben*.
action_take_obj	Workflow-Aktion *übernehmen*.
action_forward_obj	Workflow-Aktion *weiterleiten*.
action_commit_obj	Workflow-Aktion *einreichen*.
action_reject_obj	Workflow-Aktion *ablehnen*.
action_revert_obj	Workflow-Aktion *verwerfen*.
action_sign_obj	Workflow-Aktion *unterschreiben*.
action_release_obj	Workflow-Aktion *freigeben*.
action_unrelease_obj	Workflow-Aktion *zurückziehen*.

4.11.3 LogEntry-Befehle

logEntry deleteWhere

Verfügbar für: Content Management Server

Aufgabe: Löscht die auf die Suchanfrage passenden Log-Einträge.

Syntax:

```
logEntry deleteWhere {parameter value}
```

Funktionsparameter:

- *parameter* ist der Name eines Parameters, dessen Werte auf das Vorkommen von *value* untersucht werden sollen.
- *value* (string) enthält den Suchstring für den jeweiligen Parameter.

Rückgabewert bei Erfolg: Anzahl der gelöschten Einträge (number).

Erforderliche Rechte: Der Benutzer muss Superuser sein.

Beispiel:

```
CM>logEntry deleteWhere objectId 2003
```

logEntry where

Verfügbar für: Content Management Server

Aufgabe: Sucht nach Log-Einträgen, bei denen der Wert der angegebenen Parameter den jeweils angegebenen Wert enthält.

Syntax:

`logEntry where {`*`parameter value`*`}`

Funktionsparameter:

- *`parameter`* ist der Name eines Parameters, dessen Werte auf das Vorkommen von *`value`* untersucht werden sollen.
- *`value`* enthält den Suchstring für den betreffenden Parameter.

Rückgabewert bei Erfolg: die Liste der passenden Log-Einträge (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>logEntry where logTypes action_take_obj
{objectId 2012 logType action_take_obj receiver root logTime
20000103143526 logText { } userLogin root}
```

4.12 obj (Dateien)

4.12.1 Dateiparameter

Um die Werte von Dateifeldern auszulesen, benötigt man das Leserecht für die betreffende Datei. Dies gilt nicht für die folgenden Felder, deren Werte ohne jegliche Rechte abgefragt werden können: `id`, `name`, `visibleName`, `path`, `visiblePath`, `isMirror` und `objType`.

Wenn die Spalte *set* markiert ist, kann der Wert gesetzt werden, jedoch nur im Content Management Server. Bei der Template Engine ist nur der lesende Zugriff auf die Werte erlaubt. Versionsspezifische Werte beziehen sich beim Content Manager auf die Arbeitsversion, wenn es sie gibt, andernfalls auf die freigegebene Version. Bei der Template Engine beziehen sich versionsspezifische Werte immer auf die freigegebene Version, weil es in der Template Engine keine Arbeitsversionen gibt.

Feld	Typ	Erklärung	Anwend.	get	set	descr

archivedContentIds	stringlist	Liste der IDs aller archivierten Versionen.	CM	•		
children	stringlist	Liste der IDs der Dateien, die sich in dem Ordner befinden.	CM, TE	•		
committedContentId	number	Die ID der eingereichten Version der Datei.	CM	•		
contentIds	stringlist	Liste der IDs aller Versionen der Datei.	CM	•		
dependencies	string	Liefert zum Zwecke der Fehlersuche die Abhängigkeiten der Datei im Propertylist-Format. Je Eintrag werden aufgeführt: der Abhängigkeitstyp (`depType`), die abhängige Datei (`expObjId`) und die referenzierte Datei (`refObjId`). Die Abhängigkeitstypen bedeuten: 1 - `object` (Datei), 2 - `content` (Version), 3 - `reference`, 4 - `children`, 5 - `usesAll`.	TE	•		
dependingObjectIds	stringlist	Die Liste der Dateien, die von dieser Datei abhängig sind.	TE	•		
dependOnObjectIds	stringlist	Die Liste der Dateien, von denen diese Datei abhängt.	TE	•		
editedContentId	number	Die ID der Arbeitsversion der Datei.	CM	•		
exportMimeType	string	Der Mime-Typ zur Dateiendung der freigegebenen (oder, falls diese nicht existiert, der Arbeits-) Version.	CM, TE	•		
getKeys	stringlist	Liste der mit `get` abfragbaren Dateifelder.	CM, TE	•		
hasChildren	bool	Gibt an, ob die Datei Dateien enthält.	CM, TE	•		•
hasMirrors	bool	(Ab Version 6.5) Gibt an, ob es zu dieser Datei Spiegeldateien gibt.	CM, TE	•		
hasSuperLinks	bool	gibt an, ob auf die Datei verwiesen wird	CM, TE	•		•
id	string	die Datei-ID	CM, TE	•		•
isCommitted	bool	Gibt an, ob die Datei eine eingereichte Version hat	CM	•		•
isEdited	bool	Gibt an, ob die Datei eine Arbeitsversion hat	CM	•		•
isExplicitMirror	bool	(Ab Version 6.5) Gibt an, ob die Datei eine explizite (nicht vom System, sondern einem Benutzer oder Skript angelegte) Spiegeldatei ist.	CM, TE	•		
isExportable	bool	Gibt an, ob die Datei exportierbar ist (prüft, ob es eine zeitlich gültige freigegebene Version gibt, d. h. `validFrom` und `validUntil`).	CM, TE	•		

isMirror	bool	(Ab Version 6.5) Gibt an, ob die Datei eine Spiegeldatei ist.	CM, TE	•		•
isReleased	bool	Gibt an, ob die Datei eine freigegebene Version hat.	CM	•		•
isRoot	bool	Gibt an, ob die Datei der Basisordner ist	CM, TE	•		
mirrors	stringlist	(Ab Version 6.5) Liefert die Liste der IDs aller Spiegeldateien der betreffenden Datei.	CM, TE	•		
name	string	Der Name der Datei	CM, TE	•	•	•
next	string	Die ID der Datei, die in dem gleichen Ordner gemäß der Sortierkriterien der Nachfolger der spezifizierten Datei ist	CM, TE	•		
objClass	string	Der Name der Dateivorlage der Datei	CM, TE	•	•	•
objCount	string	Die Anzahl der Dateien in der Dateihierarchie (ohne implizite Spiegeldateien)	CM, TE	•	•	•
objType	string	Der Dateityp	CM, TE	•		•
objectsToRoot	stringlist	Liste der IDs der Dateien, die auf dem Pfad von der Datei zum Basisordner liegen (jeweils einschließlich)	CM, TE	•		
original	string	(Ab Version 6.5) Wenn die betreffende Datei eine Spiegeldatei ist, die ID der Originaldatei	CM, TE	•		
parent	string	Die ID des Ordners, in dem sich die spezifizierte Datei befindet	CM, TE	•	•	•

path	string	Der Pfad der Datei (gebildet aus den Namen der Dateien in den Ordnern ab dem Basisordner)	CM, TE	•		•
permalink	string	(Ab Version 6.6) Ein dauerhafter URI für diese Datei (Permalink), der auch beim Umbenennen und Verschieben der Datei erhalten bleibt.	CM	•	•	
prefixPath	string	Der Pfad der Datei, bei Ordnern mit einem „/" am Ende	CM, TE	•		
previous	string	Die ID der Datei, die in dem gleichen Ordner gemäß der Sortierkriterien der Vorgänger der spezifizierten Datei ist	CM, TE	•		
releasedContentId	number	Die ID der freigegebenen Version der Datei	CM	•		
releasedVersions	stringlist	Die Liste der IDs aller bereits freigegebenen Versionen der Datei (aktuelle freigegebene Version und alle archivierten Versionen)	CM	•		
reminderComment	string	(Ab Version 6.5) Die Anmerkungen zu der Wiedervorlage der Datei	CM	•		
reminderFrom	string	(Ab Version 6.5) Das Datum der Wiedervorlage der Datei (im 14-stelligen Format)	CM	•		
reminderGroups	stringlist	(Ab Version 6.5) Die Liste der Benutzergruppen, denen die Datei wieder vorgelegt wird	CM	•		
reminderUsers	stringlist	(Ab Version 6.5) Die Liste der Benutzer, denen die Datei wieder vorgelegt wird, ohne die Mitglieder der `reminderGroups`	CM	•		
setKeys	stringlist	Die Liste aller mit `set` setzbaren Dateifelder	CM	•		
sortValue	string	Der Wert, nach dem die Dateien innerhalb des Ordners, zu der sie gehören, sortiert werden. Wird durch die Sortierschlüssel des Ordners bestimmt.	CM, TE	•		
superLinks	stringlist	Die Liste aller IDs der Links, die auf die Datei verweisen	CM, TE	•		
superObjects	stringlist	Die Liste aller IDs der Dateien, die Links auf die Datei enthalten	CM, TE	•		
suppressExport	bool	Gibt an, ob die Datei exportiert wird.	CM, TE	•	•	•

toclist (nur Ordner)	stringlist	Die Liste der IDs der Dateien in einem Ordner, die in einer toclist (`<npsobj list="toclist"></npsobj>`) erscheinen (alle zeitlich gültigen Ordner und Dokumente mit freigegebener Version, bei denen `suppressExport` nicht gesetzt ist)	CM, TE	•		
validControlActionKeys	stringlist	Die Namen der Aktionen, die für die Datei und den jeweiligen Benutzer ausführbar sind - Untermenge aus {`edit, commit, release, unrelease, revert, reject`}	CM	•		
validCreateObjClasses	stringlist	Die Liste der Namen der Dateivorlagen, die der Benutzer beim Anlegen von Dateien im aktuellen Ordner verwenden darf. (Dies entspricht der Liste `validSubObjClasses`, die in der Dateivorlage der Datei spezifiziert sind, für die der Benutzer die erforderlichen Anlegerechte hat.)	CM	•		
validObjClasses	stringlist	Die Liste der Namen der Dateivorlagen, die der Benutzer der Datei zuweisen kann (alle auf den Dateityp passenden `validSubObjClasses` des darüber liegenden Ordners).	CM	•		
validPermissions	stringlist	Die Liste aller für die Datei verfügbaren Rechte.	CM	•		
version	number	Die Revision (Anzahl bisheriger Freigaben) der Datei.	CM, TE	•		•
visibleExportTemplates	stringlist	Die Liste der IDs der Layoutdateien, die von der Datei aus sichtbar sind und für einen Export oder die Vorschau in Frage kommen. Der Wert dieses Parameters hängt auch von der Benutzereinstellung `preferEditedTemplates` ab.	CM	•		
visibleName	string	Der Name der Datei, ergänzt um die Dateiendung (außer bei Ordnern)	CM, TE	•		
visiblePath	string	Der Pfad der Datei, ergänzt um die Dateiendung (bei Ordnern wird „`/index.`*`contentType`*" angehängt)	CM, TE	•		
workflowName	string	Der Name des Workflows, der der Datei zugeordnet ist.	CM, TE	•	•	•

parent.*objAttr*		Der entsprechende Dateifeldwert des Ordners, in dem sich die Datei befindet.	CM, TE	•		
previous.*objAttr*		Der entsprechende Dateifeldwert des Vorgängers im gleichen Ordner	CM, TE	•		
next.*objAttr*		Der entsprechende Dateifeldwert des Nachfolgers im gleichen Ordner	CM, TE	•		
up.*objAttr*		Der entsprechende Feldwert der Datei selbst wird ausgegeben, sofern er nicht leer ist, sonst wird der darüber liegende Ordner nach up.*objAttr* gefragt	CM, TE	•		
permissions.*permissionName*	stringlist	Die Liste der Gruppen, die das angegebene Zugriffsrecht auf die Datei haben.	CM, TE	•		

4.12.2 Dateibefehle

obj contentTypesForObjType

Verfügbar für: Content Management Server

Aufgabe: Liefert die Liste aller Dateiendungen (Dateinamenserweiterungen), die vom angegebenen Dateityp prinzipiell unterstützt werden. Die für eine CMS-Datei tatsächlich erlaubten Dateiendungen können jedoch andere sein, weil sie von der Vorlage der betreffenden Datei abhängen. In Vorlagen können die verfügbaren Dateiendungen mit `validContentTypes` gesetzt werden.

Syntax:

`obj contentTypesForObjType` *`objType`*

Funktionsparameter:

- *`objType`* gibt den Dateityp an, dessen unterstützte Dateiendungen ausgegeben werden sollen.

Rückgabewert bei Erfolg: die Liste der Dateiendungen (stringlist).

Erforderliche Rechte (nur CM): keine Einschränkungen.

Beispiel:

```
CM>obj contentTypesForObjType image
jpg jpeg gif
```

obj get

Verfügbar für: Content Management Server, Template Engine.

Aufgabe: Liefert den Wert des angegebenen, sich auf alle CMS-Dateien beziehenden Parameters.

Syntax:

`obj get parameter`

Funktionsparameter:

- *parameter* spezifiziert den Namen des Parameters, dessen Wert gesucht wird. Dabei sind folgende Namen erlaubt:
  - `dependenciesCount` (nur TE): die Anzahl der Einträge in der Abhängigkeitstabelle.
  - `getKeys`: die Liste der bei diesem Kommando verfügbaren Parameter.
  - `objCount`: die Anzahl der verwalteten Dateien.
  - `objTypes`: die Liste der Dateitypen.
  - `rootPubId`: die ID des Basisordners.

Rückgabewert bei Erfolg: der Wert des entsprechenden Parameters.

Erforderliche Rechte (nur CM): keine Einschränkungen.

Beispiel:

```
CM>obj get objTypes
generic image publication document template
```

obj list

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Der Befehl gibt die Liste der IDs aller CMS-Dateien aus.

Zusatzinformationen: Der Befehl liefert auch die IDs explizit angelegter Spiegeldateien, nicht jedoch die IDs der impliziten (automatisch vom System erzeugten) Spiegeldateien.

Syntax:

`obj list`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die Liste der Dateien (stringlist).

Erforderliche Rechte (nur CM): keine Einschränkungen.

Beispiel:

```
CM>obj list
2001 2003 2012 2033
```

obj search

Verfügbar für: Content Management Server

Aufgabe: Das Kommando sucht unter Verwendung des Search Engine Servers nach Dateien, die auf die übergebene Suchanfrage passen.

Syntax:

`obj search {parameter value}`

Funktionsparameter:

- *parameter* spezifiziert den Namen eines Parameters, dessen Wert als *value* übergeben wird. *parameter* kann sein:
  - `query`: Gibt an, dass *value* eine Suchanfrage in der Syntax der installierten Suchmaschine ist. Die Suchanfrage wird weder vom Content Manager noch vom Search Engine Server auf syntaktische Richtigkeit überprüft. Dieser Parameter ist obligatorisch.
  - `minRelevance` (optional): *value* gibt die minimale Relevanz an, die auf die Suchanfrage passende Dateien haben müssen, um in die Trefferliste aufgenommen zu werden. Der Wert von *value* muss eine ganze Zahl zwischen 0 und 100 (jeweils einschließlich) sein, wobei 0 die niedrigste und 100 die größte mögliche Relevanz ist.
  - `maxDocs` (optional): Gibt an, dass *value* die maximale Anzahl (positive ganze Zahl) der zurückzugebenden Treffer ist.
  - `offsetStart` (optional): Gibt an, dass *value* der Index der ersten vom Search Engine Server zu liefernden Versions-ID ist. *value* muss eine ganze Zahl sein. Der Index der ersten Version im Suchergebnis ist 1.
  - `offsetLength` (optional): Gibt an, dass *value* die Anzahl der vom Search Engine Server zu liefernden Versions-IDs enthält. *value* muss eine positive ganze Zahl enthalten.
  - `parser` (optional): Gibt an, dass *value* den Parser bezeichnet, mit dem Suchanfragen von der Suchmaschine bearbeitet werden sollen. Erlaubte Werte sind `explicit`, `freetext` und `simple` (Voreinstellung). Informationen über die Parser erhalten Sie im Handbuch *Die Verity Search Cartridge*.
  - `collections` (optional): Gibt an, dass *value* eine Liste der Collections enthält, auf die die Suche angewendet werden soll. Ist der Parameter nicht angegeben, so werden alle Collections durchsucht.
  - `returnOption` (optional): Gibt an, dass *value* die Art des Rückgabewertes bezeichnet. *value* kann sein:

    `objects` (Voreinstellung): Es wird eine Liste von Datei-IDs geliefert. Jede ID fasst die von der Suchmaschine gefundenen Versionen zusammen;

    `includeContents`: Es wird eine Liste von Listen geliefert. Jede Subliste enthält als erstes Element die Datei-ID und als weitere Elemente die IDs der Versionen dieser Datei, die auf die Suchanfrage passen.

    `statistics`: Es wird eine Liste mit vier Elementen geliefert, die folgendes enthält: `hits` *Anzahl_Treffer* `searched` *Anzahl_durchsuchter_Contents*.
- *value* ist der Wert des jeweiligen Parameters.

Rückgabewert bei Erfolg: Eine Liste, deren Inhalt vom Wert von `returnOption` abhängt (siehe oben).

Erforderliche Rechte (nur CM): keine Einschränkungen.

Zusatzinformationen:

- Der Search Engine Server liefert stets Versions-IDs, die von diesem Kommando voreingestellt zu Datei-IDs zusammengefasst werden.
- Die Werte von `maxDocs`, `offsetStart` und `offsetLength` beziehen sich auf die Versionen, die vom Search Engine Server im Suchergebnis geliefert werden, und nicht auf die Ergebnisse, die dieses Kommando liefert.

- Das Kommando gibt nur Dateien zurück, auf die der Benutzer lesend zugreifen darf.

Beispiel:

```
CM>obj search query {aktie <#AND> option} returnOption includeContents
{2073 3044.13 21429.9} {4982 6683.1}
```

obj touchAll

Verfügbar für: Template Engine

Aufgabe: Löscht den Cache und die interne Abhängigkeitstabelle. Dies bewirkt, dass beim nächsten Export sämtliche Inhalte neu exportiert werden. Daher sollte dieser Befehl bei umfangreichen Inhalten mit Bedacht eingesetzt werden.

Syntax:

```
obj touchAll
```

Zusatzinformationen: Dieser Befehl ist während des Exports nicht ausführbar.

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Beispiel:

```
TE>obj touchAll
```

obj where

Verfügbar für: Content Management Server, Template Engine

Aufgabe: sucht nach allen IDs von Dateien, die die angegebenen Suchkriterien erfüllen. Dabei werden jedoch nur Dateien ausgegeben, für die der Benutzer das Leserecht hat.

Zusatzinformationen: Der Befehl liefert keine impliziten (automatisch vom System erzeugten) Spiegeldateien.

Syntax:

```
obj where {parameter value}
```

Funktionsparameter:

- *parameter* spezifiziert die Suchkriterien oder schränkt die Trefferliste anderweitig ein. Wird *parameter* nicht angegeben, so werden alle Datei-IDs zurückgegeben. Die folgenden Parameter sind verfügbar:
  - `ids` ist eine Liste von Datei-IDs. Ist der Parameter angegeben, werden nur die Dateien mit den IDs in der als *value* angegebenen Liste durchsucht.
  - `maxResults` gibt die höchstens zurückzugebende Anzahl Treffer an. Der Wert 0 (Voreinstellung) deaktiviert diese Begrenzung. Siehe auch den Systemkonfigurationseintrag

`maxSearchResultSize`. Der Wert dieses Eintrags wird höher priorisiert als der Wert von `maxResults`.

- `condition`: *`value`* ist eine Bedingung, die eine Datei erfüllen muss, um in die Ergebnisliste aufgenommen zu werden. *`value`* ist eine Liste mit drei Elementen, die die zu vergleichende Eigenschaft, den Vergleichsoperator sowie den Wert, mit dem verglichen werden soll, enthält (in dieser Reihenfolge). Die folgende Tabelle enthält alle Kombinationen dieser drei Bestandteile:

Eigenschaft	Operator	Vergleichswert

`name`
`title`
`nameOrTitle`
`objClass`
	`is`
	`contains`
	`startsWith`
		Zeichenkette

`objType`	`is`
	`isOneOf`
		`document`
		`publication`
		`generic`
		`image`
		`template`
		Liste mit Dateitypen (siehe `is`)

`state`	`is`
	`isNot`
	`isOneOf`
		`edited`
		`committed`
		`released`
		Liste mit Workflowzuständen (siehe `is`)

`object`	`is`
	`has`
	`hasNo`
		`inactive (ab Version 6.5.0)`
		`mirror (ab Version 6.5.0)`
		`superLinks`
		`superLinks`

Es können mehrere Bedingungen angegeben werden, die dann implizit mit *UND* verknüpft werden:

```
obj where condition {name contains foo} \
condition {objType isOneOf {document publication}}
```

Wird bei `objType isOneOf` oder `state isOneOf` eine leere Liste angegeben, so ist die Treffermenge leer.

`state isNot` ist die einzige Negation. Diese ist erforderlich, weil Workflowzustände nicht exklusiv sind (eine Datei kann sowohl editiert oder eingereicht als auch freigegeben sein).

Bei `object is inactive` enthält die Treffermenge alle Dateien ohne bzw. ohne zeitlich gültige Version.

Die `obj where`-Bedingungen können auch in der Template Engine verwendet werden. Allerdings gibt es dort nur freigegebene Dateien, weshalb `state` dort das Folgende liefert:

`state is` / `isOneOf` ...

- `released`: die Bedingung wird ignoriert

  - `committed` / `edited`: die Treffermenge ist immer leer
    `state isNot` ...
  - `released`: die Treffermenge ist immer leer
  - `committed` / `edited`: die Bedingung wird ignoriert
    Die Eigenschaft `object` kann in der Template Engine nicht verwendet werden.
- *`value`* enthält den Wert zum entsprechenden Parameter.

Rückgabewert bei Erfolg: die Liste der IDs der passenden Dateien (stringlist).

Erforderliche Rechte (nur CM): keine Einschränkungen.

Beispiele:

```
CM>obj where condition {state isOneOf {edited committed}}
CM>obj where ids {2001 3002} condition {name contains foo}
CM>obj where condition {name is bar} condition {title contains foo}
CM>obj where condition {name is foo} maxResults 20
```

obj objRef addComment

Verfügbar für: Content Management Server

Aufgabe: Fügt einen Kommentar zu einer Datei hinzu. Der Kommentar wird ins Protokoll geschrieben.

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root addComment` *`comment`*

Funktionsparameter:

- *`comment`* spezifiziert den zu protokollierenden Kommentar.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionWrite` oder `permissionRoot` für die angegebene Datei haben.

Beispiel:

```
CM>obj withId 2003 addComment "Externe Links korrigiert."
CM>logEntry where objectId 2003
{objectId 2003 logType action_comment receiver {} logTime
20100622163745 logText {Externe Links korrigiert.} userLogin root}
...
```

obj objRef commit

Verfügbar für: Content Management Server

Aufgabe: Für die angegebene Datei wird die Workflow-Aktion *Einreichen* ausgeführt, wobei die Arbeitsversion in eine eingereichte Version umgewandelt wird, sofern ihr Workflow eine Unterschrift vorsieht. Andernfalls wird die Datei freigegeben.

Syntax:

`obj (withId objId) | (withPath path) | root commit {parameter value}`

Funktionsparameter:

- *parameter* spezifiziert den Namen eines Parameters der Workflowaktion. Folgende Werte sind zulässig:
  - comment gibt an, dass *value* der Kommentar ist, der in den nächsten Task eingetragen wird.
- *value* enthält den Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): der Benutzer muss das Recht permissionWrite für die angegebene Datei haben und ihr Bearbeiter sein oder Administrator der Datei sein.

Beispiel:

```
CM>obj withId 2003 commit
```

obj objRef copy

Verfügbar für: Content Management Server

Aufgabe: Die angegebene Datei wird kopiert und erhält eine Arbeitsversion.

Syntax:

`obj (withId objId) | (withPath path) | root copy {parameter value}`

Zusatzinformationen: Hat die Quelldatei eine freigegebene Version, so wird diese in die neue Arbeitsversion kopiert. Ist keine freigegebene Version vorhanden, so wird versucht, die eingereichte Version zu verwenden. Ist auch dieser nicht vorhanden, wird die Arbeitsversion verwendet. Gibt es auch diese nicht, so erhält die neue Datei eine leere Arbeitsversion.

Funktionsparameter:

- *parameter* spezifiziert den Namen eines Parameters, der für das Kopieren wichtige Daten angibt (beispielsweise den Zielordner). Folgende Werte sind zulässig:
  - parent: gibt an, dass *value* den Pfad oder die ID des Zielordners spezifiziert – fehlt der Parameter, wird als Ziel der Ordner angenommen, in dem sich die Datei befindet.
  - name: gibt an, dass *value* den Namen enthält, den die Dateikopie erhalten soll. Fehlt der Name, wird der Name der zu kopierenden Datei verwendet. Gibt es diesen in dem Zielordner bereits, wird ein neuer Name nach dem Muster *Namex* berechnet, wobei *x* eine Zahl ist.
  - recursive: wenn die betreffende Datei ein Ordner ist, gibt *value* an, ob darin enthaltene Dateien ebenfalls (rekursiv) kopiert werden sollen (YES, TRUE oder 1) oder nicht (NO, FALSE oder 0). Links auf Dateien innerhalb des kopierten Baumes verweisen auf die Kopien, während Links auf Dateien außerhalb des Baumes auf die bisherigen Linkziele zeigen. Voreinstellung: FALSE.
- *value* enthält den Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: die ID der neuen Datei (string).

Erforderliche Rechte (nur CM):

- Der Benutzer muss das Recht permissionRead für die Quelldatei haben.

- Der Benutzer muss das Recht `permissionCreateChildren` im Zielordner haben.
- Der Benutzer muss das in der Vorlage der Zieldatei definierte globale Dateianlegerecht haben.

Beispiel:

```
CM>obj withId 20099 copy parent /news/latestnews name incoming
65811
```

obj objRef create

Verfügbar für: Content Management Server

Aufgabe: Erzeugt eine neue Datei mit einer Arbeitsversion. Die angegebene Datei, in der die neue Datei angelegt werden soll, muss ein Ordner sein.

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root create {`*`parameter value`*`}`

Funktionsparameter:

- *`parameter`* ist der Name eines für die Dateierzeugung wichtigen Parameters. Der Wert des Parameters wird als entsprechender Wert *`value`* angegeben. *`parameter`* kann sein:
  - `objClass`: der Name der Dateivorlage, das der neuen Datei zugeordnet werden soll.
  - `name`: der Name, den die neue Datei erhalten soll. Wird er nicht angegeben, so wird ein Name nach dem Muster *`newx`* erzeugt, wobei *`x`* ein Zähler ist, der von 0 bis 999 zählt.
  - `suppressExport`: gibt an, ob diese Datei exportiert werden soll.
- *`value`* enthält den Wert für den betreffenden Parameter.

Rückgabewert bei Erfolg: die ID der neu erzeugten Datei (string).

Erforderliche Rechte (nur CM):

- Der Benutzer muss im Zielordner das Recht `permissionCreateChildren` haben.
- Der Benutzer muss das in der Vorlage der Zieldatei definierte globale Dateianlegerecht haben.

Beispiel: Einen Ordner *newslist* im Basisordner anlegen:

```
CM>obj root create name newslist objClass publication
4812
```

obj objRef createAndLoad

Verfügbar für: Content Management Server

Aufgabe: Legt eine neue Datei mit einer Arbeitsversion an und lädt den angegebenen Inhalt. Die angegebene Datei, in der die neue Datei angelegt werden soll, muss ein Ordner sein.

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root createAndLoad {`*`parameter value`*`}`

Zusatzinformationen:

- Einer der Parameter `blob, blob.plain, blob.base64`, `blob.stream` oder `file` muss angegeben werden. Werden mehrere dieser Parameter angegeben, so ist undefiniert, welcher ausgewertet wird.
- Der Parameter `contentType` muss angegeben werden.
- Die Angabe der Dateivorlage kann entfallen, wenn eine Dateiendung angegeben wurde, die lediglich in einer einzigen Dateivorlage als `validContentType` eingetragen ist, so dass die Dateivorlage eindeutig ermittelt werden kann.

Funktionsparameter:

- *`parameter`* ist der Name eines für die Dateierzeugung wichtigen Parameters. Der Wert des Parameters wird im entsprechenden *`value`*-Parameter angegeben. *`parameter`* kann sein:
  - `blob`, `blob.plain`, `blob.base64` oder `blob.stream` gibt an, dass *`value`* den zu importierenden Inhalt in der richtigen Kodierung enthält bzw. dass *`value`* ein Streaming-Ticket enthält, unter dem der zu importierende Inhalt hochgeladen wurde. Werden mehrere dieser Parameter angegeben, so ist undefiniert, welcher ausgewertet wird.
  - `charset`: der Zeichensatz des Inhalts bei Dateien, die keine Bilder oder Ressourcen sind. Wird er nicht angegeben, so wird der Zeichensatz aus der Benutzereinstellung *`charset`* verwendet. Der Content Manager konvertiert den Inhalt nach UTF-8. Die Liste der verfügbaren Zeichensätze kann mit dem Tcl-Befehl *`encoding names`* ermittelt werden.
  - `contentType`: die Dateiendung, die der übergebene Inhalt hat.
  - `file`: ein relativer, aus höchstens zwei Komponenten bestehender Pfad zu der richtig kodierten zu importierenden Datei. Der Pfad bezieht sich auf das temporäre Verzeichnis des Benutzers und darf nicht mit dem übergeordneten Verzeichnis beginnen. Dieser Parameter kann nur von Entwicklern sinnvoll eingesetzt werden.
  - `objClass`: der Name der Dateivorlage, das der neuen Datei zugeordnet werden soll.
  - `name`: der Name, den die neue Datei erhalten soll. Wird er nicht angegeben, so wird ein Name erzeugt.
  - `suppressExport`: gibt an, ob die Datei exportiert werden soll.
- *`value`* enthält den Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: die ID der neuen Datei (string).

Erforderliche Rechte (nur CM):

- Der Benutzer muss das Recht `permissionCreateChildren` im Zielordner haben.
- Der Benutzer muss das in der Vorlage des Zielordners gesetzte globale Dateianlegerecht haben.

Beispiel:

```
CM>obj root createAndLoad name news objClass newspub \
blob.base64 [encodeFile /opt/NPS/upload/newslist.html]
news
```

obj objRef createAndLoadXml

Verfügbar für: Content Management Server

Aufgabe: Erzeugt eine neue Datei mit einer Arbeitsversion und lädt den angegebenen XML-Inhalt. Der Befehl kann nur für Ordner ausgeführt werden (weil nur in Ordnern Dateien angelegt werden können).

Syntax:

`obj (withId *objId*) | (withPath *path*) | root createAndLoadXml {*parameter value*}`

Funktionsparameter:

- *parameter* ist der Name eines für die Dateierzeugung wichtigen Parameters. Der Wert des Parameters wird im entsprechenden *value*-Parameter angegeben. *parameter* kann sein:
  - `xmlBlob`: der Inhalt im XML-Format, der in die Version der neuen Datei geladen werden soll.
  - `name`: der Name, den die neue Datei erhalten soll. Wird er nicht angegeben, so wird ein Name erzeugt.
- *value* enthält den Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: die ID der neuen Datei (string).

Erforderliche Rechte (nur CM):

- der Benutzer muss das Recht `permissionCreateChildren` in dem Zielordner haben.
- der Benutzer muss das globale Dateianlegerecht haben, das in der Vorlage der Zieldatei gesetzt wurde.

Beispiel:

```
CM>obj root createAndLoadXml name news xmlBlob [obj withPath /olds releasedContent get
 xmlBlob]
news
```

obj objRef deactivate

Verfügbar für: Content Management Server (ab Version 6.5.0)

Aufgabe: Dieser Befehl deaktiviert die angegebene Datei. Dies geschieht, indem sie zeitlich ungültig gemacht wird.

Syntax:

`obj (withId *objId*) | (withPath *path*) | root deactivate`

Zusatzinformationen: Die Datei erhält den Status `inactive`, der als Suchkriterium beim Befehl `obj where` angegeben werden kann.

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss Superuser sein oder das Recht `permissionWrite` für die angegebene Datei haben.

Beispiel:

```
CM>obj withId 2003 deactivate
```

obj objRef debugExport

Verfügbar für: Content Management Server (ab Version 6.5.0)

Aufgabe: Der Befehl simuliert den Export der Datei mit der angegebenen ID zu dem Zweck, fehlerhaften Code in Layouts ausfindig zu machen. Aus der Ausgabe ist ersichtlich, welche NPSOBJ-Tags aus welchen Quell-Layouts in welcher Reihenfolge ausgewertet werden und welche Fehler bei der Auswertung aufgetreten sind.

Zusatzinformationen:

- Der Befehl liefert Angaben zu den folgenden Fehlern: Nicht geschlossene Tags, ungleiche Anzahl öffnender und schließender Tags, nicht erlaubte Felder in NPSOBJ-Tags, nicht erlaubte Werte von NPSOBJ-Tag-Attributen, lesenden Zugriff auf nicht definierte Namen.
- Der Inhalt von `systemExecute`-Anweisungen wird nicht auf Fehler untersucht und nicht formatiert. Die Ausgabe solcher Anweisungen dagegen wird formatiert.

Syntax:

```
obj withId objectId debugExport
  [templateName templateName]
  [detailed (yes | no)]
  [quoteHtml (yes | no)]
  [errorPrefix errorPrefix]
  [errorSuffix errorSuffix]
  [htmlPrefix htmlPrefix]
  [htmlSuffix htmlSuffix]
  [infoPrefix infoPrefix]
  [infoSuffix infoSuffix]
  [preferEditedTemplates (yes | no)]
  [allowEditedContents (yes | no)]
```

Funktionsparameter:

- *templateName*: Gibt an, mit welcher initialen Layoutdatei der Exporttest durchgeführt werden soll. Ist `templateName` nicht angegeben, wird die benutzerspezifische Standardlayoutdatei verwendet (voreingestellt `mastertemplate`).
- *detailed*: Gibt an, ob sich die Ausgabe nur auf Fehlermeldungen beschränken (`no`) oder ausführlich sein soll (`yes`). Der voreingestellte Wert ist `no`.
- *quoteHtml*: Gibt an, ob die Zeichen <, > und & als HTML-Entities ausgegeben werden sollen (`yes`) oder nicht (`no`). Die Präfix- und Suffix-Parameter sind hiervon nicht betroffen. Der voreingestellte Wert ist `no`.
- *errorPrefix*: Gibt eine Zeichenkette an, die vor jeder Fehlermeldung ausgegeben werden soll. Der voreingestellte Wert ist `"***"`.
- *errorSuffix*: Gibt eine Zeichenkette an, die nach jeder Fehlermeldung ausgegeben werden soll (voreingestellt leer).
- *htmlPrefix*: Gibt eine Zeichenkette an, die vor HTML-Text ausgegeben werden soll. Der voreingestellte Wert ist "<pre>".
- *htmlSuffix*: Gibt eine Zeichenkette an, die nach HTML-Text ausgegeben werden soll. Der voreingestellte Wert ist "</pre>".

- *infoPrefix*: Gibt eine Zeichenkette an, die vor jeder NPSOBJ-Information ausgegeben werden soll (voreingestellt leer).
- *infoSuffix*: Gibt eine Zeichenkette an, die nach jeder NPSOBJ-Information ausgegeben werden soll (voreingestellt leer).
- *preferEditedTemplates*: Gibt an, ob die Arbeitsversionen der Layouts gegenüber den freigegebenen Versionen bevorzugt verwendet werden sollen. Der voreingestellte Wert entspricht dem gleichnamigen Wert aus den Benutzereinstellungen, der auch für die Vorschau verwendet wird.
- *allowEditedContents*: Gibt an, ob auf Arbeitsversionen zurückgegriffen werden darf, wenn freigegebene Versionen nicht verfügbar sind. Der voreingestellte Wert ist `no`.

Rückgabewert bei Erfolg: der formatierte Exportbericht (string).

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalExport` oder `permissionRead` für die angegebene Datei haben.

Beispiel: (Ausgabe stark gekürzt)

```
CM>obj withId 25687 debugExport detailed yes
<HTML>
    <HEAD>
  ...
NPSOBJ insertvalue=var: title
The title
END NPSOBJ insertvalue=var
</TITLE>
    </HEAD>
NPSOBJ insertvalue=var: body
<a ...>Linked text</a>

*** FEHLER [140008] Ein NPSOBJ-Tag enthielt kein Kommandoattribut.
...
```

obj objRef delete

Verfügbar für: Content Management Server

Aufgabe: Löscht die angegebene Datei.

Syntax:

`obj (withId` *objId*`) | (withPath` *path*`) delete`

Zusatzinformationen: Die Datei kann nur gelöscht werden, wenn sie nicht der Basisordner ist, sie keine Dateien enthält und nicht von Links referenziert wird. Links in archivierten Versionen verhindern jedoch nicht, dass eine Datei gelöscht wird. In solchen Links werden alle Informationen über die Zieldatei bis auf `expectedPath` entfernt, und `expectedPath` wird auf den Pfad gesetzt, den die Datei zum Zeitpunkt der Löschung hat.

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss Superuser sein oder das Recht `permissionRoot` für die angegebene Datei haben.

Beispiel:

```
CM>obj withId 87193 delete
```

obj objRef deleteFromIndex

Verfügbar für: Content Management Server

Aufgabe: Löscht bei Einsatz des Search Servers die im Index der Suchmaschine enthaltenen Angaben zur spezifizierten Datei.

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root deleteFromIndex`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionRoot` für die betreffende Datei haben.

Beispiel:

```
CM>obj withId 85392 deleteFromIndex
```

obj objRef description

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Liefert eine String-Repräsentation der Daten der Datei mit der angegebenen ID.

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root description`

Zusatzinformationen: die Repräsentation wird im Property-List-Format eines Dictionarys formatiert (siehe Das Property-List-Format).

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die String-Repräsentation der Datei (string). Beim CM enthalten die Daten auch Informationen über den Workflow-Status, nicht jedoch bei der TE, weil die TE nur freigegebene Versionen kennt.

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionRead` für die angegebene Datei haben.

Beispiel:

```
CM>obj root description

{
  hasChildren = 1;
  hasSuperLinks = 1;
  id = 2001;
  isCommitted = 0;
```

```
  isEdited = 1;
  isReleased = 1;
  name = ROOTPUB;
  objClass = publication;
  objType = publication;
  path = "/";
  suppressExport = 0;
  version = 1;
}
```

obj objRef edit

Verfügbar für: Content Management Server

Aufgabe: Für die angegebene Datei wird eine Arbeitsversion angelegt. Es wird ein Fehler erzeugt, wenn die Datei bereits eine Arbeitsversion hat.

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root edit {`*`parameter value`*`}`

Funktionsparameter:

- *`parameter`* ist der Name eines Parameters, dessen Wert im entsprechenden *`value`*-Parameter angegeben ist. *`parameter`* kann sein:
  - `comment`: ist der Kommentar, der in den nächsten Task eingetragen wird. Der Parameter ist optional.
  - `contentId`: die ID einer zur Datei gehörenden Version (siehe den Dateiparameter `releasedVersions`). Der Parameter ist optional. Ist er nicht angegeben, wird eine Kopie der freigegebenen Version erstellt. Hat die Datei keine freigegebene Version, wird eine leere Arbeitsversion angelegt.
- *`value`* enthält den Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionWrite` für die angegebene Datei haben.

Beispiel:

```
CM>obj withId 2007 edit
```

obj objRef editedContent

Verfügbar für: Content Management Server

Aufgabe: Funktionen aus dieser Funktionsgruppe erlauben den Zugriff auf die Arbeitsversion oder die eingereichte Version einer Datei.

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root editedContent` *`contentSubCommand`*

Funktionsparameter:

- *contentSubCommand* gibt das Subkommando an; siehe Versionen.

Rückgabewert bei Erfolg: Abhängig vom Subkommando.

Erforderliche Rechte (nur CM): Abhängig vom Subkommando.

Beispiel:

```
CM>obj withId 25687 editedContent get title
Der Titel der Arbeitsversion
```

Weitere Beispiele zu diesem Befehl finden Sie im Abschnitt über den content-Befehl.

obj objRef exportSubtree

Verfügbar für: Content Management Server

Aufgabe: Exportiert die Dateihierarchie ab dem angegebenen Ordner.

Syntax:

`obj (withId objId) | (withPath path) | root exportSubtree {parameter value}`

Funktionsparameter:

- *parameter* spezifiziert den Namen eines Parameters, mit dessen Wert das Exportergebnis modifiziert werden kann. Folgende Werte sind für *parameter* zulässig:
  - `absoluteFsPathPrefix`: gibt an, dass als Dateipfade exportierte Links als absolute Pfade exportiert werden sollen und *value* als deren Präfix verwendet werden soll. Dieser Parameter hebt im Rahmen dieses Befehls die Werte der entsprechenden Parameter im Systemkonfigurationseintrag `export` auf. Ist dieser nicht angegeben, wird `/` verwendet.
  - `absoluteUrlPrefix`: wie `absoluteFsPathPrefix`, jedoch für die Pfade in URLs.
  - `exportCharset`: gibt an, dass *value* den beim Export für generierte Dokumente zu verwenden Zeichensatz benennt. Die unterstützten Werte sind im Systemkonfigurationseintrag `export.charsetMap` aufgeführt. Ist dieser Parameter nicht angegeben, wird der Wert des Systemkonfigurationseintrags `export.charset` verwendet. Ist dieser nicht vorhanden, wird `utf-8` verwendet.
  - `filePrefix`: gibt an, dass *value* einen Prefix enthält, der den Exportdateinamen voranzustellen ist.
  - `maxDepth`: gibt an, dass *value* eine ganze Zahl enthält die Anzahl der zu exportierenden Ordnerebenen spezifiziert. Ist dieser Wert nicht angegeben, ist die Anzahl nicht begrenzt.
  - `purgeCollections`: gibt an, dass *value* festlegt, ob die Collections vor dem Export gelöscht werden sollen (`1`, Voreinstellung) oder nicht (`0`).
  - `templateName`: gibt an, dass *value* den Namen einer Layoutdatei enthält, die (statt des benutzerspezifischen Basislayouts) beim Export verwendet werden soll. Ist dieser Parameter nicht angegeben, wird der Wert des Benutzerkonfigurationseintrags `defaultTemplate` verwendet. Ist dieser nicht vorhanden, wird `mastertemplate` verwendet.
- *value* enthält den Wert für den betreffenden Parameter.

Zusatzinformationen:

- Dieser Befehl kann nur auf Ordner angewendet werden.

- Beim Export wird eine der Ordnerhierarchie entsprechende Verzeichnisstruktur erzeugt, in der für jeden Ordner ein eigenes Verzeichnis erstellt wird. In diesem Verzeichnis wird der Inhalt des Ordners als Datei `index.html` abgelegt. Dokumente werden in dieses Verzeichnis als *`dokumentname`*`.html` exportiert, Bilder und Ressourcen erhalten die jeweilige Dateinamenserweiterung.
- Mit dem Systemkonfigurationseintrag `export.abortOnError` (`YES` oder `NO`) lässt sich steuern, ob der Export bei Fehlern abgebrochen oder fortgesetzt werden soll.
- Läuft der Search Engine Server und wurde in der Systemkonfiguration des Content Managers der Eintrag `indexing.staticExport.isActive` auf `true` gesetzt, so werden während des Exports die UTF-8-kodierten Webdokumente in die konfigurierten Collections indiziert.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): der Benutzer muss das Recht `permissionGlobalExport` für die Datei haben.

Beispiel:

```
CM>obj withId 4567 exportSubtree filePrefix /tmp
```

obj objRef forward

Verfügbar für: Content Management Server

Aufgabe: Reicht die Datei an die nächste Gruppe im Bearbeitungs-Workflow weiter. Hierdurch entsteht eine neue Aufgabe (`task`).

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root forward [comment` *`taskComment`*`]`

Funktionsparameter:

- *`taskComment`* ist der Kommentar, der in dem neuen Task eingetragen werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM):

- der Benutzer muss das Recht `permissionWrite` für die angegebene Datei haben.
- der Benutzer muss der Bearbeiter der angegebenen Datei sein.

Beispiel:

```
CM>obj withId 27814 forward
```

obj objRef get

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Liefert den Wert des angegebenen Dateifeldes.

Syntax:

`obj (withId objId) | (withPath path) | root get objAttr`

Zusatzinformationen: Anstelle von Dateifeldern können auch Versionsfelder angegeben werden, wenn die Datei wenigstens eine Version hat. Bezüglich der zurückgegebenen Werte haben freigegebene Versionen Vorrang vor Arbeitsversionen oder eingereichten Versionen. Die Template Engine verfügt nur über freigegebene Versionen.

Funktionsparameter:

- *objAttr* gibt den Namen des Dateifeldes an, dessen Wert gesucht wird (siehe Dateifelder).

Rückgabewert bei Erfolg: der Wert des abgefragten Dateifeldes (string).

Erforderliche Rechte (nur CM): der Benutzer muss das Recht `permissionRead` für die angegebene Datei haben. Die Dateifelder `id`, `name`, `visibleName`, `path`, `visiblePath` und `objType` können auch ohne `permissionRead` ausgelesen werden.

Beispiel:

```
CM>obj withId 2003 get children
2033 4127 98314
```

obj objRef getHierarchy

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Liefert die unterhalb des angegebenen Ordners liegende Ordnerhierarchie zurück.

Syntax:

`obj (withId objId) | (withPath path) | root getHierarchy {parameter value}`

Zusatzinformationen: Die Hierarchie enthält nur die IDs der Dateien, auf die der Benutzer lesend zugreifen darf (TE-Benutzer haben immer root-Zugriff und können daher alle Dateien lesen). Die Hierarchie wird als eine Tcl-Liste mit zwei Elementen zurückgegeben. Das erste Element dieser Liste kann 0 oder 1 sein und zeigt an, ob die Hierarchie abgeschnitten werden musste (1) oder nicht (0). Das zweite Element ist eine Liste, die die eigentliche Hierarchie enthält. In dieser Liste ist jede CMS-Datei, die kein Ordner ist, durch ihre ID repräsentiert. Ordner dagegen sind als Listen repräsentiert, deren erstes Element die ID des Ordners und deren zweites Element der Inhalt des Ordners in Form einer Liste ist.

Funktionsparameter:

- *parameter* spezifiziert den Namen eines Parameters, dessen Wert im entsprechenden *value* erwartet wird. Es gibt die folgenden Parameternamen:
  - `maxDepth`: gibt die maximale Tiefe an, die die ermittelte Hierarchie aufweisen darf. Der voreingestellte Wert ist 4, sofern er nicht durch die Benutzereinstellung `maxHierarchyDepth` (in den persönlichen Einstellungen oder den allgemeinen Benutzereinstellungen `userManagement.preferences`) überschrieben wurde.

    Der Wert `-1` steht für eine unbeschränkte Tiefe (ab Version 6.7.2).
  - `maxLines`: gibt die Anzahl der Dateien an, die maximal in der zurückgegebenen Hierarchie enthalten sein dürfen. Der voreingestellte Wert ist 200, sofern er nicht durch

die Benutzereinstellung `maxHierarchyLines` (in den persönlichen Einstellungen oder den allgemeinen Benutzereinstellungen `userManagement.preferences`) überschrieben wurde.

Der Wert `-1` steht für eine unbeschränkte Anzahl (ab Version 6.7.2).

- `objTypes`: gibt die Liste der Dateitypen an, auf die die Hierarchie beschränkt werden soll. Voreingestellt sind nur Dateien vom Typ `publication` in der Hierarchie enthalten. Wird dieser Parameter angegeben und fehlt in der Liste der Typ `publication`, so ergibt sich keine Hierarchie.

- *`value`* enthält den Wert für den betreffenden Parameter.

Rückgabewert bei Erfolg: die Hierarchie als kodierte Tcl-Liste (stringlist).

Erforderliche Rechte (nur CM): der Benutzer muss das Recht `permissionRead` für die angegebene Datei haben.

Beispiel:

```
CM>obj root getHierarchy
0 {2001 {2012 {2086 2099}}}
```

Verwenden Sie den folgenden Befehl, um Ihre persönliche `maxHierarchyLines`-Einstellung zu ändern.

```
CM>userConfig setTexts maxHierarchyLines 250
```

obj objRef giveTo

Verfügbar für: Content Management Server

Aufgabe: Reicht eine Datei zur Bearbeitung an eine andere Gruppe oder einen anderen Benutzer weiter.

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root giveTo {`*`parameter value`*`}`

Funktionsparameter:

- *`parameter`* kann die folgenden Werte haben:
  - `userLogin` gibt an, dass der folgende Wert *`value`* den Benutzer enthält, dem der Task zugeordnet wird. Es muss entweder dieser Parameter oder `groupName` angegeben werden.
  - `groupName` gibt an, dass der folgende Wert *`value`* die Gruppe enthält, der der erzeugte Task zugeordnet wird. Wenn *`userLogin`* nicht angegeben ist, muss dieser Wert angegeben sein.
  - `taskComment` ist optional und bewirkt, dass der folgende Wert *`value`* als Kommentar in den neuen Task eingetragen wird.
- *`value`* ist der Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): der ausführende Benutzer muss das Recht `permissionWrite` für die angegebene Datei haben. Der Empfänger (Benutzer oder Gruppe) dieses Tasks muss das Recht `permissionWrite` haben.

Zusatzinformationen: Ab Version 6.5 ergeben sich folgende Änderungen: der ausführende Benutzer muss das Recht `permissionWrite` für die angegebene Datei haben oder er muss der Bearbeiter der Datei sein. Für den Empfänger der Datei bestehen keine Einschränkungen.

Beispiel:

```
CM>obj withId 65123 giveTo groupName admins
```

obj objRef mget

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Liefert zur spezifizierten Datei die Werte der angegebenen Dateifelder.

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root mget {`*`objAttr`*`}`

Zusatzinformationen: Anstelle von Dateifeldern können auch Versionsfelder angegeben werden, wenn die Datei wenigstens eine Version hat. Bezüglich der zurückgegebenen Werte haben freigegebene Versionen Vorrang vor Arbeitsversionen oder eingereichten Versionen. In der Template Engine existieren nur freigegebene Versionen.

Funktionsparameter:

- *`objAttr`* spezifiziert den Namen des Dateifeldes, dessen Wert gesucht wird (siehe Dateifelder).

Rückgabewert bei Erfolg: die Werte der abgefragten Dateifelder (stringlist).

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionRead` für die angegebene Datei haben. Die Dateifelder `id`, `name`, `visibleName`, `path`, `visiblePath` und `objType` können auch ohne `permissionRead` ausgelesen werden.

Beispiel:

```
CM>obj withId 2003 mget objClass isEdited
publication 1
```

obj objRef mirror

Verfügbar für: Content Management Server (ab Version 6.5.0)

Aufgabe: Legt eine Spiegeldatei der spezifizierten Datei an. Die Originaldatei darf selbst keine Spiegeldatei sein.

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root mirror {`*`parameter value`*`}`

Funktionsparameter:

- `parent`: gibt an, dass *`value`* den Pfad oder die ID des Zielordners spezifiziert – fehlt der Parameter, wird als Ziel der Ordner angenommen, in dem sich die Originaldatei befindet.

- `name`: gibt an, dass *`value`* den Namen enthält, den die Spiegeldatei erhalten soll. Fehlt der Name, wird der Name der Originaldatei verwendet. Gibt es diesen in dem Zielordner bereits, wird ein neuer Name berechnet, indem die nächste verfügbare Zahl an den Namen angehängt wird.

Rückgabewert bei Erfolg: die ID der neuen Datei (string).

Erforderliche Rechte:

- Der Benutzer muss das Recht `permissionRead` für die Originaldatei haben.
- Der Benutzer muss das Recht `permissionCreateChildren` im Zielordner haben.
- Der Benutzer muss das globale Recht `permissionGlobalMirrorHandling` haben.

Zusatzinformationen: Das Recht `permissionGlobalMirrorHandling` beinhaltet nicht das Recht, Spiegeldateien zu löschen. Beim Anlegen einer Spiegeldatei wird jedoch das Recht `permissionRoot` dieser Spiegeldatei den gleichen Gruppen zugewiesen, die im `parent` der Spiegeldatei das Recht `permissionCreateChildren` haben. Dadurch wird sichergestellt, dass Benutzer, die eine Spiegeldatei angelegt haben, diese auch wieder löschen können (solange sie über ihre Gruppenzugehörigkeit `permissionRoot` oder das globale Recht `permissionGlobalRoot` haben).

Beispiel:

```
CM>obj withId 20099 mirror parent /news/latestnews name incoming
65811
```

obj objRef permission get

Verfügbar für: Content Management Server

Aufgabe: liefert die Liste der Benutzergruppen, deren Mitglieder für die angegebene Datei das angegebene Dateirecht haben (analog zu `obj objRef get permissions.permission`).

Syntax:

`obj (withId objId) | (withPath path) | root permission permission get`

Funktionsparameter:

- *`permission`* gibt das Dateirecht an, dessen Inhaber-Benutzergruppen ausgegeben werden sollen. Verwenden Sie `obj withId id get validPermissions`, um die Liste der existierenden Dateirechte zu ermitteln.

Rückgabewert bei Erfolg: Die Liste der Namen der Gruppen, die das betreffende Dateizugriffsrecht haben (stringlist).

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionRead` für die angegebene Datei haben.

Beispiel:

```
CM>obj withId 2001 permission permissionCreateChildren get
admins
```

obj objRef permission grantTo

Verfügbar für: Content Management Server

Aufgabe: Der angegebenen Gruppe wird das angegebene Recht für die betreffende Datei erteilt. Andere Rechte der Gruppe bleiben unberührt.

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root permission` *`permission`* `grantTo group {`*`group`*`}`

Funktionsparameter:

- *`permission`* gibt das zu erteilende Recht an. Das Dateifeld `validPermissions` liefert die Namen der existierenden Rechte.
- *`group`* gibt den Namen der Gruppe an, der das Recht erteilt werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionRoot` für die angegebene Datei haben.

Beispiel:

```
CM>obj withId 41735 permission permissionCreateChildren grantTo editors
```

obj objRef permission isGrantedToGroup

Verfügbar für: Content Management Server

Aufgabe: Prüft, ob der angegebenen Gruppe das spezifizierte Recht für die betreffende Datei erteilt wurde.

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root permission` *`permission`* `isGrantedToGroup` *`group`*

Zusatzinformationen: die Mitglieder der Gruppe haben immer sämtliche dateibezogenen Rechte, wenn sie das Recht `permissionRoot` haben oder Superuser sind, auch wenn ihnen andere dateibezogene Rechte nicht erteilt wurden.

Funktionsparameter:

- *`permission`* gibt das festzustellende Recht an. Das Dateifeld `validPermissions` liefert die Namen der existierenden Rechte.
- *`group`* gibt den Namen der Gruppe an, bei der festgestellt werden soll, ob sie das angegebene Recht hat.

Rückgabewert bei Erfolg: 1, wenn die Gruppe das Recht hat, sonst 0 (bool).

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionRead` für die angegebene Datei haben.

Beispiel:

```
CM>obj withId 41735 permission permissionCreateChildren \
isGrantedToGroup editors
1
```

obj objRef permission isGrantedToUser

Verfügbar für: Content Management Server

Aufgabe: Prüft, ob einer der Gruppen, in denen der angegebene Benutzer Mitglied ist, das spezifizierte Recht für die betreffende Datei erteilt wurde.

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root permission` *`permission`* `isGrantedToUser` *`login`*

Zusatzinformationen: Die Mitglieder der Gruppe haben immer sämtliche dateibezogenen Rechte, wenn sie das dateibezogene Recht `permissionRoot` haben oder Superuser sind, auch wenn ihnen andere dateibezogene Rechte nicht erteilt wurden.

Funktionsparameter:

- *`permission`* gibt das festzustellende Recht an. Das Dateifeld `validPermissions` liefert die Namen der existierenden Rechte.
- *`login`* gibt den Anmeldenamen des Benutzers an, bei dem festgestellt werden soll, ob er das angegebene Recht hat.

Rückgabewert bei Erfolg: 1, wenn der Benutzer das Recht hat, sonst 0 (bool).

Erforderliche Rechte: Der Benutzer muss das Recht `permissionRead` für die angegebene Datei haben.

Beispiel:

```
CM>obj withId 41735 permission permissionCreateChildren \
isGrantedToUser katrin
1
```

obj objRef permission revokeFrom

Verfügbar für: Content Management Server

Aufgabe: Der angegebenen Gruppe wird das spezifizierte Recht für die betreffende Datei entzogen.

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root permission` *`permission`* `revokeFrom group {`*`group`*`}`

Funktionsparameter:

- *`permission`* gibt das der Gruppe zu entziehende Recht an. Das Dateifeld `validPermissions` liefert die Namen der existierenden Dateirechte.

- *group* gibt den Namen der Gruppe an, der das Recht entzogen werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): der Benutzer muss das Recht permissionRoot für die angegebene Datei haben.

Beispiel:

```
CM>obj withId 41735 permission permissionCreateChildren revokeFrom editors
```

obj objRef permission set

Verfügbar für: Content Management Server

Aufgabe: Die Gruppen, die das angegebene Zugriffsrecht auf eine Datei haben, werden neu festgelegt.

Syntax:

obj (withId *objId*) | (withPath *path*) | root permission *permission* set {*permSpec*}

Funktionsparameter:

- *permission* gibt das Recht an, auf das sich die Zugriffsspezifikation bezieht. Das Dateifeld validPermissions liefert die Namen der existierenden Rechte.
- *permSpec* gibt den Namen einer Gruppe an, der das Recht *permission* erteilt werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): der Benutzer muss das Recht permissionRoot für die angegebene Datei haben.

Beispiel:

```
CM>obj withId 41735 permission permissionCreateChildren set editors admins
```

obj objRef reject

Verfügbar für: Content Management Server

Aufgabe: Bricht den aktuellen Workflow ab und beginnt mit einem neuen (Workflow-Aktion *Ablehnen*).

Syntax:

obj (withId *objId*) | (withPath *path*) | root reject [comment *taskComment*]

Zusatzinformationen: Falls die Datei eine eingereichte Version hat, wird sie in eine Arbeitsversion umgewandelt.

Funktionsparameter:

- *taskComment* ist der Kommentar, der in dem neuen Task eingetragen werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM):

- Wenn es eine Arbeitsversion gibt, muss der Benutzer das Recht `permissionWrite` für die Datei haben.
- Wenn es eine eingereichte Version gibt, muss der Benutzer das Recht `permissionRead` für die Datei haben und Mitglied in einer der verbleibenden unterschriftsberechtigten Gruppen sein oder das Recht `permissionRoot` für die angegebene Datei haben.

Beispiel:

```
CM>obj withId 2003 reject
```

obj objRef release

Verfügbar für: Content Management Server

Aufgabe: Die angegebene Datei wird freigegeben.

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root release {parameter` *`value`*`}`

Zusatzinformationen: Sollte die Datei noch nicht eingereicht sein, so wird sie vorher eingereicht (d. h. einer `commit`-Operation unterzogen). Sollten noch Unterschriften fehlen, so werden sie erzeugt.

Funktionsparameter:

- *`parameter`* spezifiziert den Namen eines Parameters der Workflowaktion. Folgende Werte sind zulässig:
  - *`comment`* gibt an, dass *`value`* der Kommentar ist, der in der Logdatei erscheint (bei der Workflow-Aktion *Freigeben* wird kein Task erzeugt).
- *value* enthält den Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM):

- Wenn es eine Arbeitsversion gibt, muss der Benutzer der Bearbeiter sein und das Schreibrecht haben (`permissionWrite`). Ferner muss die Freigabe der letzte Schritt im Workflow der Version sein, es sei denn, der Benutzer hat das Dateiadministrationsrecht. In diesem Fall kann er die Version unabhängig vom Workflow freigeben, wenn er der Bearbeiter ist.
- Wenn es eine eingereichte Version gibt und nur eine Unterschrift fehlt, muss der Benutzer Mitglied in einer Gruppe sein, die zu dieser Unterschrift berechtigt ist, und das Leserecht haben.

Beispiel:

```
CM>obj withId 2003 release
```

obj objRef releasedContent

Verfügbar für: Content Management Server

Aufgabe: Funktionen aus dieser Funktionsgruppe erlauben den Zugriff auf die freigegebene Version einer Datei.

Syntax:

```
obj (withId objId) | (withPath path) | root releasedContent versionSubCommand
```

Funktionsparameter:

- *versionSubCommand* gibt das Subkommando an; siehe Versionen.

Rückgabewert bei Erfolg: abhängig vom Subkommando.

Erforderliche Rechte (nur CM): abhängig vom Subkommando.

Beispiel: (Beispiele zu diesem Befehl finden Sie im Abschnitt Versionen.)

obj objRef removeActiveContents

Verfügbar für: Content Management Server

Aufgabe: Diese Funktion löscht alle aktiven Versionen der angegebenen Datei, d. h. die aktuelle Arbeits- oder eingereichte Version und die aktuelle freigegebenen Version. Archivierte Versionen werden von dieser Funktion nicht gelöscht (siehe `content withId contentId delete`).

Syntax:

```
obj (withId objId) | (withPath path) | root removeActiveContents
```

Zusatzinformationen: Falls eine freigegebene Version existiert, so wird sie nur gelöscht, wenn die Archivierung ausgeschaltet ist. Ansonsten wird sie in eine archivierte Version umgewandelt.

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): der Benutzer muss der Administrator der Datei sein, zu der die Version gehört.

Beispiel:

```
CM>obj withId 2003 removeActiveContents
```

obj objRef removeArchivedContents

Verfügbar für: Content Management Server

Aufgabe: Diese Funktion löscht alle archivierten Versionen der Datei, d. h. alle Versionen bis auf die aktuelle Arbeits- oder eingereichte Version und die aktuelle freigegebene Version.

Syntax:

```
obj (withId objId) | (withPath path) | root removeArchivedContents
```

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): der Benutzer muss der Administrator der Version sein, zu der der Content gehört.

Beispiel:

```
CM>obj withId 2003 removeArchivedContents
```

obj objRef revert

Verfügbar für: Content Management Server

Aufgabe: Löscht die Arbeitsversion der angegebenen Datei.

Syntax:

`obj (withId objId) | (withPath path) | root revert [comment taskComment]`

Funktionsparameter:

- *taskComment* ist der Kommentar, der in dem neuen Task eingetragen werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionWrite` für die angegebene Datei haben.

Beispiel:

```
CM>obj withId 20033 revert
```

obj objRef set

Verfügbar für: Content Management Server

Syntax:

`obj (withId objId) | (withPath path) | root set {objAttr value}`

Aufgabe: Setzt bei einer Datei die angegebenen Felder auf die spezifizierten Werte.

Funktionsparameter:

- *objAttr* das Feld der Datei, dessen Wert gesetzt werden soll (siehe Dateifelder).
- *value* ist der zu setzende Wert des betreffenden Dateifeldes.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionRoot` für die angegebene Datei haben.

Beispiel:

```
CM>obj withId 2003 set parent /news/newslist
```

obj objRef setPermissions

Verfügbar für: Content Management Server

Aufgabe: Zunächst wird allen Benutzergruppen, die das angegebene Recht haben, dieses Recht entzogen. Anschließend wird den angegebenen Gruppen das Recht erteilt. Dies entspricht der Neuzuweisung eines Zugriffsrechts zu den angegebenen Gruppen.

Syntax:

`obj (withId objId) | (withPath path) | root setPermissions {permission permSpec}`

Funktionsparameter:

- *permission* gibt das neu zuzuweisende Dateirecht an.
- *permSpec* ist eine Liste von Gruppen, denen das Recht *permission* erteilt werden soll. Allen nicht in dieser Liste enthaltenen Gruppen wird das Recht entzogen.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): der Benutzer muss das Recht `permissionRoot` für die angegebene Datei haben.

Beispiel:

```
CM>obj withId 20013 setPermissions permissionRoot admins \
permissionCreateChildren {editors subeditors}
```

obj objRef sign

Verfügbar für: Content Management Server

Aufgabe: Versieht die eingereichte Version der angegebenen Datei mit der Signatur des angemeldeten Benutzers und erzeugt eine neue Aufgabe ("task").

Syntax:

`obj (withId objId) | (withPath path) | root sign {parameter value}`

Zusatzinformationen:

- Die letzte Unterschrift in einem Workflow kann nur geleistet werden, indem die Datei freigegeben wird.
- Ein Dateiadministrator ist immer dazu berechtigt, die Versionen seiner Dateien abzuzeichnen.

Funktionsparameter:

- *parameter* spezifiziert den Namen eines Parameters der Workflowaktion. Folgende Werte sind zulässig:

- `comment` gibt an, dass *`value`* der Kommentar ist, der in den nächsten Task eingetragen wird.
- *`value`* enthält den Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionRead` für die angegebene Datei haben und Mitglied einer Gruppe sein, die dazu berechtigt ist, die Unterschrift zu leisten.

Beispiel:

```
CM>obj withPath /news/newslist sign
```

obj objRef take

Verfügbar für: Content Management Server

Aufgabe: Der Befehl macht den angemeldeten Benutzer zum Bearbeiter der Datei und erzeugt eine neue Aufgabe (`task`).

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root take [comment` *`taskComment`*`]`

Funktionsparameter:

- *`taskComment`*: ist der Kommentar, der in dem neuen Task eingetragen werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionWrite` für die angegebene Datei haben.

Beispiel:

```
CM>obj withId 7916 take
```

obj objRef touch

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Wenn der Befehl in der Template Engine ausgeführt wird, so invalidiert er die freigegebene Version der betreffenden Datei (und simuliert damit eine Änderung). Dies bewirkt, dass die Datei beim nächsten Exportzyklus auf dem Live-Server aktualisiert wird.
Beim Einsatz im Content Manager wird ein neuer Update Record für die freigegebene Version der Datei erzeugt. Dies bewirkt ebenfalls, dass die Datei beim nächsten Exportzyklus auf dem Live-Server aktualisiert wird.

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root touch`

Zusatzinformationen: In der Template Engine ist der Befehl während des Exports nicht ausführbar.

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionGlobalExport` haben.

Beispiel:

```
CM>obj withId 91663 touch
```

obj objRef unrelease

Verfügbar für: Content Management Server

Aufgabe: Der Befehl nimmt die Freigabe der angegebenen Datei zurück.

Zusatzinformationen: Die aktuelle freigegebene Version wird archiviert, wenn die Archivierung eingeschaltet ist. Wenn es keine Arbeitsversion gibt, so wird die freigegebene Version in eine neue Arbeitsversion übernommen. Die Datei verfügt anschließend über keine freigegebene Version mehr.

Syntax:

`obj (withId *objId*) | (withPath *path*) | root unrelease [comment *taskComment*]`

Funktionsparameter:

- *taskComment* ist ein Kommentar, der in der Logdatei erscheint.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionRoot` für die angegebene Datei haben.

Beispiel:

```
CM>obj withId 24521 unrelease
```

obj objRef updateCache

Verfügbar für: Template Engine

Aufgabe: Diese Anweisung bewirkt, dass die Template Engine die betreffende CMS-Datei sofort in das Offline-Verzeichnis exportiert, also unabhängig vom inkrementellen Export. Dies kann bei der Suche nach Exportfehlern nützlich sein.

Syntax:

`obj (withId *objId*) | (withPath *path*) | root updateCache`

Zusatzinformationen: Der Befehl ist während des Exports nicht ausführbar. Bevor der Befehl zum Zwecke der Fehlersuche ausgeführt wird, sollten im Systemkonfigurationseintrag `server.log` die `level`-Werte für die Protokolldateien `info.log` und `error.log` auf `3` gesetzt und die Template Engine neu gestartet werden. Wird anschließend der Befehl ausgeführt, können den genannten Protokolldateien (im instanzenspezifischen `log`-Verzeichnis) Details über den Export der betreffenden

CMS-Datei entnommen werden. Die `level`-Werte sollten nach Abschluss der Untersuchung wieder auf einen niedrigeren Wert gesetzt werden.

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: keine.

Beispiel:

```
CM>obj withId 91663 updateCache
```

obj objRef updateInIndex

Verfügbar für: Content Management Server

Aufgabe: Indiziert alle Versionen der angegebenen Datei neu. Das Kommando kann nur verwendet werden, wenn die Infopark Search Cartridge eingesetzt wird.

Syntax:

`obj (withId` *`objId`*`) | (withPath` *`path`*`) | root updateInIndex`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der Benutzer muss das Recht `permissionRoot` für die betreffende Datei haben.

Beispiel:

```
CM>obj withId 24521 updateInIndex
```

obj objRef verifyExport

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Der Befehl prüft nach einem Export des angegebenen Ordners, ob es zu jeder zu exportierenden Datei eine Ausgabedatei gibt und ob es keine Ausgabedateien zu Dateien gibt, die nicht exportiert werden dürfen. Im Wesentlichen werden also das Vorhandensein einer freigegebenen Version, deren Gültigkeitszeitraum und dessen Feld `suppressExport` geprüft. Der Befehl testet nicht, ob es Ausgabedateien für nicht existierende Dateien gibt.

Zusatzinformationen:

- Mit dem Kommando kann auch ermittelt werden, ob der exportierte Datenbestand der Template Engine mit dem internen Datenbestand des Content Managers übereinstimmt, d. h. ob alle vom Content Manager für den Export vorgesehenen Dateien tatsächlich auch von der Template Engine exportiert wurden. Hierfür führt man den Befehl mit dem Content Manager aus und gibt als Verzeichnis das Export-Verzeichnis der Template Engine an (siehe Beispiel).

- Das Ergebnis dieses Befehls ist nur verlässlich, wenn nach dem Export bis zum Ende der Verifizierung keine Dateien geändert werden.

Syntax:

`obj (withId *objId*) | (withPath *path*) | root verifyExport filePrefix *exportDir*`

Funktionsparameter:

- *exportDir* ist der Pfad des Verzeichnisses, in dem sich die zu untersuchenden Ausgabedateien befinden (wird wie bei exportSubtree angegeben).

Rückgabewert bei Erfolg: keiner. Bei Misserfolg werden zeilenweise die relativen Pfade der fehlenden und unerwartet vorhandenen Dateien ausgegeben (siehe Beispiel).

Erforderliche Rechte (nur CM): der Benutzer benötigt das Recht permissionGlobalExport.

Beispiel:

```
CM>obj withPath /pub verifyExport \
filePrefix /opt/Infopark/NPS/instance/default/export/online/docs
missing: /index.html
missing: /pub/p35.html
unexpected: /pub/s44.html
```

4.13 objClass (Dateivorlagen)

4.13.1 Dateivorlagenparameter

Parameter	Typ	Erklärung	get	set	create	descr
attributeGroupNames	stringlist	Die Namen der Feldergruppen der Dateivorlage. Die Reihenfolge der zurückgegebenen Namen entspricht der der Gruppen. (Siehe attributeGroup.)	•			
attributes	stringlist	die Liste der Felder, die in der Dateivorlage verwendet werden	•	•	•	•

availableBlobEditors	stringlist	Die Liste der Editoren, mit denen Benutzer den Hauptinhalt der Arbeitsversion bearbeiten können. Die Liste kann eine beliebige Kombination der folgenden Elemente enthalten: `internalEditor`, `externalEditor`, `tinymceEditor`, `htmlEditor`. In CMS Fiona 6.8.0 wurde der Microsoft HTML-Editor (`msieEditor`) vom TinyMCE (`tinymceEditor`) abgelöst.	•	•	•	
bodyTemplateName	string	Der Name der Layoutdatei, mit der die jeweiligen Hauptinhalte aller Dateien exportiert werden, die auf der Dateivorlage basieren.	•	•	•	
canCreateNewsItem	bool	Gibt an, ob für die Dateien mit dieser Vorlage ein News-Eintrag erzeugt werden soll.	•	•	•	•
completionCheck	string	Kundenspezifisches Tcl-Skript, mit dem zusätzliche Vollständigkeits-Überprüfungen vorgenommen werden können. Das Skript wird immer dann aufgerufen, wenn eine Version eingereicht wird. Eine Version kann nur eingereicht werden, wenn alle Links aufgelöst, die obligatorischen Felder mit gültigen Werten belegt sind und der String `completionCheck` leer ist oder ein Skript enhält, das als Ergebnis 1 zurückliefert.	•	•	•	
contentTypes	stringlist	Die Liste der für die Version einer Datei mit dieser Vorlage zulässigen Dateinamensendungen. Ergibt sich aus `validContentTypes` und `obj contentTypesForObjType`.	•			
createPermission	string	Das für die Erzeugung einer Datei mit dieser Vorlage erforderliche Recht.	•	•	•	•

defaultAttributeGroupName	string	Der Name der Basisfeldgruppe (`baseGroup`).	•			
emptyAttributeGroups	stringlist	Die Liste der Feldergruppen, denen keine Felder zugewiesen wurden.	•			
customBlobEditorUrl	string	Die URL, an die ein Request gesendet wird, wenn der Hauptinhalt einer Arbeitsversion mit dem Zusatz-Editor bearbeitet werden soll (siehe auch `availableBlobEditors`).	•	•	•	
displayTitle	string	Der in der HTML-Benutzerschnittstelle angezeigte Titel der Dateivorlage	•			
getKeys	stringlist	Liste der mit `get` abfragbaren Parameter	•			
goodAttributeGroupAttributes	stringlist	Liste der Felder, die in Feldergruppen der Dateivorlage aufgenommen werden können.	•			
goodAttributes	stringlist	Liste der Felder, die in die Dateivorlage aufgenommen werden können.	•			
goodMandatoryAttributes	stringlist	Liste der Felder, die in die Liste der `mandatoryAttributes` der Dateivorlage aufgenommen werden können.	•			
goodPresetAttributes	stringlist	Liste der Felder, die in die Liste der `presetAttributes` der Dateivorlage aufgenommen werden können.	•			
goodPresetFromParentAttributes	stringlist	Liste der Felder, die in die Liste der `presetFromParentAttributes` der Dateivorlage aufgenommen werden können.	•			
isEnabled	bool	gibt an, ob die Dateivorlage Dateien zugewiesen werden darf	•	•	•	•
localizedTitle	string	Der Titel `title.language` in der Sprache, die der Benutzer eingestellt hat. Ist dieser leer, wird `name` zurückgegeben.	•			
mandatoryAttributes	stringlist	Liste der obligatorischen Felder einer Datei mit dieser Dateivorlage (nur Versions-	•	•	•	•

		oder Dateifelder (siehe `obj get` *`parameter`*)				
name	string	der Name der Dateivorlage	•	•	•	•
objType	string	Der Typ einer Datei mit dieser Vorlage (document, publication, template, image, generic)	•		•	•
presetAttributes	stringlist	Liste der Felder, die bei der Dateierzeugung mit vordefinierten Werten aus der Dateivorlage belegt werden. Die Liste enthält paarweise die Feldnamen und die Werte (zulässig sind alle in der `attributes`-Liste aufgeführten und die vordefinierten Felder).	•	•	•	•
presetFromParentAttributes	stringlist	Liste der Felder, deren Werte aus dem Ordner, der über einer Datei mit dieser Vorlage liegt, übernommen werden (zulässig sind alle in der `attributes`-Liste aufgeführten und die vordefinierten Felder)	•	•	•	•
recordSetCallback	string	Kundenspezifischer Tcl-Code, der aufgerufen wird, wenn einer Arbeitsversion einer Datei mit dieser Vorlage Feldwerte zugewiesen werden.	•	•	•	
setKeys	stringlist	Die Liste der mit `set` setzbaren Parameter	•			
title	string	Der Titel der Dateivorlage in der benutzerspezifischen Sprache	•	•	•	•
title.*language*	string	Der Titel der Dateivorlage in der jeweiligen Sprache (aus der `localizer`-Sektion der Systemkonfiguration).	•	•	•	•
validContentTypes	stringlist	Die Liste der für die Version einer Datei mit dieser Vorlage zulässigen Dateiendungen. Wenn dieser Wert leer ist, sind die Dateiendungen erlaubt, die `obj contentTypesForObjType` für den Dateityp dieser Vorlage ergibt.	•	•	•	•

validSortKeys	stringlist	Die für diese Dateivorlage als `sortKey` verwendbaren Felder (ab Version 6.7.1)	•			
validSortOrders	stringlist	Die als `sortOrder` setzbaren Werte (ab Version 6.7.1)	•			
validSortTypes	stringlist	Die als `sortType` setzbaren Werte (ab Version 6.7.1)	•			
validSubObjClassCheck	string	Tcl-Code, der aufgerufen wird, wenn eine Datei in einem Ordner angelegt werden soll, der auf dieser Dateivorlage beruht.	•	•	•	
validSubObjClasses	stringlist	Die Liste der Namen der Dateivorlagen, die für Dateien eines Ordners mit dieser Vorlage zulässig sind. Diese Liste darf nur in Dateivorlagen belegt sein, mit denen Dateien vom Typ `publication` (Ordner) erzeugt werden.	•	•	•	•
workflowModification	string	Tcl-Code, der aufgerufen wird, bevor der Arbeitsversion einer Datei, die auf dieser Dateivorlage beruht, ein Workflow zugewiesen wird.	•	•	•	
xmldtd	string	Die zur Dateivorlage gehörende XML-DTD	•			

4.13.2 Dateivorlagenbefehle

objClass create

Verfügbar für: Content Management Server

Aufgabe: Der Befehl legt eine neue Dateivorlage an.

Syntax:

```
objClass create {parameter value}
```

Funktionsparameter:

- *parameter* spezifiziert den Namen des Dateivorlagenparameters, dessen Wert bei der Erzeugung gesetzt werden soll (siehe Dateivorlagenparameter).
- *value* der zu setzende Wert des betreffenden Dateivorlagenparameters.

Rückgabewert bei Erfolg: der Name der Dateivorlage (string).

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel:

```
CM>objClass create name newslist objType publication title \
News-Liste validSubObjClasses news attributes {abstract source}
newslist
```

objClass goodAvailableBlobEditorsForObjType

Verfügbar für: Content Management Server

Aufgabe: Der Befehl liefert die Liste der Editoren, mit denen der Hauptinhalt oder Feldwerte der Arbeitsversion einer Datei des angegebenen Typs bearbeitet werden können.

Zusatzinformationen: Die Liste der Editoren, mit denen der Hauptinhalt oder Feldwerte einer Version tatsächlich bearbeitet werden können, kann aus dem Dateivorlagenparameter `availableBlobEditors` gelesen werden.

Syntax:

`objClass goodAvailableBlobEditorsForObjType` *`objType`*

Funktionsparameter:

- *`objType`* spezifiziert den Typ der Datei.

Rückgabewert bei Erfolg: die Liste der verfügbaren Editoren für den angegebenen Dateityp (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>objClass goodAvailableBlobEditorsForObjType document
internalEditor externalEditor tinymceEditor htmlEditor
```

objClass list

Verfügbar für: Content Management Server

Aufgabe: Gibt die Liste der Namen sämtlicher Dateivorlagen aus.

Syntax:

`objClass list`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die Liste der Dateivorlagen (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>objClass list
document generic image news newslist publication template
```

objClass validAttributes

Verfügbar für: Content Management Server

Aufgabe: Gibt die Liste der zusätzlichen (d.h. der kundenspezifischen) Felder aus, die in der Gesamtheit der Dateivorlagen verwendet werden.

Syntax:

`objClass validAttributes`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die Liste der Feldnamen (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>objClass validAttributes
abstract keywords kosten levels list size source
```

Mit dem folgenden Tcl-Code können Sie feststellen, in welchen Vorlagen ein Feld (hier `myfield`) verwendet wird:

```
set fieldname myfield
foreach c [objClass list] {
  if { [lsearch  [objClass withName $c get attributes] $fieldname]  != -1} {
    puts $c
  }
}
```

objClass where

Verfügbar für: Content Management Server

Aufgabe: ermöglicht die Suche nach Dateivorlagen, bei denen der Wert der angegebenen Parameter die jeweils spezifizierten Zeichenkette enthält.

Syntax:

`objClass where {`*`parameter value`*`}`

Funktionsparameter: (Werden keine Parameter angegeben, so werden die Namen aller Dateivorlagen ausgegeben.)

- *`parameter`* spezifiziert den Namen des Parameters, nach dessen Wert gefragt werden soll oder legt die Anzahl der Suchergebnisse fest. Folgende Parameter sind erlaubt:
  - `maxResults` gibt an, dass *`value`* eine Zahl enthält, die die maximale Anzahl Suchergebnisse festlegt. Ist diese Zahl kleiner oder gleich null, ist die Anzahl der Suchergebnisse nicht beschränkt.
  - `name` gibt an, dass die Namen der Dateivorlagen auf das Vorkommen von *`value`* untersucht werden.
  - `isEnabled` gibt an, dass *`value`* einen Wahrheitswert enthält und nur die Namen der Dateivorlagen zurückgegeben werden sollen, bei denen der Wert des Parameters `isEnabled`

dem des Wahrheitswerts entspricht. (Nur Dateivorlagen, bei denen `isEnabled` wahr ist, können Dateien zugewiesen werden.)

- `objType` gibt an, das nur die Namen der Dateivorlagen geliefert werden sollen, deren Dateityp in der als *`value`* angegebenen Liste enthalten ist.

- *`value`* enthält den Wert des entsprechenden Parameters.

Rückgabewert bei Erfolg: die Liste der Namen der passenden Dateivorlagen (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>objClass where objType publication
newslist publication
```

objClass objClassRef delete

Verfügbar für: Content Management Server

Aufgabe: löscht die angegebene Dateivorlage.

Syntax:

`objClass withName` *`objClassName`* `delete`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel:

```
CM>objClass withName newslist delete
```

objClass objClassRef description

Verfügbar für: Content Management Server

Aufgabe: Liefert eine String-Repräsentation der Daten der Dateivorlage mit dem angegebenen Namen.

Zusatzinformationen: die Repräsentation wird im Property-List-Format eines Dictionarys formatiert (siehe Das Property-List-Format).

Syntax:

`objClass withName` *`objClassName`* `description`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die String-Repräsentation der Dateivorlage (string).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>objClass withName newslist description
{
  attributes = (
    abstract,
    source
  );
  isEnabled = 1;
  name = newslist;
  objType = publication;
  presetAttributes = {};
  title = "News-Liste";
  validSubObjClasses = (
    news
  );
}
```

objClass objClassRef get

Verfügbar für: Content Management Server

Aufgabe: Liefert von der spezifizierten Dateivorlage den Wert des angegebenen Parameters.

Syntax:

`objClass withName` *`objClassName`* `get` *`parameter`*

Funktionsparameter:

- *`parameter`* spezifiziert den Namen des Parameters, dessen Wert gesucht wird (siehe Dateivorlagenparameter).

Rückgabewert bei Erfolg: der Wert des angegebenen Parameters (string).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>objClass withName newslist get title
News-Liste
```

objClass objClassRef mget

Verfügbar für: Content Management Server

Aufgabe: Liefert die Werte der angegebenen Parameter der spezifizierten Dateivorlage.

Syntax:

`objClass withName` *`objClassName`* `mget {`*`parameter`*`}`

Funktionsparameter:

- *`parameter`* spezifiziert den Namen des Parameters, dessen Wert gesucht wird. (siehe Dateivorlagenparameter).

Rückgabewert bei Erfolg: die Liste der Werte der betreffenden Parameter (stringlist)

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>objClass withName newslist mget objType title
publication News-Liste
```

objClass objClassRef set

Verfügbar für: Content Management Server

Aufgabe: Setzt bei der angegebenen Dateivorlage die spezifizierten Parameter auf die angegebenen Werte.

Zusatzinformationen: Ab Version 6.6.1 von Infopark CMS Fiona kann der Name einer Dateivorlage mit dem `set`-Befehl geändert werden. Eine solche Umbenennung bewirkt, dass alle CMS-internen Referenzen auf die Vorlage automatisch angepasst werden. Hierdurch wird unter Umständen ein kompletter Neuexport auf dem Live-Server (bei Einsatz der Template Engine) ausgelöst. In Layouts und Skripten müssen die Namen manuell angepasst werden.

Syntax:

`objClass withName` *`objClassName`* `set {`*`parameter value`*`}`

Funktionsparameter:

- *`parameter`* spezifiziert den Parameter der Dateivorlage, dessen Wert gesetzt werden soll (siehe Dateivorlagenparameter).
- *`value`*: der zu setzende Wert des jeweiligen Parameters.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel:

```
CM>objClass withName news set title News-Artikel
```

4.14 reminder (Wiedervorlage)

Mit dem ab Version 6.5.0 Verfügbaren Befehl `reminder` können Wiedervorlagen angelegt, angezeigt und gelöscht werden.

4.14.1 reminder define

Verfügbar für: Content Management Server

Aufgabe: Richtet einen Wiedervorlagetermin mit den angegebenen Parametern ein.

Syntax:

`reminder define {`*`parameter value`*`}`

Funktionsparameter:

- *parameter* kann einer der folgenden Werte sein:
  - objectId: ID der Datei, für die der Wiedervorlagetermin eingerichtet werden soll.
  - from: Der Wiedervorlagetermin im 14-stelligen Datumsformat (siehe Beispiel).
  - users: Die Liste der Benutzer, für die der Wiedervorlagetermin eingerichtet werden soll.
  - groups: Die Liste der Gruppen, für die der Wiedervorlagetermin eingerichtet werden soll.
  - comment: Anmerkung für die Wiedervorlage (beispielsweise Bearbeitungshinweis).
- *value* enthält den jeweiligen Wert des als *parameter* angegebenen Parameters.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Schreibrechte für die angegebene Datei

Beispiel:

```
CM> reminder define objectId 2016 from 20101221080000 \
users {"mustermann"} comment "Slogan aktualisieren!"
```

4.14.2 reminder delete

Verfügbar für: Content Management Server

Aufgabe: Löscht einen Wiedervorlagetermin.

Syntax:

reminder delete {*parameter value*}

Funktionsparameter:

- *parameter* muss objectId sein. Dieser Wert gibt an, dass *value* die ID der Datei ist, deren Wiedervorlage gelöscht werden soll.
- *value* enthält den jeweiligen Wert des als *parameter* angegebenen Parameters.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Schreibrechte für die angegebene Datei

Beispiel:

```
CM> reminder delete objectId 2016
```

4.14.3 reminder where

Verfügbar für: Content Management Server

Aufgabe: Der Befehl erlaubt es nach Wiedervorlagen zu suchen, die bestimmte Kriterien erfüllen.

Syntax:

`reminder where {parameter value}`

Funktionsparameter:

- *parameter* definiert ein Kriterium, das die gefundenen Wiedervorlagen erfüllen müssen. Es können mehrere Parameter angegeben werden:
  - `from`: Das Wiedervorlagedatum muss nach dem als *value* angegebenen Datum (im 14-stelligen Format) liegen.
  - `until`: Das Wiedervorlagedatum muss vor dem als *value* angegebenen Datum (im 14-stelligen Format) liegen.
  - `user`: Der als *value* angegebene Benutzer ist ein Empfänger.
  - `group`: Die als *value* angegebene Benutzergruppe (d.h. ihre Mitglieder) ist ein Empfänger.
  - `objectID`: Die Wiedervorlage gehört zu einer Datei, die sich in einer Teilhierarchie befindet. Diese Teilhierarchie beginnt mit dem Ordner, dessen ID als *value* angegeben ist.
- *value* enthält den zum entsprechenden *parameter* gehörenden Wert.

Rückgabewert bei Erfolg: Eine Liste mit den Daten der gefundenen Wiedervorlagen.

Erforderliche Rechte: keine

Beispiel:

```
CM> reminder where objectId 2016 until 20101225000000 user mustermann
{objectId 2016 from 201001221080000 users mustermann objectType document path /news/
article1 groups {} comment {Slogan aktualisieren!}}
```

4.15 statistics (Statistische Informationen)

4.15.1 Statistikparameter

Parameter	Typ	Erklärung	Anwend.	get	descr
getKeys	stringlist	Liste der `statistics` -Parameter, die mit `get` abgefragt werden können.	CM, TE	•	
names	stringlist	Liefert die Liste der Schlüsselwörter, die der Subbefehl `compute` akzeptiert.	CM, TE	•	
formats	stringlist	Liefert die Formate, die der Subbefehl `compute` akzeptiert.	CM, TE	•	

4.15.2 Statistikbefehle

statistics compute

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Dieser Befehl erzeugt Statistiken über Dateien, Versionen oder Links.

Syntax:

`statistics compute {parameter value}`

Funktionsparameter:

- *parameter* kann sein:
  - name: gibt an, dass *value* eines der folgenden Schlüsselwörter enthält, mit denen die Art der zu berechnenden Statistik angezeigt wird.
    - all (CM, TE): gibt die Ergebnisse aller im Folgenden genannten Funktionen der jeweiligen Applikation aus.
    - contents (CM): erzeugt statistische Informationen über Versionen.
    - dependencies (TE): erzeugt statistische Informationen über Abhängigkeiten.
    - export (TE): erzeugt statistische Informationen zum inkrementellen Export.
    - links (CM): erzeugt statistische Informationen über Links.
    - objects (CM, TE): gibt statistische Informationen über Dateien aus.
    - topReferences (TE): erzeugt Liste der Dateien mit den meisten Abhängigkeiten.
  - format: gibt optional an, dass *value* eines der folgenden Ausgabeformate enthält.
    - tcl: das Ergebnis wird als Tcl-Liste zurückgegeben.
    - text: das Ergebnis wird als normaler Text zurückgegeben. Dies ist die Voreinstellung.
- *value* enthält den Wert für den betreffenden Parameter.

Rückgabewert bei Erfolg: Die angeforderten statistischen Angaben.

Erforderliche Rechte (nur CM): Der Benutzer benötigt das Recht permissionGlobalRoot.

Beispiel:

```
TE>statistics compute name contents
contents
--------
total contents : 258 (100.0%)
edited contents : 8 ( 3.1%)
released contents : 192 ( 74.4%)
archived contents : 58 ( 22.5%)
avg contents/object : 1.3
```

statistics get

Verfügbar für: Content Management Server, Template Engine

Aufgabe:Liefert den Wert des angegebenen statistics-Parameters.

Syntax:

`statistics get parameter`

Funktionsparameter:

- *parameter* spezifiziert den Namen des Parameters, dessen Wert gesucht wird. Die gültigen Namen sind im Abschnitt Statistics-Parameter aufgeführt.

Rückgabewert bei Erfolg: der Wert des entsprechenden Parameters.

Erforderliche Rechte (nur CM): Der Benutzer benötigt das Recht `permissionGlobalRoot`.

Beispiel:

```
TE>statistics get names
all contents links objects
```

4.16 systemConfig (Systemkonfiguration)

Der `systemConfig`-Befehl ermöglicht den schreibenden und lesenden Zugriff auf die Systemkonfiguration, die der Content Manager und die Template Engine beim Programmstart aus ihrer Initialisierungsdatei einlesen. Änderungen an der Systemkonfiguration wirken sich nur auf die laufende Applikation aus. Die betreffende Initialisierungsdatei wird dabei nicht aktualisiert. Dauerhafte Änderungen müssen also in der entsprechenden Initialisierungsdatei vorgenommen werden.

Daten, die in der Systemkonfiguration mit dem `systemConfig`-Befehl abgelegt werden sollen, müssen im XML-Format vorliegen. In Versionen des Infopark CMS bis vor 5.0 wurde stattdessen das Property-List-Format verwendet.

4.16.1 Das Property-List-Format

In einer Property List (Eigenschaftenliste) werden Namen Werte zugewiesen. Dies geschieht nach folgendem Muster:

```
name = wert;
```

Jede Zuweisung wird durch ein Semikolon abgeschlossen. Die Werte in Property Lists können Zeichenketten (Strings), Dictionarys (Listen von Name-Wert-Paaren) oder Arrays (Listen von Werten) sein.

String

Ein String ist eine Zeichenkette, die von Anführungszeichen umschlossen ist oder aus einer beliebigen Kombination der Zeichen ‘a’ - ‘z’, ‘A’ - ‘Z’, ‘0’ - ‘9’ und dem Unterstrich ‘_’ besteht.

Zulässige Zeichenketten sind beispielsweise:

```
Zeichenkette
"Kette aus Zeichen"
```

Anführungszeichen können in Zeichenketten, die von Anführungszeichen umschlossen sind, verwendet werden, indem ihnen wie im folgenden Beispiel ein umgekehrter Schrägstrich vorangestellt wird:

```
"Das Wort \"und\" besteht aus drei Zeichen"
```

Dictionary

Ein Dictionary ist eine Liste von Name-Wert-Paaren, die mit einer öffnenden geschweiften Klammer eingeleitet und mit einer schließenden geschweiften Klammer abgeschlossen wird:

```
server = {
  host = localhost;
  tclPort = 3001;
  secondPort = 3002;
};
```

Jeder Wert eines Name-Wert-Paares in einem Dictionary kann ein String, ein Array oder wiederum ein Dictionary sein.

Array (Liste)

Ein Array ist eine Liste von Zeichenketten, Dictionarys oder Arrays, die mit einer öffnenden runden Klammer eingeleitet und einer schließenden runden Klammer abgeschlossen wird. Die Elemente in einer Liste werden durch ein Komma voneinander getrennt. Die folgenden Beispiele enthalten zulässige Arrays:

```
name1 = (Ein, Array, mit, "fünf", Elementen);
name2 = ((Ein, Array), mit, (drei, Elementen));
name3 = (
  {
  name1 = Ein;
  name2 = Array;
  },
  {
  name3 = aus;
  name4 = (zwei, Dictionarys);
  }
);
```

Kommentare

Property Lists können Kommentare enthalten. Zwei Schrägstriche leiten einen Kommentar ein, der den Rest der Zeile umfasst:

```
// Dies ist der erste Kommentar
```

Kommentare, die von Daten umschlossen sind, müssen wie in folgendem Beispiel mit der Zeichenfolge „/*" eingeleitet und der Zeichenfolge „*/" abgeschlossen werden:

```
/* Dies ist der zweite Kommentar */
```

4.16.2 Systemkonfigurationsbefehle

systemConfig formatDate

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Der Befehl ist obsolet. Er hat den gleichen Funktionsumfang wie formatDateTime.

Syntax:

```
systemConfig formatDate date
```

systemConfig formatDateTime

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Formatiert das angegebene Datum. Datum und Uhrzeit werden in die Zeitzone des Servers konvertiert.

Zusatzinformationen: Als Zeitzone des Servers wird der Konfigurationswert `userManagement.preferences.timeZone` verwendet. Als Ausgabeformat wird der Wert aus dem Dictionary `export.validDateTimeOutputFormats` verwendet, auf den der Wert `userManagement.preferences.dateTimeOutputFormatName` verweist. Wird dieser Unterbefehl über `userConfig` aufgerufen, so wird der benutzerspezifische Wert von `dateTimeOutputFormatName`verwendet.

Syntax:

```
systemConfig formatDateTime dateTime
```

Funktionsparameter:

- *`dateTime`* ist das zu formatierende Datum. Es muss in der Form *YYYYMMDDhhmmss* angegeben werden. Enthält das Datum zu wenige Stellen, wird die Jahreszahl mit Nullen, Monats- und Tageszahl mit *01* und Stunden, Minuten und Sekunden mit Nullen aufgefüllt. Überzählige Stellen werden ignoriert.

Rückgabewert bei Erfolg: Der formatierte Zeitstempel (string).

Erforderliche Rechte (nur CM): keine

Beispiel:

```
CM>systemConfig formatDateTime 20000115180001
15.01.2000 19:00 MET
```

systemConfig getAttributes

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Liefert zum Systemkonfigurationselement mit dem Pfad *`elemPath`* die spezifizierten Tag-Attribute und ihre Werte.

Syntax:

```
systemConfig getAttributes elemPath {attrName}
```

Funktionsparameter:

- *`elemPath`* ist der Pfad des Konfigurationselements, dessen Tag-Attribute gesucht werden. Die Bestandteile eines Pfades müssen durch Punkte voneinander getrennt werden.
- *`attrName`* ist der Name eines Tag-Attributs, dessen Wert geliefert werden soll. Sind keine Tag-Attribute angegeben, so werden alle Tag-Attribute mit ihren Werten geliefert.

Rückgabewert bei Erfolg: Die Liste der Tag-Attribute mit ihren Werten. In dieser Liste bildet jedes Attribut mit seinem Wert ein Name-Wert-Paar.

Erforderliche Rechte (nur CM): keine

Beispiel:

```
CM>systemConfig getAttributes userManagement.globalPermissions
type list
```

systemConfig getCounts

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Liefert die Anzahl der Elemente, die unter dem Konfigurationselement mit dem angegebenen Pfad gespeichert sind.

Syntax:

`systemConfig getCounts {`*`elemPath`*`}`

Funktionsparameter:

- *`elemPath`* ist der Pfad eines Konfigurationselements, dessen Anzahl Elemente gesucht wird. Die Bestandteile eines Pfades müssen durch Punkte voneinander getrennt werden.

Rückgabewert bei Erfolg: Eine Liste mit ebenso vielen Zahlen wie Pfade angegeben wurden. Jede Zahl bezeichnet die Anzahl Unterelemente des entsprechenden Konfigurationselements.

Erforderliche Rechte (nur CM): keine

Beispiel:

```
CM>systemConfig getCounts content.globalPermissions
5
```

systemConfig getElements

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Liefert Systemkonfigurationselemente mit dem angegebenen Pfad.

Syntax:

`systemConfig getElements {`*`elemPath`*`}`

Funktionsparameter:

- *`elemPath`* ist der Pfad eines Konfigurationselements, das geliefert werden soll. Die Bestandteile eines Pfades müssen durch Punkte voneinander getrennt werden.

Rückgabewert bei Erfolg: Eine Liste mit ebenso vielen Konfigurationselementen wie Pfade angegeben wurden. Die Elemente sind nicht formatiert.

Erforderliche Rechte (nur CM): keine

Beispiel:

```
CM>systemConfig getElements tuning.master tuning.slave
```

systemConfig getKeys

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Liefert die Namen der Unterelemente der spezifizierten Systemkonfigurationselemente.

Syntax:

`systemConfig getKeys {`*`elemPath`*`}`

Funktionsparameter:

- *`elemPath`* ist der Pfad eines Konfigurationselements, dessen Unterlemente geliefert werden sollen. Die Bestandteile eines Pfades müssen durch Punkte voneinander getrennt werden.

Rückgabewert bei Erfolg: Eine Liste, die für jeden angegebenen *`elemPath`* die Liste der Namen der Unterelemente enthält.

Erforderliche Rechte (nur CM): keine

Beispiel:

```
CM>systemConfig getKeys tuning.master tuning.slave
{maxSlaves slaveIdleTimeout slaveExecMaxFailures slaveExecArguments
slaveShutdownTimeout slaveStartupTimeout minIdleSlaves}
{maxNumberOfRequests requestTimeout}
```

systemConfig getTexts

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Die Funktion liefert die Inhalte der spezifizierten Systemkonfigurationselemente.

Zusatzinformationen: Ein Konfigurationselement kann als Inhalt entweder Unterelemente oder einen Zeichenkettenwert enthalten, nicht jedoch beides. Daher liefert die Funktion eine leere Zeichenkette, wenn das auszulesende Konfigurationselement Unterelemente hat.

Syntax:

`systemConfig getTexts {`*`elemPath`*`}`

Funktionsparameter:

- *`elemPath`* ist der Pfad eines Konfigurationselements, dessen Inhalt geliefert werden soll. Die Bestandteile eines Pfades müssen durch Punkte voneinander getrennt werden.

Rückgabewert bei Erfolg: Die Liste der Inhalte der spezifizierten Konfigurationselemente. Die Inhalte sind nicht formatiert.

Erforderliche Rechte (nur CM): keine

Beispiel:

```
CM>systemConfig getTexts tuning.master.cm.minIdleSlaves content.sortKey
3 name
```

systemConfig installedLanguages

Aufgabe: Liefert die Liste der Sprachen (als Kürzel), für die es Localizer in der Systemkonfiguration gibt.

Syntax:

`systemConfig installedLanguages`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: Die Liste der verfügbaren Sprachen.

Erforderliche Rechte: keine.

Beispiel:

```
CM>systemConfig installedLanguages
en de fr es it
```

systemConfig parseInputDate

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Interpretiert den angegebenen Wert als Eingabewert für ein Datum und liefert den entsprechenden 14-stelligen Datumsstring. Datum und Uhrzeit werden dabei von der Zeitzone des Servers in die Zeitzone GMT umgewandelt.

Zusatzinformationen: Als Zeitzone des Servers wird der Konfigurationswert `userManagement.preferences.timeZone` verwendet. Sind Datumsangaben mehrdeutig, so wird zu ihrer Interpretation der Konfigurationswert `userManagement.preferences.dateTimeInputFormatName` verwendet.

Syntax:

`systemConfig parseInputDate` *`inputDate`*

Funktionsparameter:

- *`date`* ist die zu interpretierende Eingabe.

Rückgabewert bei Erfolg: Das Datum als 14-stellige Zeichenkette.

Erforderliche Rechte (nur CM): keine.

Beispiel:

```
CM>systemConfig parseInputDate {15.1.2000 19:00 MET}
20000115180000
```

systemConfig removeAttributes

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Die Funktion löscht die angegebenen Tag-Attribute des Systemkonfigurationselements mit dem angegebenen Pfad.

Zusatzinformationen: Dieser Befehl kann nur beim Applikationsstart und im Single-Modus verwendet werden.

Syntax:

```
systemConfig removeAttributes elemPath {attrName}
```

Funktionsparameter:

- *elemPath* ist der Pfad des Konfigurationselements, dessen Tag-Attribute gelöscht werden sollen. Die Bestandteile des Pfades müssen durch Punkte voneinander getrennt werden.
- *attrName* ist der Name eines Tag-Attributs, das gelöscht werden soll. Ist kein Tag-Attribut angegeben, so werden alle Attribute gelöscht.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der angemeldete Benutzer muss ein Superuser sein.

Beispiel:

```
CM>systemConfig removeAttributes tuning.master
```

systemConfig removeKeys

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Die Funktion löscht die angegebenen Elemente aus der Systemkonfiguration.

Zusatzinformationen: Dieser Befehl kann nur beim Applikationsstart und im Single-Modus verwendet werden.

Syntax:

```
systemConfig removeKeys {elemPath}
```

Funktionsparameter:

- *elemPath* ist der Pfad eines Konfigurationselements, das gelöscht werden soll. Die Bestandteile eines Pfades müssen durch Punkte voneinander getrennt werden.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der eingeloggte Benutzer muss ein Superuser sein.

Beispiel:

```
CM>systemConfig removeKeys content.globalPermissions tuning
```

systemConfig setAttributes

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Die Funktion ändert die Werte von Tag-Attributen oder fügt die als Name-Wert-Paare angegebenen Tag-Attribute zu einem Systemkonfigurationselement hinzu.

Zusatzinformationen: Dieser Befehl kann nur beim Applikationsstart und im Single-Modus verwendet werden.

Syntax:

`systemConfig setAttributes` *`elemPath`* {*`name value`*}

Funktionsparameter:

- *`elemPath`* ist der Pfad des Konfigurationselements, zu dem Tag-Attribute hinzugefügt werden sollen. Die Bestandteile des Pfades müssen durch Punkte voneinander getrennt werden.
- *`name`* ist der Name eines Tag-Attributs, das zum Konfigurationselement hinzugefügt werden soll. Hat das Element bereits ein solches Attribut, so wird dessen Wert auf den neuen Wert gesetzt.
- *`value`* ist der neue Wert des entsprechenden mit *`name`* referenzierten Attributs.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der eingeloggte Benutzer muss ein Superuser sein.

Beispiel:

```
CM>systemConfig setAttributes server.cm fileName cm.xml
```

systemConfig setElements

Aufgabe: Die Funktion ersetzt Systemkonfigurationselemente oder fügt Elemente zur Systemkonfiguration hinzu.

Zusatzinformationen: Dieser Befehl kann nur beim Applikationsstart und im Single-Modus verwendet werden.

Syntax:

`systemConfig setElements` {*`elemPath element`*}

Funktionsparameter:

- *`elemPath`* ist der Pfad eines zu ersetzenden oder zur Systemkonfiguration hinzuzufügenden Elements. Das Element selbst wird als unmittelbar darauf folgende Zeichenkette *`element`* angegeben. Die Bestandteile des Pfades müssen durch Punkte voneinander getrennt werden.
- *`element`* ist das neue Konfigurationselement mit dem Pfad *`elemPath`*.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der eingeloggte Benutzer muss ein Superuser sein.

Beispiel:

```
CM>systemConfig setElements tuning.master \
{<master fileName="master.xml"><maxSlaves>2</maxSlaves><slaveIdleTimeout>60
</slaveIdleTimeout><slaveExecMaxFailures>4</slaveExecMaxFailures>
<slaveExecArguments/><slaveShutdownTimeout>5</slaveShutdownTimeout>
<slaveStartupTimeout>100</slaveStartupTimeout><minIdleSlaves>1
</minIdleSlaves></master>}
```

systemConfig setTexts

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Die Funktion ersetzt den Inhalt von Systemkonfigurationselementen oder fügt Elemente zur Systemkonfiguration hinzu.

Zusatzinformationen:

- Dieser Befehl kann nur beim Applikationsstart und im Single-Modus verwendet werden.
- Ein Konfigurationselement kann als Inhalt entweder Unterelemente oder einen Zeichenkettenwert enthalten, nicht jedoch beides. Daher werden alle Unterelemente eines Konfigurationselements gelöscht, wenn dessen Inhalt mit `setTexts` gesetzt wird.

Syntax:

`systemConfig setTexts {`*`elemPath elemText`*`}`

Funktionsparameter:

- *`elemPath`* ist der Pfad eines zu ersetzenden oder zur Systemkonfiguration hinzuzufügenden Elements. Der Inhalt des Elements (ohne die es umgebenden Tags) wird als unmittelbar darauf folgende Zeichenkette *`elemText`* angegeben. Die Bestandteile des Pfades müssen durch Punkte voneinander getrennt werden.
- *`elemText`* ist der Inhalt des neuen Konfigurationselements mit dem Pfad *`elemPath`*.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Der angemeldete Benutzer muss ein Superuser sein.

Beispiele:

```
CM>systemConfig setTexts tuning.master.cm.minIdleSlaves 3 content.sortKey title
CM>systemConfig setTexts userManagement.preferences.maxHierarchyLines 250
```

systemConfig validInputCharsets

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Die Funktion liefert den Wert des Systemkonfigurationseintrags `inputCharsets` als Liste. Diese Liste umfasst die Namen der Zeichensätze, die beim Import von Contents zulässig sind.

Syntax:

`systemConfig validInputCharsets`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): keine Einschränkungen.

Beispiel:

```
CM>systemConfig validInputCharsets
utf-8 iso8859-1 iso8859-2
```

systemConfig validTimeZoneNames

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Die Funktion liefert die Liste der zulässigen Zeitzonennamen.

Syntax:

```
systemConfig validTimeZoneNames
```

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): keine Einschränkungen.

Beispiel:

```
CM>systemConfig validTimeZoneNames
GMT-12 GMT-11 HST HST US/Hawaii GMT-10 US/Hawaii Canada/Yukon US/Yukon GMT-9
Canada/Pacific US/Pacific-New PST8PDT US/Pacific GMT-8 US/Arizona MST GMT-7
US/Mountain MST7MDT Canada/Mountain Canada/Central Canada/East-Saskatchewan
US/Central GMT-6 CST6CDT EST EST5EDT US/East-Indiana Canada/Eastern US/Eastern
GMT-5 Canada/Atlantic GMT-4 Canada/Newfoundland GMT-3 GMT-2 GMT-1 GMT GB-Eire WET
Iceland UTC Greenwich Universal CET GMT+1 Poland MET GMT+2 EET Turkey GMT+3 W-SU
GMT+4 GMT+5 GMT+6 GMT+7 Australia/West Singapore GMT+8 Japan GMT+9 Australia/North
GMT+10 Australia/Queensland Australia/South Australia/Victoria Australia/Tasmania
GMT+11 Australia/NSW NZ GMT+12 GMT+13
```

4.17 task (Aufgaben)

4.17.1 Task-Parameter

Attribut	Typ	Erklärung	get	descr
comment	string	Kommentar der auslösenden Workflow-Aktion	•	•

displayTitle	string	Der in der HTML-Benutzerschnittstelle angezeigte Titel der Aufgabe	•	
getKeys	stringlist	Liste der mit `get` abfragbaren Parameter	•	
groupName	string	Name der Gruppe, für deren Mitglieder der Eintrag gilt	•	•
objId	string	ID der Datei, auf das sich der Eintrag bezieht	•	•
taskType	string	Die Art der durchzuführenden Aufgabe (`edit`, `sign`)	•	•
timeStamp	string	Datum der Erzeugung der Aufgabe	•	•
title	stringlist	Titel der Aufgabe	•	•
userLogin	string	Name des Nutzers, für den der Eintrag gilt	•	•

4.17.2 Task-Befehle

task list

Verfügbar für: Content Management Server

Aufgabe: Der Befehl gibt die Liste sämtlicher Task-IDs zurück.

Syntax:

```
task list
```

Funktionsparameter: keine.

Rückgabewert bei Erfolg: Liste der Task-IDs (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>task list
2140.18 5351.3 2035.7 2893.19 2252.55 5358.23 2042.12 2259.9 5148.121
```

task where

Verfügbar für: Content Management Server

Aufgabe: Gibt die IDs der Tasks zurück, die den angegebenen Kriterien genügen.

Syntax:

```
task where {parameter value}
```

Funktionsparameter:

- *parameter* Der Name des Parameters, dessen Wert in *value* folgt und nach dem gesucht wird. Werden mehrere Parameter angegeben, so werden sie mit *und* verknüpft. Parameter kann sein:

- `maxResults` gibt an, dass *`value`* eine Zahl enthält, die die maximale Anzahl Suchergebnisse festlegt. Ist diese Zahl kleiner oder gleich null, ist die Anzahl der Suchergebnisse nicht beschränkt.
- `userLogin`: Es werden nur die Tasks ausgegeben, die dem in *`value`* angegebenen Benutzer zugeordnet sind.
- `groupNames`: Es werden nur die Tasks ausgegeben, die den in der Stringliste *`value`* angegebenen Gruppen zugeordnet sind.
- `taskText`: *`value`* enthält einen Text, der im Kommentar (`comment`) oder Titel (`title`) eines Tasks vorkommen muss.
- `objId`: Nur Tasks, die sich auf die Datei mit der in *`value`* angegebenen ID beziehen, werden aufgeführt.
- `taskType` : Nur Tasks des in *`value`* angegebenen Typs werden aufgeführt. Der Typ kann sein:
  - `edit`: Nur Tasks in einem Bearbeitungsworkflow werden berücksichtigt.
  - `sign`: Nur Tasks in einem Unterschriftenworkflow werden berücksichtigt.
- *`value`* ist der Wert des jeweiligen Parameters.

Rückgabewert bei Erfolg: Liste der IDs der Tasks (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>task where taskType edit
2014.16 2035.6 2042.31 2049.13 2056.27 2063.5 2070.36 2084.44 2098.7
```

task taskRef description

Verfügbar für: Content Management Server

Aufgabe: Liefert eine String-Repräsentation der Daten des Tasks mit der angegebenen ID.

Zusatzinformationen: die Repräsentation wird im Property-List-Format eines Dictionarys formatiert.

Syntax:

`task withId` *`taskId`* `description`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: String-Repräsentation des Tasks (string).

Erforderliche Rechte: keine.

Beispiel:

```
CM>task withId 2014.17 description
{
  objId = 2001;
  taskType = edit;
  timeStamp = 20011129221803;
  title = "new\ content\ created";
  userLogin = root;
}
```

task get

Available for: Content Management Server

Aufgabe: Liefert den Wert des angegebenen Parameters für den Task *taskId*.

Syntax:

`task withId taskId get attrName`

Funktionsparameter:

- *attrName* ist der Name des Taskparameters, dessen Wert gesucht wird (siehe Taskparameter).

Rückgabewert bei Erfolg: der Wert des angegebenen Taskparameters (string).

Erforderliche Rechte: keine.

Beispiel:

```
CM>task withId 2014.17 get title
new content created
```

task taskRef mget

Verfügbar für: Content Management Server

Aufgabe: Liefert die Werte der angegebenen Parameter einer Aufgabe ("task").

Syntax:

`task withId taskId mget {attrName}`

Funktionsparameter:

- *attrName* spezifiziert den Namen des Taskparameters, dessen Wert gesucht wird (siehe Taskparameter).

Rückgabewert bei Erfolg: Liste der Taskparameter (stringlist)

Erforderliche Rechte (nur CM): keine.

Beispiel:

```
CM>task withId 2014.17 mget taskType title
edit {new content created}
```

4.18 user (Benutzer)

4.18.1 Benutzerparameter

Parameter	Typ	Erklärung	get	set	create	descr
defaultGroup	string	Standardgruppe des Benutzers	•	•	•	•
displayTitle	string	der in der HTML-Benutzerschnittstelle angezeigte Titel des Benutzers (eine Kombination aus Login und vollem Namen)	•			
encryptedPassword	string	Verschlüsseltes Passwort des Benutzers	•	•	•	•
email	string	E-Mail-Adresse des Benutzers	•	•	•	•
externalAttrNames	stringlist	Liste der kundenspezifischen Benutzerfelder des Benutzers	•			
getKeys	stringlist	Liste der mit `get` abfragbaren Parameter	•			
globalPermissions	stringlist	Liste der globalen Rechte	•	•	•	•
groups	stringlist	Liste der Gruppen, in denen der Benutzer Mitglied ist	•	•	•	•
login	string	Login des Benutzers	•		•	•
owner	string	Verwalter des Benutzers	•	•	•	•
password	string	Klartext-Passwort des Benutzers		•	•	
realName	string	Voller Name des Benutzers	•	•	•	•
setKeys	stringlist	Liste der mit `set` setzbaren Parameter	•			
userLocked	bool	Gibt an, ob der Benutzer gesperrt ist	•	•	•	•
externalAttribute	von Attribut	Wert eines Benutzerfeldes	•	•	•	•

Verwalter eines Benutzers

Wenn ein Benutzer angelegt wird, wird der eingeloggte Benutzer automatisch zum direkten Verwalter (*owner*) des neuen Benutzers. Der Verwalter hat das Recht, Felder und Rechte des neuen Benutzers zu ändern. Das Login des direkten Verwalters ist im Parameter `owner` des Benutzers eingetragen.

Der Verwalter eines Benutzers hat auch die Möglichkeit, einen anderen Wert in den Parameter `owner` des Benutzers einzutragen. Dieser andere Wert kann entweder ein Login oder ein Gruppenname sein. Mit der Eintragung eines neuen Verwalters tritt der vorherige Verwalter seine Rechte an einen anderen Benutzer oder an eine Gruppe ab.

Ist im Parameter `owner` eines Benutzers eine Gruppe eingetragen, so gelten alle Mitglieder der Gruppe als indirekte Verwalter.

Um die Parameter eines Benutzers ändern zu können, muss man der direkte oder der indirekte Verwalter des Benutzers sein und das Recht `permissionGlobalUserEdit` haben. Ein Benutzer kann also seine Daten nicht selbst ändern, es sei denn, er ist sein eigener direkter oder indirekter Verwalter.

Die Wirkungsweise des Parameters `owner` gilt analog auch für Gruppen.

4.18.2 Benutzerbefehle

user create

Verfügbar für: Content Management Server

Aufgabe: Der Befehl legt einen neuen Benutzer mit den angegebenen Parameterwerten an.

Zusatzinformationen:

- Beim Anlegen eines Benutzers muss der Parameter `login` gesetzt werden. `login` kann nicht nachträglich geändert werden.
- Der `owner` des Benutzers wird auf den eingeloggten Benutzer gesetzt, falls `owner` nicht beim Aufruf von `create` als Parameter angegeben ist.

Syntax:

`user create {`*`parameter value`*`}`

Funktionsparameter:

- *parameter* spezifiziert den Namen des User-Parameters, dessen Wert bei der Erzeugung gesetzt werden soll (siehe Die User-Parameter). Die User-Parameter `login` und `defaultGroup` müssen angegeben werden.
- *`value`* ist der zu setzende Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: das Login des neuen Benutzers (string).

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalUserEdit` haben. Er muss ferner der Verwalter der Standardgruppe des neuen Benutzers sein.

Beispiel:

```
CM>user create login katrin realName {Katrin Landsberg} \
defaultGroup admins
katrin
```

user list

Verfügbar für: Content Management Server

Aufgabe: Listet die Logins sämtlicher verwalteter Benutzer auf.

Syntax:

`user list`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: Liste der Logins der Benutzer (stringlist).

Erforderliche Rechte: keine.

Beispiel:

```
CM>user list
admin katrin lange michael root walter
```

user where

Verfügbar für: Content Management Server

Aufgabe: Listet die Logins sämtlicher verwalteter Benutzer auf, in deren Name oder Login der angegebene String vorkommt.

Syntax:

`user where {`*`parameter value`*`}`

Funktionsparameter:

- *`parameter`* spezifiziert den Namen des Parameters, nach dessen Wert gefragt werden soll oder legt die Anzahl der Suchergebnisse fest. Folgende Parameter sind erlaubt:
  - `maxResults` gibt an, dass *`value`* eine Zahl enthält, die die maximale Anzahl Suchergebnisse festlegt. Ist diese Zahl kleiner oder gleich null, ist die Anzahl der Suchergebnisse nicht beschränkt.
  - `userText` gibt an, dass *`value`* eine Zeichenkette ist, nach der in den User-Parametern `name` und `login` gesucht werden soll.
- *`value`* ist der Wert des entsprechenden Parameters.

Rückgabewert bei Erfolg: Liste der Logins der Benutzer (stringlist).

Erforderliche Rechte: keine.

Beispiel:

```
CM>user where userText michael
lange michael
```

user userRef addToGroups

Verfügbar für: Content Management Server

Aufgabe: Nimmt den Benutzer mit dem Login *`login`* in jede der angegebenen Gruppen auf, falls er nicht schon Mitglied ist.

Zusatzinformationen: Durch diese Aktion wird nicht der Benutzer *`login`* verändert, sondern die angegebenen Gruppen. Daher sind zur Ausführung des Kommandos die Rechte zur Modifikation der Gruppen notwendig.

Syntax:

`user withLogin `*`login`*` addToGroups {`*`groupName`*`}`

Funktionsparameter:

- *groupName* ist der Name einer vorhandenen Benutzergruppe. Es können beliebig viele Gruppen angegeben werden.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte:

- Der eingeloggte Benutzer muss Superuser sein, oder
- der eingeloggte Benutzer muss das Recht `permissionGlobalUserEdit` haben und direkter oder indirekter Verwalter (`owner`) jeder Gruppe sein, in die der Benutzer *login* aufgenommen werden soll.

Beispiel:

```
CM>user withLogin katrin addToGroups editors
```

user userRef checkPassword

Verfügbar für: Content Management Server

Aufgabe: Prüft, ob der Benutzer mit dem Login *login* das angegebene Passwort hat.

Syntax:

`user withLogin` *login* `checkPassword` *password*

Funktionsparameter:

- *password*: Ein unverschlüsseltes Passwort.

Rückgabewert bei Erfolg: 1, wenn das aus *password* entstehende verschlüsselte Passwort mit dem `encryptedPassword` des Benutzers übereinstimmt, andernfalls 0.

Erforderliche Rechte: keine.

Beispiel:

```
CM>user withLogin katrin checkPassword za1WpM
0
```

user userRef delete

Verfügbar für: Content Management Server

Aufgabe: Löscht den Benutzer mit dem Login *login*.

Zusatzinformationen: War der Benutzer der Verwalter anderer Benutzer oder von Benutzergruppen, so wird als deren neuer Verwalter der Benutzer `root` eingetragen.

Syntax:

`user withLogin` *login* `delete`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte:

- Der eingeloggte Benutzer muss Superuser sein, oder
- der eingeloggte Benutzer muss das Recht `permissionGlobalUserEdit` haben und direkter oder indirekter `owner` des Benutzers *`login`* sein.

Beispiel:

```
CM>user withLogin katrin delete
```

user userRef description

Verfügbar für: Content Management Server

Aufgabe: Liefert eine String-Repräsentation der Daten des Benutzers mit dem Login *`login`*.

Zusatzinformationen: die Repräsentation wird im Property-List-Format eines Dictionarys formatiert (siehe Das Property-List-Format). Die dargestellten Parameter sind im Abschnitt Die User-Parameter aufgeführt.

Syntax:

`user withLogin` *`login`* `description`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: String-Repräsentation des Benutzers (string).

Erforderliche Rechte: keine.

Beispiel:

```
CM>user withLogin katrin description
{
  defaultGroup = admins;
  globalPermissions = (
  );
  groups = (
    admins
  );
  login = katrin;
  owner = root;
  realName = "Katrin\ Landsberg";
  userLocked = 0;
}
```

user userRef get

Verfügbar für: Content Management Server

Aufgabe: Liefert den Wert des angegebenen User-Parameters des Benutzers mit dem Login *`login`*.

Syntax:

`user withLogin login get paramName`

Funktionsparameter:

- *paramName* spezifiziert den Namen des Parameters, dessen Wert gesucht wird (siehe Die User-Parameter).

Rückgabewert bei Erfolg: Wert des angegebenen User-Parameters (string).

Erforderliche Rechte: keine.

Beispiel:

```
CM>user withLogin katrin get realName
Katrin Landsberg
```

user userRef grantGlobalPermissions

Verfügbar für: Content Management Server

Aufgabe: Erteilt dem Benutzer mit dem Login *login* die angegebenen Rechte.

Syntax:

`user withLogin login grantGlobalPermissions {permission}`

Funktionsparameter:

- *permission* Die Bezeichnung für ein globales Recht. Welche globalen Rechte zulässig sind, ist in der Systemkonfiguration unter dem Schlüssel `globalPermissions` eingetragen.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte:

- Der eingeloggte Benutzer muss Superuser sein, oder
- der eingeloggte Benutzer muss das Recht `permissionGlobalUserEdit` haben und direkter oder indirekter `owner` des Benutzers *login* sein.

Beispiel:

```
CM>user withLogin katrin grantGlobalPermissions permissionGlobalRoot
```

user userRef hasGlobalPermission

Verfügbar für: Content Management Server

Aufgabe: Prüft, ob der Benutzer mit dem Login *login* das angegebene globale Recht hat.

Zusatzinformationen: Ein Benutzer hat ein bestimmtes Recht, wenn er selbst das Recht `permission` hat, wenn er Superuser ist oder wenn er Mitglied einer Gruppe ist, die dieses Recht hat.

Syntax:

`user withLogin login hasGlobalPermission permission`

Funktionsparameter:

- *permission* Die Bezeichnung für ein globales Recht.

Rückgabewert bei Erfolg: 1, wenn der Benutzer das angegebene Recht hat, sonst 0.

Erforderliche Rechte: keine.

Beispiel:

```
CM>user withLogin katrin hasGlobalPermission permissionGlobalRTCEdit1
```

user userRef isOwnerOf

Verfügbar für: Content Management Server

Aufgabe: Prüft, ob der Benutzer *login* direkter oder indirekter Verwalter (owner) des Benutzers mit dem Login oder der Gruppe mit dem Namen *ownedLoginOrGroup* ist (siehe Verwalter eines Benutzers).

Syntax:

`user withLogin login isOwnerOf ownedLoginOrGroup`

Funktionsparameter:

- *ownedLoginOrGroup* Login für den gefragt wird, ob sein owner *login* ist.

Rückgabewert bei Erfolg: 1, wenn der Benutzer *login* direkter oder indirekter owner des Benutzers mit dem Login oder der Gruppe mit dem Namen *ownedLoginOrGroup* ist, sonst 0.

Erforderliche Rechte: keine.

Beispiel:

```
CM>user withLogin michael isOwnerOf katrin
1
```

user userRef isSuperUser

Verfügbar für: Content Management Server

Aufgabe: Prüft, ob der Benutzer mit dem Login *login* Superuser ist.

Zusatzinformationen: Ein Benutzer ist Superuser, wenn sein Login root ist, er das Recht permissionGlobalRoot hat oder wenn er Mitglied einer Gruppe ist, die dieses Recht hat. Dieser Befehl hat dasselbe Ergebnis wie der Befehl `user withLogin login hasGlobalPermission permissionGlobalRoot`.

Syntax:

`user withLogin login isSuperUser`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: 1, wenn der Benutzer Superuser ist, sonst 0.

Erforderliche Rechte: keine.

Beispiel:

```
CM>user withLogin katrin isSuperUser
1
```

user userRef mget

Verfügbar für: Content Management Server

Aufgabe: Liefert die Werte der angegebenen User-Parameter des Benutzers mit dem Login *login*.

Syntax:

`user withLogin` *login* `mget {`*paramName*`}`

Funktionsparameter:

- *paramName* spezifiziert den Namen des Parameters, dessen Wert gesucht wird.

Rückgabewert bei Erfolg: Liste der Parameterwerte (stringlist).

Erforderliche Rechte: keine.

Beispiel:

```
CM>user withLogin katrin mget defaultGroup userLocked
admins 0
```

user userRef removeFromGroups

Verfügbar für: Content Management Server

Aufgabe: Entfernt den Benutzer mit dem Login *login* aus den angegebenen Gruppen.

Zusatzinformationen: Ein Benutzer kann nicht aus seiner Standardgruppe entfernt werden.

Syntax:

`user withLogin` *login* `removeFromGroups {`*groupName*`}`

Funktionsparameter:

- *groupName* Name einer Gruppe, aus der der Benutzer entfernt werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte:

- Der eingeloggte Benutzer muss Superuser sein, oder

- der eingeloggte Benutzer muss das Recht `permissionGlobalUserEdit` haben und direkter oder indirekter `owner` jeder Gruppe sein, aus der der Benutzer *`login`* entfernt werden soll.

Beispiel:

```
CM>user withLogin katrin removeFromGroups admins
```

user userRef revokeGlobalPermissions

Verfügbar für: Content Management Server

Aufgabe: Entzieht dem Benutzer mit dem Login *`login`* die angegebenen Rechte.

Syntax:

`user withLogin` *`login`* `revokeGlobalPermissions {`*`permission`*`}`

Funktionsparameter:

- *`permission`* bezeichnet ein globales Recht.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte:

- Der eingeloggte Benutzer muss Superuser sein, oder
- der eingeloggte Benutzer muss das Recht *`permissionGlobalUserEdit`* haben und direkter oder indirekter Verwalter (`owner`) des Benutzers *`login`* sein.

Beispiel:

```
CM>user withLogin katrin revokeGlobalPermissions permissionGlobalRTCEdit
```

user userRef set

Verfügbar für: Content Management Server

Aufgabe: Setzt beim Benutzer mit dem Login *`login`* die angegebenen Parameter auf die entsprechenden Werte.

Zusatzinformationen:

- Wird der Parameter `owner` explizit auf den Leerstring gesetzt, so wird der eingeloggte Benutzer als `owner` eingesetzt.

Syntax:

`user withLogin` *`login`* `set {`*`paramName value`*`}`

Funktionsparameter:

- *`paramName`* spezifiziert den Namen des User-Parameters, dessen Wert gesetzt werden soll (siehe Die User-Parameter).
- *`value`* ist der zu setzende Wert des angegebenen Parameters.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte:

- Der eingeloggte Benutzer muss Superuser sein, oder
- der eingeloggte Benutzer muss das Recht `permissionGlobalUserEdit` haben und direkter oder indirekter Verwalter (`owner`) des Benutzers *`login`* sein.
- Ist der eingeloggte Benutzer kein Superuser und hat auch nicht das Recht `permissionGlobalUserEdit`, so kann er nur seine eigenen User-Parameter `defaultGroup`, `email`, `password` und `realName` sowie die Werte seiner Benutzerfelder modifizieren. Als neue `defaultGroup` kann er nur eine Gruppe angeben, in der er Mitglied ist.

Beispiel:

```
CM>user withLogin katrin set userLocked 0
```

4.19 userAttribute (Benutzerfelder)

4.19.1 Benutzerfeldparameter

Parameter	Typ	Erklärung	get	set	create	descr
displayTitle	string	Der in der HTML-Benutzerschnittstelle angezeigte Titel des Benutzerfeldes (entspricht dem Namen)	•			
getKeys	stringlist	Liste der mit `get` abfragbaren Parameter	•			
name	string	Name des Benutzerfeldes	•		•	•
setKeys	stringlist	Liste der mit `set` setzbaren Parameter	•			
type	string	Typ des Benutzerfeldes (zulässig: string (default), date, enum, multienum)	•	•	•	•
values	stringlist	Aufzählungswerte (nur bei *enum*- und *multienum*- Benutzerfeldern)	•	•		•

4.19.2 Benutzerfeldbefehle

userAttribute create

Verfügbar für: Content Management Server

Aufgabe: der Befehl erzeugt ein neues Benutzerfeld.

Syntax:

`userAttribute create {`*`parameter value`*`}`

Funktionsparameter:

- *parameter* ist der Name eines Parameters, dessen Wert bei der Erzeugung des Feldes durch *value* gesetzt wird. Es muss mindestens der Name des Feldes angegeben werden!
- *value* ist der zu setzende Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: der Name des Benutzerfeldes (string).

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalUserAttributeEdit` haben.

Beispiel:

```
CM>userAttribute create name address
address
```

userAttribute list

Verfügbar für: Content Management Server

Aufgabe: Der Befehl gibt die Liste der Namen sämtlicher kundenspezifischer Benutzerfelder aus.

Syntax:

```
userAttribute list
```

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die Liste der Benutzerfeldnamen (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>userAttribute list
address phone
```

userAttribute types

Verfügbar für: Content Management Server

Aufgabe: Der Befehl gibt die Liste der verfügbaren Benutzerfeldtypen aus.

Syntax:

```
userAttribute types
```

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die Liste der Benutzerfeldtypen (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>userAttribute types
```

```
enum string date multienum
```

userAttribute where

Verfügbar für: Content Management Server

Aufgabe: Ermöglicht die Suche nach allen Namen von Benutzerfeldern, bei denen der Wert der angegebenen Parameter den jeweils spezifizierten String enthält.

Syntax:

`userAttribute where {`*`parameter value`*`}`

Funktionsparameter:

- *`parameter`* spezifiziert den Namen des Parameters, nach dessen Wert gefragt werden soll oder legt die Anzahl der Suchergebnisse fest. Folgende Parameter sind erlaubt:
  - `maxResults` gibt an, dass *`value`* eine Zahl enthält, die die maximale Anzahl Suchergebnisse festlegt. Ist diese Zahl kleiner oder gleich null, ist die Anzahl der Suchergebnisse nicht beschränkt.
  - `name`: die Namen der Benutzerfelder sollen auf das Vorkommen von *`value`* untersucht werden.
- *`value`* enthält den Wert des jeweiligen Parameters.

Rückgabewert bei Erfolg: die Liste der Namen der passenden Benutzerfelder (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>userAttribute where name phone
phone phone_car phone_home
```

userAttribute userAttrRef addEnumValues

Verfügbar für: Content Management Server

Aufgabe: Fügt zu einem Aufzählungs- (`enum`) oder Mehrfachaufzählungs-Benutzerfeld (`multienum`) die angegebenen Aufzählungswerte hinzu, falls diese nicht bereits existieren.

Syntax:

`userAttribute withName `*`attrName`*` addEnumValues {`*`enumValue`*`}`

Funktionsparameter:

- *`enumValue`*: gibt einen Aufzählungswert an, der zu den bereits existierenden Aufzählungswerten des Benutzerfeldes hinzugefügt werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: der Benutzer muss das Recht `permissionGlobalUserAttributeEdit` haben.

Beispiel:

```
CM>userAttribute withName department addEnumValues \
```

```
administration development marketing services
```

userAttribute userAttrRef delete

Verfügbar für: Content Management Server

Aufgabe: Der Befehl löscht das angegebene Benutzerfeld.

Syntax:

`userAttribute withName` *`attrName`* `delete`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalUserAttributeEdit` haben.

Beispiel:

```
CM>userAttribute withName address delete
```

userAttribute userAttrRef description

Verfügbar für: Content Management Server

Aufgabe: Liefert eine String-Repräsentation der Daten des Benutzerfeldes mit dem angegebenen Namen.

Zusatzinformationen: die String-Repräsentation wird im Property-List-Format eines Dictionarys formatiert (siehe Das Property-List-Format).

Syntax:

`userAttribute withName` *`attrName`* `description`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die String-Repräsentation des Benutzerfeldes (string).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>userAttribute withName department description
{
  name = department;
  type = multienum;
  values = (
    administration,
    development,
    marketing,
    services
  );
}
```

userAttribute userAttrRef get

Verfügbar für: Content Management Server

Aufgabe: Liefert den Wert eines Benutzerfeldparameters.

Syntax:

`userAttribute withName *attrName* get *parameter*`

Funktionsparameter:

- *parameter* spezifiziert den Namen des Parameters, dessen Wert gesucht wird (siehe Benutzerfeld-Parameter).

Rückgabewert bei Erfolg: der Wert des angegebenen Parameters (string).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>userAttribute withName department get type
multienum
```

userAttribute userAttrRef mget

Verfügbar für: Content Management Server

Aufgabe: Liefert Parameterwerte eines Benutzerfeldes.

Syntax:

`userAttribute withName *attrName* mget {*parameter*}`

Funktionsparameter:

- *parameter* spezifiziert den Namen des Parameters, dessen Wert gesucht wird (siehe Benutzerfeld-Parameter).

Rückgabewert bei Erfolg: die Liste der Werte der betreffenden Parameter (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>userAttribute withName department mget type values
multienum {administration development marketing services}
```

userAttribute userAttrRef removeEnumValues

Verfügbar für: Content Management Server

Aufgabe: Löscht bei einem enum- oder multienum-Benutzerfeld die angegebenen Aufzählungswerte.

Syntax:

`userAttribute withName attrName removeEnumValues {enumValue}`

Funktionsparameter:

- *enumValue* gibt einen Aufzählungswert an, der aus den existierenden Aufzählungswerten des Benutzerfeldes gelöscht werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalUserAttributeEdit` haben.

Beispiel:

```
CM>userAttribute withName department removeEnumValues sales services
```

userAttribute userAttrRef set

Verfügbar für: Content Management Server

Aufgabe: Setzt Parameterwerte des Benutzerfeldes mit dem Namen *attrName*.

Syntax:

`userAttribute withName attrName set {parameter value}`

Funktionsparameter:

- *parameter*: spezifiziert den Parameter des Benutzerfeldes, dessen Wert gesetzt werden soll (siehe Benutzerfeld-Parameter).
- *value* ist der zu setzende Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalUserAttributeEdit` haben.

Beispiel:

```
CM>userAttribute withName department set type enum
```

4.20 userConfig (persönliche Einstellungen)

4.20.1 userConfig

Verfügbar für: Content Management Server

Aufgabe: Der `userConfig`-Befehl ermöglicht den schreibenden und lesenden Zugriff auf die Einstellungen des angemeldeten Benutzers. Die Voreinstellungen hierfür sind in der Systemkonfiguration unterhalb der Elemente `userManagement.preferences` und `userManagement.preferences.guiPreferences` gespeichert. Änderungen, die ein Benutzer an seinen Voreinstellungen vornimmt, werden dauerhaft gespeichert.

Zusatzinformationen:

- Bei den Subkommandos `parseInputDate` und `formatDateTime` werden die benutzerspezifischen Konfigurationswerte verwendet.
- Ein Superuser kann über das Kommando `sudo` die Einstellungen jedes Benutzers ändern.

Syntax:

`userConfig` *`command`*

Funktionsparameter:

- `command` ist ein Subkommando. Der `userConfig`-Befehl verfügt über dieselben Subkommandos wie der `systemConfig`-Befehl. Zusätzlich steht beim `userConfig`-Befehl das Subkommando `getAll` zur Verfügung. Mit diesem Kommando kann das komplette benutzerspezifische Konfigurationselement ausgelesen werden.

Beispiel:

```
CM>userConfig getTexts timeZone language
MET de
```

4.21 userConfigForUser (persönliche Einstellungen anderer Benutzer)

4.21.1 userConfigForUser

Verfügbar für: Content Management Server

Aufgabe: Der `userConfigForUser`-Befehl funktioniert analog zum `userConfig`-Befehl. Während dieser jedoch den Zugriff auf die persönlichen Einstellungen des eingeloggten Benutzers erlaubt, können Sie mit dem `userConfigForUser`-Befehl die Einstellungen eines jeden Benutzers, dessen Verwalter Sie sind, auslesen und setzen.

Syntax:

`userConfigForUser` *`login command`*

Funktionsparameter:

- *`login`* ist der Anmeldename des Benutzers, dessen persönliche Einstellungen Sie auslesen oder setzen möchten.
- *`command`* ist ein Subkommando. Der `userConfigForUser`-Befehl verfügt über dieselben Subkommandos wie der `systemConfig`-Befehl.

Erforderliche Rechte: Sie können mit dem Befehl nur auf Ihre eigenen und die Einstellungen der Benutzer zugreifen, deren Verwalter Sie sind. Als Superuser sind Sie implizit der Verwalter aller Benutzer.

Beispiel:

```
CM>userConfigForUser mustermann setElements guiPreferences.startArea
 <startArea>wizard_page</startArea>
CM>userConfigForUser mustermann getElements guiPreferences.startArea
```

```
{<?xml version="1.0" encoding="UTF-8"?>
<startArea>browse_page</startArea>}
```

4.22 workflow (Workflows)

4.22.1 Workflow-Parameter

Parameter	Typ	Erklärung	get	set	create	descr
allowsMultiple Signatures	bool	gibt an, ob ein und dieselbe Person eine Version mehrfach signieren darf.	•	•	•	•
displayTitle	string	Der in der HTML-Benutzerschnittstelle angezeigte Titel des Workflows	•			
editGroups	stringlist	Die Gruppen im Bearbeitungs-Workflow	•	•	•	•
getKeys	stringlist	Liste der mit `get` abfragbaren Parameter	•			
isEnabled	bool	gibt an, ob neue Workflows dieses Typs begonnen werden dürfen	•	•	•	•
name	string	Name des Workflows	•		•	•
setKeys	stringlist	die Liste der mit `set` setzbaren Parameter	•			
signatureDefs	stringlist	Die Unterschriften und unterschriftsberechtigten Gruppen in der Abzeichnungsphase des Workflows. Jeder String ist eine 2-elementige Liste aus Feldname und Gruppe	•	•	•	•
title	string	Der Titel des Workflows in der benutzerspezifischen Sprache	•	•	•	•
title.*language*	string	Der Titel des Workflows in der jeweiligen Sprache.	•	•	•	•

4.22.2 Workflow-Befehle

workflow create

Verfügbar für: Content Management Server

Aufgabe: Der Befehl legt einen neuen Workflow an.

Syntax:

`workflow create {`*`parameter value`*`}`

Funktionsparameter:

- *parameter* spezifiziert den Namen des Workflow-Parameters, dessen Wert beim Erzeugen gesetzt werden soll (siehe Workflow-Parameter).
- *value* ist der zu setzende Wert des betreffenden Workflow-Parameters.

Rückgabewert bei Erfolg: der Name des Workflows (string).

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel:

```
CM>workflow create name wf_sig_news editGroups news_editors \
signatureDefs {{sig_news newsadmins}}
wf_sig_news
```

workflow list

Verfügbar für: Content Management Server

Aufgabe: Der Befehl listet die Namen sämtlicher Workflows auf.

Syntax:

`workflow list`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die Liste der Workflownamen (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>workflow list
wf_edit_news wf_sig_news
```

workflow where

Verfügbar für: Content Management Server

Aufgabe: Ermöglicht die Suche nach allen Workflownamen, die den angegebenen String enthalten.

Syntax:

`workflow where {`*parameter value*`}`

Funktionsparameter:

- *parameter* spezifiziert den Namen des Parameters, nach dessen Wert gefragt werden soll oder legt die Anzahl der Suchergebnisse fest. Folgende Parameter sind erlaubt:
  - `maxResults` gibt an, dass *value* eine Zahl enthält, die die maximale Anzahl Suchergebnisse festlegt. Ist diese Zahl kleiner oder gleich null, ist die Anzahl der Suchergebnisse nicht beschränkt.
  - `name` gibt an, dass *value* der Substring ist, nach dessen Vorkommen in den Workflow-Namen gesucht wird.

- *value* ist der Wert des betreffenden Parameters.

Rückgabewert bei Erfolg: die Liste der Namen der passenden Workflows (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>workflow where name news
wf_edit_news wf_sig_news
```

workflow workflowRef delete

Verfügbar für: Content Management Server

Aufgabe: Der Befehl löscht den angegebenen Workflow.

Syntax:

`workflow withName` *`workflowName`* `delete`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel:

```
CM>workflow withName wf_edit_news delete
```

workflow workflowRef description

Verfügbar für: Content Management Server

Aufgabe: Liefert eine String-Repräsentation der Daten des Workflows mit dem angegebenen Namen.

Zusatzinformationen: die Repräsentation wird im Property-List-Format eines Dictionarys formatiert (siehe Das Property-List-Format).

Syntax:

`workflow withName` *`workflowName`* `description`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die String-Repräsentation des Workflows (string).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>workflow withName wf_sig_news description
{
  allowsMultipleSignatures = 1;
```

```
  editGroups = (
    editors
  );
  isEnabled = 1;
  signatureDefs = (
    (
      "sig_news",
      newsadmins
    )
  );
}
```

workflow workflowRef get

Verfügbar für: Content Management Server

Aufgabe: Der Befehl liefert einen Parameterwert des Workflows mit dem Namen *workflowName*.

Syntax:

`workflow withName` *workflowName* `get` *parameter*

Funktionsparameter:

- *parameter* spezifiziert den Namen des Parameters, dessen Wert gesucht wird (siehe Workflow-Parameter).

Rückgabewert bei Erfolg: der Wert des angegebenen Parameters (string).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>workflow withName wf_sig_news get isEnabled
1
```

workflow workflowRef mget

Verfügbar für: Content Management Server

Aufgabe: Der Befehl liefert Parameterwerte des Workflows mit dem Namen *workflowName*.

Syntax:

`workflow withName` *workflowName* `mget {`*parameter*`}`

Funktionsparameter:

- *parameter* spezifiziert den Namen des Parameters, dessen Wert gesucht wird (siehe Workflow-Parameter).

Rückgabewert bei Erfolg: die Liste der Werte der betreffenden Parameter (stringlist).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>workflow withName wf_sig_news mget editGroups signatureDefs
editors {{sig_news newsadmins}}
```

workflow workflowRef set

Verfügbar für: Content Management Server

Aufgabe: Der Befehl setzt die Werte der angegebenen Parameter des Workflows *workflowName*.

Syntax:

`workflow withName` *workflowName* `set {`*parameter value*`}`

Funktionsparameter:

- *parameter* spezifiziert den Parameter des Workflows, dessen Wert gesetzt werden soll (siehe Workflow-Parameter).
- *value* ist der zu setzende Wert des angegebenen Parameters.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel:

```
CM>workflow withName wf_sig_news set isEnabled 0 title {News workflow}
```

4.23 Callback-Funktionen

4.23.1 exportFillCallback

Verfügbar für: Template Engine

Aufgabe: Kundenspezifische Prozedur, die die Export-Verzeichnishierarchie der Template Engine mit der entsprechenden Verzeichnishierarchie auf dem Live-Server synchronisiert. Die Funktion kann dies erreichen, indem sie beispielsweise `rsync` ausführt.

Zusatzinformationen: siehe Import und Export.

Syntax:

`exportFillCallback` *offlineTree onlineTree*

Funktionsparameter:

- *offlineTree*: Der Pfad der Export-Verzeichnishierarchie, der sich aus dem symbolischen Link `offline` im instanzenspezifischen Verzeichnis `export` ergibt.
- *onlineTree*: Der Pfad der Export-Verzeichnishierarchie, der sich aus dem symbolischen Link `online` im instanzenspezifischen Verzeichnis `export` ergibt.

Rückgabewert bei Erfolg: keiner

4.23.2 exportSwitchCallback

Verfügbar für: Template Engine

Aufgabe: Kundenspezifische Prozedur, die die Export- und Live-Verzeichnishierarchien auf dem Live-Server (beispielsweise über eine sichere Verbindung) vertauscht und gegebenenfalls weitere Aktionen ausführt.

Zusatzinformationen: siehe Import und Export.

Syntax:

`exportSwitchCallback` *`offlineTree onlineTree`*

Funktionsparameter:

- *`offlineTree`*: Der Pfad der Export-Verzeichnishierarchie, der sich aus dem symbolischen Link `offline` im instanzenspezifischen Verzeichnis `export` ergibt.
- *`onlineTree`*: Der Pfad der Export-Verzeichnishierarchie, der sich aus dem symbolischen Link `online` im instanzenspezifischen Verzeichnis `export` ergibt.

Rückgabewert bei Erfolg: keiner

4.23.3 exportSyncCallback

Verfügbar für: Template Engine

Aufgabe: Kundenspezifische Prozedur, die die entfernte Export-Verzeichnishierarchie auf den Stand der Live-Verzeichnishierarchie bringt, d. h. die beiden Hierarchien synchronisiert.

Zusatzinformationen: siehe Import und Export.

Syntax:

`exportSyncCallback` *`offlineTree onlineTree`*

Funktionsparameter:

- *`offlineTree`*: Der Pfad der Export-Verzeichnishierarchie, der sich aus dem symbolischen Link `offline` im instanzenspezifischen Verzeichnis `export` ergibt.
- *`onlineTree`*: Der Pfad der Export-Verzeichnishierarchie, der sich aus dem symbolischen Link `online` im instanzenspezifischen Verzeichnis `export` ergibt.

Rückgabewert bei Erfolg: keiner

4.23.4 importCallback

Verfügbar für: Template Engine (ab Version 6.7.2)

Aufgabe: Kundenspezifische Prozedur, die das Import-Journal verarbeitet. Im Zuge dieser Vararbeitung können beispielsweise weitere Dateien für den Export markiert werden (siehe den Befehl obj touch).

Zusatzinformationen: siehe Import und Export.

Syntax:

`importCallback` *`importJournal`*

Funktionsparameter:

- *`importJournal`*: Der Pfad zum Import-Journal. Die Funktion darf die Datei nicht modifizieren.

Rückgabewert bei Erfolg: keiner

Das Import-Journal

Das Import-Journal ist eine Aufzeichnung aller für eine Datei relevanten Aktionen während der Import-Phase. Das Journal enthält pro Zeile einen Eintrag der Form

```
Code Dateipfad
```

`Code` ist ein einzelnes Zeichen, das die Art der Änderung spezifiziert. Der *`Dateipfad`* ist der CMS-Pfad der betreffenden Datei. Zeilen, die mit einem Doppelkreuz (#) beginnen, sind Kommentare.

Das Import-Journal zeichnet die folgenden Änderungen auf und markiert sie mit dem jeweils angegebenen Code:

- `c`: Die Datei mit dem angebenen Pfad wurde angelegt.
- `u`: Ein Dateifeld (beispielsweise `suppressExport`) wurde geändert. Dieser Typ wird nicht bei Änderungen des Namens oder des darüber liegenden Ordners (`parent`) aufgezeichnet. Stattdessen werden dafür die Typen `m` und `M` aufgezeichnet.
- `r`: Die Datei wurde gelöscht.
- `m`: Die Datei wurde verschoben (der Pfad hat sich geändert). Der angegebene Pfad ist der bisherige Pfad der Datei. Auf diesen Eintrag folgt stets unmittelbar ein `M`-Eintrag.
- `M`: Die Datei wurde verschoben (der Pfad hat sich geändert). Der angegebene Pfad ist der neue Pfad der Datei. Dieser Eintrag folgt stets unmittelbar auf einen `m`-Eintrag.
- `U`: Der Inhalt der Datei wurde geändert (die Datei wurde freigegeben).
- `R`: Der Inhalt der Datei wurde gelöscht (die Datei wurde zurückgezogen).

Das Import-Journal wird bei erfolgreicher Durchführung der Phase `importCB` geleert. Liefert der Import-Callback einen Fehler, so bleiben alle Einträge im Journal erhalten. Neue Einträge werden ans Ende des Journals angehängt.

Beispiel eines Import-Journals:

```
# This file is generated automatically.
# Only the TE is allowed to modify it.
u /
c /internet
c /internet/playland
c /internet/playland/de
c /internet/playland/de/wirueberuns
c /internet/playland/de/wirueberuns/faq
c /internet/playland/de/wirueberuns/faq/probiermodus
c /internet/playland/de/wirueberuns/faq/schadstofffreiheit
c /internet/playland/de/wirueberuns/faq/bestellung
c /internet/playland/de/wirueberuns/faq/verfuegbarkeit
U /
U /internet
U /internet/playland
U /internet/playland/de
U /internet/playland/de/wirueberuns
U /internet/playland/de/wirueberuns/faq
```

```
U /internet/playland/de/wirueberuns/faq/probiermodus
U /internet/playland/de/wirueberuns/faq/schadstofffreiheit
U /internet/playland/de/wirueberuns/faq/bestellung
U /internet/playland/de/wirueberuns/faq/verfuegbarkeit
```

4.24 Andere administrative Anweisungen

4.24.1 app closeDbConnection

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Schließt die Verbindung zur Datenbank.

Syntax:

`app closeDbConnection`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: da dieser Befehl ausgeführt wird, bevor sich ein Benutzer einloggt, gibt es keine Möglichkeit, seine Ausführung von den Rechten eines CMS-Benutzers abhängig zu machen.

Zusatzinformationen: Dieser Befehl kann nur beim Applikationsstart und im Single-Modus verwendet werden.

Beispiel:

```
app closeDbConnection
```

4.24.2 app export

Verfügbar für: Template Engine

Aufgabe: Dieser Befehl schaltet die Template Engine von der Import-Phase in die Export-Phase, publiziert die Webdokumente jedoch nicht.

Zusatzinformationen:

- Der Befehl ist auch im Tcl-Client verfügbar und wird asynchron ausgeführt, d. h. kehrt sofort zurück.
- Der Befehl darf während der Export-Phase nicht ausgeführt werden.
- Nach der Phase `export-fill` kehrt die Template Engine in die Import-Phase zurück (siehe auch Funktionsweise der Template Engine).

Syntax:

`app export`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

4.24.3 app get

Verfügbar für: Content Management Server, Template Engine, Search Engine Server

Aufgabe: Liefert den Wert des angegebenen Systemparameters. Dieser Befehl ist auch im Tcl-Client verfügbar.

Syntax:

`app get` *`parameter`*

Funktionsparameter:

- *`parameter`*: der Name des Parameters, dessen Wert ermittelt werden soll. Zulässige Namen sind:
  - `phase` (nur TE): die Teilphase, in der sich die Template Engine befindet. Zur Erläuterung der Phasen siehe Funktionsweise der Template Engine.
  - `phaseDetails` (nur TE): liefert zum Zwecke der Fehlersuche zusätzlich zur aktuellen Teilphase Detailinformationen über den Fortschritt der Phase. Der Rückgabewert ist je nach Phase (siehe Parameter `phase`) folgendermaßen aufgebaut:
    - `import {objects` *`objectCount recursiveObjectCount`*`}`:
      `objectCount` gibt an, wieviele Dateien exportiert werden müssen. Zu Beginn der Import-Phase sind das alle Dateien, die in der letzten Export-Phase nicht exportiert werden konnten. *`recursiveObjectCount`* gibt an, bei wievielen Dateien auch der darunter liegende Teilbaum exportiert werden muss. Zu Beginn der Import-Phase ist *`recursiveObjectCount`* gleich 0. Während der Import-Phase können diese Werte nur ansteigen.
    - `export-init {objects` *`objectCount recursiveObjectCount`*`}`:
      Die Werte haben die gleiche Bedeutung wie bei der Phase `import`. Zum Ende der Phase `export-init` ist *`recursiveObjectCount`* gleich 0.
    - `export-fill {`*`sliceType count percentage`*`}` ...:
      `sliceType` ist der Typ eines Ausschnitts aus der Liste der zu exportierenden Dateien. Für jeden der folgenden Typen wird eine Unterliste in die Rückgabeliste eingefügt: `available`, `inProgress`, `complete`, `failed`, `total`. Die Bezeichner haben die folgende Bedeutung:
      `count`: Aktuelle Anzahl Ausschnitte dieses Typs. Ist der *`sliceType`* gleich `total`, wird die Gesamtzahl aller Ausschnitte ausgegeben.
      `percentage`: gibt an, wieviel Prozent aller Ausschnitte diesen Typ haben. Der Prozentsatz beim Typ `total` ist immer 100.
    - Bei den restlichen Phasen liefert `phaseDetails` lediglich die Bezeichnung der Phase.
  - `today`: das aktuelle Datum (14 Stellen), 0.00 Uhr, in GMT.
  - `version`: die Versionszeichenkette, die man auch mit `-version` auf der Kommandozeile erhält.
  - `appName`: der Name der Applikation.
  - `rootConfigPath`: der Pfad zur Hauptkonfigurationsdatei `nps.xml` .
  - `timeZone`: die Zeitzone, die sich aus der Rechnerzeit ergibt.
  - `baseDir`: das Installationsverzeichnis von NPS.
  - `binDir`: das Verzeichnis unterhalb des Instanzenverzeichnisses, in dem sich die Startskripte befinden.
  - `commonScriptDir`: der absolute Pfad des instanzenspezifischen Verzeichnisses `script/common`.

- `configDir`: das Verzeichnis unterhalb des Instanzenverzeichnisses, in dem sich die Konfigurationsdateien befinden.
- `dataDir`: das Verzeichnis unterhalb des Instanzenverzeichnisses, in dem sich Daten wie Blobs und Streaming-Tickets befinden.
- `instanceDir`: das Instanzenverzeichnis.
- `libDir`: das Verzeichnis unterhalb des Installationsverzeichnisses, in dem sich die Bibliotheken und ausführbaren Dateien befinden.
- `logDir`: das Verzeichnis unterhalb des Instanzenverzeichnisses, in dem sich die Protokolldateien befinden.
- `scriptDir`: das Verzeichnis unterhalb des Instanzenverzeichnisses, in dem sich die Skripte befinden. Das Verzeichnis `cm` wird vom Content Manager und der Template Engine gemeinsam verwendet, das Verzeichnis `ses` vom Search Engine Server.
- `shareDir`: das Verzeichnis unterhalb des Installationsverzeichnisses, in dem sich die von allen Instanzen gemeinsam benötigten Daten befinden, beispielsweise die Dokumentation.
- `tmpDir`: das Verzeichnis unterhalb des Instanzenverzeichnisses, in dem sich temporäre Dateien wie beispielsweise PID-Dateien befinden.

Rückgabewert bei Erfolg:

- `phase` und `rank`: der Wert des Parameters (string).
- `phaseDetails`: Details über die jeweilige Phase (stringlist).
- `timeZone`: die Zeitzone (string).
- `*Dir`: der absolute Pfad des Verzeichnisses (string).

Erforderliche Rechte (nur CM): keine Einschränkungen.

4.24.4 app makeDbConnection

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Stellt eine Verbindung zur Datenbank her.

Syntax:

```
app makeDbConnection
```

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: da dieser Befehl ausgeführt wird, bevor sich ein Benutzer einloggt, gibt es keine Möglichkeit, seine Ausführung von den Rechten eines CMS-Benutzers abhängig zu machen.

Zusatzinformationen: Dieser Befehl kann nur beim Applikationsstart und im Single-Modus verwendet werden.

4.24.5 app publish

Verfügbar für: Template Engine

Aufgabe: Dieser Befehl initiiert einen kompletten Export-Zyklus.

Zusatzinformationen:

- Der Befehl ist auch im Tcl-Client verfügbar und wird asynchron ausgeführt, d. h. kehrt sofort zurück.
- Der Befehl kann bei Fehlern während des Exports verwendet werden, um den Export fortzusetzen. Siehe auch Funktionsweise der Template Engine.

Syntax:

```
app publish
```

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

4.24.6 clearUsermanCache

Verfügbar für: Content Management Server

Aufgabe: Das Kommando löscht sämtliche zwischengespeicherten Benutzerdaten.

Zusatzinformationen: Das Kommando sollte verwendet werden, wenn ein externer Benutzermanager (über die Interface-Datei `usermanAPI.tcl`) an den Content Manager angebunden wurde und Benutzerdaten in diesem Benutzermanager geändert wurden. Der gelöschte Cache wird anschließend mit aktuellen Benutzerdaten gefüllt, sobald solche Daten angefordert werden.

Syntax:

```
clearUsermanCache
```

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: keine Einschränkungen.

4.24.7 decodeData

Verfügbar für: Content Management Server, Template Engine, Search Engine Server

Aufgabe: Dekodiert den angegebenen base64-kodierten, binären String.

Syntax:

`decodeData` *`encodedString`*

Funktionsparameter:

- *`encodedString`* ist der base64-kodierte String.

Rückgabewert bei Erfolg: die dekodierte Zeichenkette.

Erforderliche Rechte (nur CM): keine Einschränkungen.

4.24.8 decodeFile

Verfügbar für: Content Management Server, Template Engine, Search Engine Server

Aufgabe: Dekodiert den angegebenen base64-kodierten String und schreibt ihn in die angegebene Datei.

Syntax:

`decodeFile` *`filename encodedString`*

Funktionsparameter:

- *`filename`* ist der Name der Datei.
- *`encodedString`* ist der base64-kodierte String.

Rückgabewert bei Erfolg: 1

Erforderliche Rechte (nur CM): keine Einschränkungen.

4.24.9 decodeToString

Verfügbar für: Content Management Server, Search Engine Server

Aufgabe: Dekodiert die angegebene base64-kodierte UTF-8-Zeichenkette.

Syntax:

`decodeToString` *`encodedString`*

Funktionsparameter:

- *`encodedString`* ist der base64-kodierte String.

Rückgabewert bei Erfolg: Die dekodierte Zeichenkette.

Erforderliche Rechte (nur CM): keine Einschränkungen.

4.24.10 encodeData

Verfügbar für: Content Management Server, Template Engine, Search Engine Server

Aufgabe: Gibt einen String base64-kodiert zurück.

Syntax:

`encodeData` *`stringToEncode`*

Funktionsparameter:

- *`stringToEncode`* ist der zu kodierende String.

Rückgabewert bei Erfolg: der base64-kodierte String.

Erforderliche Rechte (nur CM): keine Einschränkungen.

4.24.11 encodeFile

Verfügbar für: Content Management Server, Template Engine, Search Engine Server

Aufgabe: Lädt die angegebene Datei und gibt ihren Inhalt als base64-kodierten String zurück.

Syntax:

`encodeFile` *`filename`*

Funktionsparameter:

- *`filename`* ist der Name der Datei.

Rückgabewert bei Erfolg: der base64-kodierte Inhalt der Datei (string).

Erforderliche Rechte (nur CM): keine Einschränkungen.

4.24.12 filterTags

Verfügbar für: Content Management Server

Aufgabe: Das Kommando entfernt aus einem Hauptinhalt unerwünschte Tags oder Tag-Attribute.

Syntax:

`filterTags blob` *`blob mode taglist`*

Funktionsparameter:

- *`blob`* ist der zu filternde Hauptinhalt.
- *`mode`* gibt an, wie mit dem Hauptinhalt verfahren werden soll. *`mode`* kann sein:
  - `allowedTags`: die Liste *`taglist`* enthält die Namen der Tags, die im Hauptinhalt vorkommen dürfen. Alle anderen Tags werden entfernt.
  - `disallowedTags`: die Liste *`taglist`* enthält die Namen der Tags, die aus dem Hauptinhalt entfernt werden sollen. Alle anderen Tags verbleiben im Hauptinhalt.
- *`taglist`* enthält die Liste der Tags, die (abhängig von *`mode`*) im Hauptinhalt verbleiben dürfen oder aus dem Hauptinhalt entfernt werden sollen. Ein Element von *`taglist`* kann wiederum eine Liste mit zwei Elementen sein. Das erste Element wird als der Name eines Tags interpretiert, das zweite als die Liste der Attribute dieses Tags. Ist *`mode`* gleich `allowedTags`, so sind bei diesem Tag die aufgeführten Attribute erlaubt und alle anderen werden entfernt. Ist *`mode`* gleich `disallowedTags`, so sind bei diesem Tag die aufgeführten Attribute nicht erlaubt und werden entfernt.

Rückgabewert bei Erfolg: der gefilterte Hauptinhalt.

Erforderliche Rechte (nur CM): keine.

Beispiel:

```
filterTags blob $oldBlob allowedTags {p br img}
```

4.24.13 getRegisteredCommands

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Gibt die beim Server registrierten Kommandos zurück. In Standardinstallationen wird diese Prozedur beim Start des Tcl-Clients aufgerufen, um die Namen der Prozeduren zu ermitteln, die der Tcl-Client zur Ausführung an den Tcl-Server weiterreichen muss.

Syntax:

```
getRegisteredCommands
```

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die Liste der registrierten Kommandos (stringlist).

Erforderliche Rechte (nur CM): keine Einschränkungen.

4.24.14 indexAllObjects

Aufgabe: Indiziert alle Versionen aller Dateien neu. Diese Prozedur kann nur verwendet werden, wenn der Search Engine Server eingesetzt wird.

Syntax:

```
indexAllObjects
```

Funktionsparameter: keine.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Bis Version 6.7.0 muss der Benutzer das Recht `permissionRoot` für alle Dateien haben, was normalerweise nur bei einem Superuser der Fall ist. Ab Version 6.7.1 muss der Benutzer ein Superuser sein.

Beispiel:

```
CM>indexAllObjects
done.
```

4.24.15 listTasks

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur gibt Informationen über Aufgaben (*tasks*) aus. Die Aufgaben können nach Kriterien gefiltert werden.

Syntax:

`listTasks {`*`parameter value`*`}`

Funktionsparameter:

- *`parameter`* und *`value`*: siehe `task where`.

Rückgabewert bei Erfolg: Liste von Aufgaben mit Angabe des Aufgabentyps, des Besitzers der Aufgabe, des Pfades der betreffenden Datei sowie des Zeitstempels. Die Abkürzungen für den Aufgabentyp bedeuten: `E` = Edit (Bearbeiten), `S` = Sign (Abzeichnen), `F` = FixLink (Bearbeiten). In der Owner-Spalte bedeuten `G` = Group und `U` = User.

werden nur die IDs der Dateien ausgegeben, für die der aktuelle Benutzer das Recht <code>permissionRead</code> hat.</p>

Beispiel:

```
listTasks taskType edit
 T  Owner    Path                              Time
E U root     /internet/index.html              2010090709
E U root     /internet/incoming/index.html     2010090710
2 tasks
```

4.24.16 listTasksOfGroup

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur gibt Informationen über Aufgaben (*tasks*) einer Benutzergruppe aus. Die Aufgaben können nach Kriterien gefiltert werden.

Syntax:

`listTasksOfGroup groupName {`*`parameter value`*`}`

Funktionsparameter:

- *`login`*: der Name der Gruppe, deren Aufgaben ausgegeben werden sollen.
- *`parameter`* und *`value`*: siehe task where.

Rückgabewert bei Erfolg: Liste von Aufgaben mit Angabe des Aufgabentyps, des Besitzers der Aufgabe, des Pfades der betreffenden Datei sowie des Zeitstempels. Die Abkürzungen für den Aufgabentyp bedeuten: `E` = Edit (Bearbeiten), `S` = Sign (Abzeichnen), `F` = FixLink (Bearbeiten). In der Owner-Spalte bedeuten `G` = Group und `U` = User.

Erforderliche Rechte: Es werden nur die IDs der Dateien ausgegeben, für die der aktuelle Benutzer das Recht `permissionRead` hat.

Beispiel:

```
CM>listTasksOfGroup redakteure taskType edit
 T  Owner         Path                              Time
E G redakteure    /internet/de/index.html           2010090709
1 task
```

4.24.17 listTasksOfUser

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur gibt Informationen über Aufgaben (*tasks*) eines Benutzers aus. Die Aufgaben können nach Kriterien gefiltert werden.

Syntax:

`listTasksOfUser login {`*`parameter value`*`}`

Funktionsparameter:

- *`login`*: der Anmeldename des Benutzers, dessen Aufgaben ausgegeben werden sollen.
- *`parameter`* und *`value`*: siehe task where.

Rückgabewert bei Erfolg: Liste von Aufgaben mit Angabe des Aufgabentyps, des Besitzers der Aufgabe, des Pfades der betreffenden Datei sowie des Zeitstempels. Die Abkürzungen für den Aufgabentyp bedeuten: `E` = Edit (Bearbeiten), `S` = Sign (Abzeichnen), `F` = FixLink (Bearbeiten). In der Owner-Spalte bedeuten `G` = Group und `U` = User.

Erforderliche Rechte: Es werden nur die IDs der Dateien ausgegeben, für die der aktuelle Benutzer das Recht `permissionRead` hat.

Beispiel:

```
CM>listTasksOfUser redakteur taskType edit
 T  Owner         Path                              Time
E U redakteur     /internet/de/index.html           2010090709
1 task
```

4.24.18 loadFile

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Gibt den Inhalt der angegebenen Datei zurück.

Syntax:

`loadFile` *`filename`*

Funktionsparameter:

- *`filename`* ist der Name der Datei.

Rückgabewert bei Erfolg: der Inhalt der Datei (binär).

Erforderliche Rechte (nur CM): keine Einschränkungen.

Beispiel:

```
CM>set a [loadFile /tmp/filename]
```

4.24.19 loadTextFile

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Gibt den Inhalt der angegebenen Textdatei zurück.

Syntax:

`loadTextFile` *`filename`* [*`charset`*]

Funktionsparameter:

- *`filename`* ist der Name der Datei.
- *`charset`* ist die Zeichenkodierung der zu lesenden Datei. Dieser Parameter ist optional. Die Voreinstellung ist `utf-8`. Geben Sie in der Tcl-Shell `encoding names` ein, um die Liste der verfügbaren Zeichenkodierungen ausgeben zu lassen.

Rückgabewert bei Erfolg: der Inhalt der Datei.

Erforderliche Rechte (nur CM): keine Einschränkungen.

Beispiel:

```
CM>set a [loadTextFile /tmp/textfile iso8859-1]
```

4.24.20 logMessage

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Der Befehl schreibt die angegebene Meldung in die applikationsspezifische Protokolldatei, sofern sie nicht ausführlicher ist als in der Systemkonfiguration eingestellt wurde.

Syntax:

`logMessage` *`level message`*

Funktionsparameter:

- *`level`* ist eine ganze Zahl zwischen 0 und 3 (jeweils einschließlich), die die Ausführlichkeit der Meldung anzeigt (3 = am ausführlichsten). In der Systemkonfiguration kann eingestellt werden (Eintrag `log`), wie ausführlich die Meldungen sein dürfen, die in die Protokolldatei geschrieben werden.
- *`message`* ist die zu protokollierende Meldung.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Keine Einschränkungen.

Beispiel:

```
TE>>logMessage 1 {Der Export wurde begonnen}
```

4.24.21 logWarning

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Der Befehl schreibt die angegebene Meldung als Warnung in die applikationsspezifische Protokolldatei.

Syntax:

`logWarning` *`message`*

Funktionsparameter:

- *`message`* ist die zu protokollierende Warnung.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): Keine Einschränkungen.

Beispiel:

```
CM>logWarning {Dies ist die Meldung}
```

4.24.22 logout

Verfügbar für: Content Management Server

Aufgabe: Beendet eine Verbindung mit dem Content Management Server.

Syntax:

`logout [`*`login`*`]`

Funktionsparameter:

- *`login`* ist der Anmeldename des Benutzers, dessen Verbindung beendet werden soll. Man muss ein Superuser oder der Verwalter des auszuloggenden Benutzers sein, um einen anderen Benutzer als sich selbst ausloggen zu können. Ist der Parameter nicht angegeben, so wird die aktuelle Verbindung getrennt.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Siehe oben, Funktionsparameter `login`.

Beispiel:

```
CM>logout mustermann
```

4.24.23 news where

Available for: Content Management Server, Template Engine

Aufgabe: Das Kommando liefert eine Liste von News-Datei-IDs. Die IDs werden absteigend nach Datum sortiert geliefert.

Syntax:

`news where [channels channellist] [length length]`

Funktionsparameter:

- *`channellist`* ist eine optional anzugebende Liste der zu berücksichtigenden Channels. Ist dieser Parameter nicht angegeben, werden die Channels nicht berücksichtigt.
- *`length`* gibt optional die maximale Länge der Liste an.

Rückgabewert bei Erfolg: eine Liste von Datei-IDs.

Erforderliche Rechte (nur CM): keine.

Beispiel:

```
CM>news where channels {politics sports} length 20
```

4.24.24 now

Verfügbar für: Content Management Server, Template Engine

Aufgabe: liefert die aktuelle Uhrzeit im 14-stelligen kanonischen Format.

Syntax:

`now`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: die aktuelle Uhrzeit (string).

Erforderliche Rechte (nur CM): keine Einschränkungen.

Beispiel:

```
CM>now
20110325163159
```

4.24.25 stream downloadBase64

Verfügbar für: Content Management Server, Search Engine Server

Management Server unter einer Ticket-ID abgelegten Daten base64-kodiert.</p>

Zusatzinformationen: Der Befehl kann nur im `-single`-Modus oder aus Assistenten heraus aufgerufen werden.

Syntax:

`stream downloadBase64 ticketId`

Funktionsparameter:

- *ticketId* ist eine Ticket-ID, die beispielsweise von den Kommandos `stream uploadFile` oder `stream uploadBase64` zurückgegeben wurde.

Rückgabewert bei Erfolg: die base64-kodierten Daten.

Erforderliche Rechte: keine Einschränkungen

Beispiel:

```
CM>stream downloadBase64 2098
PCFET0NUWVBFIEhUTUwgUFVCTElDICItL...
```

4.24.26 stream downloadFile

Verfügbar für: Content Management Server, Search Engine Server

Aufgabe: Das Kommando speichert die im Content Management Server unter einer Ticket-ID abgelegten Daten in einer Datei.

Zusatzinformationen: Das Kommando kann nur im `-single`-Modus oder aus Assistenten heraus aufgerufen werden.

Syntax:

`stream downloadFile` *`ticketId path`*

Funktionsparameter:

- *ticketId* ist eine Ticket-ID, die beispielsweise von den Kommandos `stream uploadFile` oder `stream uploadBase64` zurückgegeben wurde.
- *path* ist der Pfad, unter dem die Daten gespeichert werden sollen.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: keine Einschränkungen

Beispiel:

```
CM>stream downloadFile 2098 /tmp/testfile
```

4.24.27 stream uploadBase64

Verfügbar für: Content Management Server, Search Engine Server

Aufgabe: Das Kommando dekodiert eine base64-kodierte Zeichenkette und überträgt sie per Streaming zum Content Management Server, so dass andere Applikationen oder Skripte die Daten von dort abholen können.

Zusatzinformationen: Der Befehl kann nur im `-single`-Modus oder aus Assistenten heraus aufgerufen werden.

Syntax:

`stream uploadBase64 base64EncodedString`

Funktionsparameter:

- *base64EncodedString* ist die kodierte Zeichenkette.

Rückgabewert bei Erfolg: die Streaming-Ticket-ID, mit der die Datei abgeholt werden kann.

Erforderliche Rechte: keine Einschränkungen

Beispiel:

```
stream uploadBase64 [encodeFile /tmp/test]
2098
```

4.24.28 stream uploadFile

Verfügbar für: Content Management Server, Search Engine Server

Aufgabe: Das Kommando überträgt den Inhalt einer Datei per Streaming zum Content Management Server, so dass andere Applikationen oder Skripte sie von dort abholen können.

Zusatzinformationen: Das Kommando kann nur im `-single`-Modus oder aus Assistenten heraus aufgerufen werden.

Syntax:

`stream uploadFile path`

Funktionsparameter:

- *path* ist der Pfad der Datei, die zum Content Management Server übertragen werden soll.

Rückgabewert bei Erfolg: die Streaming-Ticket-ID, mit der die Datei abgeholt werden kann.

Erforderliche Rechte: keine Einschränkungen

```
CM>stream uploadFile /tmp/test
2097
```

4.24.29 sudo

Verfügbar für: Content Management Server

Aufgabe: Der Befehl gibt einem Superuser die Möglichkeit, ein Kommando unter einem anderen Login auszuführen.

Syntax:

`sudo login command {parameter}`

Funktionsparameter:

- *login* ist der Login, unter dem das Kommando ausgeführt werden soll.

- *command* ist das auszuführende Kommando. Dieses Kommando muss mit `safeInterp alias` registriert worden sein.
- *parameter* ist ein Parameter, der an das Kommando zu übergeben ist.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss Superuser sein, um den Befehl ausführen zu können.

Beispiel:

```
CM>sudo redakteur obj withPath /internet/playland forward
```

4.24.30 today

Verfügbar für: Content Management Server, Template Engine

Aufgabe: liefert das aktuelle Datum im 14-stelligen kanonischen Format.

Zusatzinformationen: Die Angabe für die Stunden entspricht clientseitig 0 Uhr.

Syntax:

`today`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: das aktuelle Datum (string).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>today
20110308230000
CM>systemConfig formatDateTime [today]
09.03.2011 00:00
```

4.24.31 valueForLocalizerKey

Verfügbar für: Content Management Server, Template Engine

Aufgabe: liefert aus dem Systemkonfigurationseintrag `localizers` eine Zeichenkette, lokalisiert in der Sprache des Benutzers.

Zusatzinformationen: Der Schlüssel `localizers` wurde in Infopark CMS Fiona 6 aus der Systemkonfiguration entfernt. In Assistenten können Zeichenketten mit Hilfe von Variablen lokalisiert werden. Siehe die mitgelieferten Assistenten.

Syntax:

`valueForLocalizerKey` *key* [*category*]

Funktionsparameter:

- *key* ist der Name des Schlüssels, dessen sprachspezifischer Wert zurückgegeben werden soll.
- *category* ist der Name einer Kategorie, die im `localizers`-Element enthalten ist.

Rückgabewert bei Erfolg: die lokalisierte Zeichenkette.

Erforderliche Rechte (nur CM): keine Einschränkungen.

Beispiel:

```
CM>valueForLocalizerKey meinKey meineKategorie
Der Wert von meinKey in meiner Sprache aus meineKategorie
```

4.24.32 whoami

Verfügbar für: Content Management Server

Aufgabe: Gibt das Login des aktuellen Benutzers zurück.

Syntax:

`whoami`

Funktionsparameter: keine.

Rückgabewert bei Erfolg: das Login des aktuellen Benutzers (string).

Erforderliche Rechte: keine Einschränkungen.

Beispiel:

```
CM>whoami
redakteur
```

4.24.33 writeFile

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Speichert Daten in einer Datei. Die Daten können binär sein, d. h. beispielsweise Nullzeichen enthalten.

Syntax:

`writeFile` *`filename data`*

Funktionsparameter:

- *filename* ist der Name der Datei.
- *data* sind die zu speichernden Daten.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): keine Einschränkungen.

Beispiel:

```
CM>writeFile /tmp/test [obj withId 2001 get exportBlob]
```

4.24.34 writeTextFile

Verfügbar für: Content Management Server, Template Engine

Aufgabe: Speichert Textdaten in einer Datei.

Syntax:

`writeTextFile` *`filename data`* [*`charset`*]

Funktionsparameter:

- *`filename`* ist der Name der Datei.
- *`data`* sind die zu speichernden Daten.
- *`charset`* ist die Zeichenkodierung des zu speichernden Textes. Dieser Parameter ist optional. Die Voreinstellung ist `utf-8` .

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte (nur CM): keine Einschränkungen.

Beispiel:

```
writeTextFile /tmp/text {Dies ist ein Beispiel}
```

4.25 Zusätzliche Tcl-Prozeduren

In der Datei `clientCmds.tcl`, die der Content Manager beim Start ausführt, ist eine Reihe nützlicher Tcl-Prozeduren definiert, deren Funktionen hier beschrieben sind.

4.25.1 contentService

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur ermöglicht es, das ContentService-Interface des Content Management Servers in Tcl-Skripten zu verwenden. Sie überträgt den Inhalt eines CMS-Ordners zu einer anderen Instanz.

Syntax:

`contentService` *`sourceUrl sourceLogin sourcePassword sourcePath destUrl destLogin destPassword destPath`*

Funktionsparameter:

- *`sourceUrl`*: Die URL der Quellinstanz.

- *sourceLogin, sourcePassword*: Die Anmeldedaten des Benutzers der Quellinstanz, über den die CMS-Dateien ausgelesen werden sollen.
- *sourcePath*: Der Pfad des auszulesenden Ordners in der Quellinstanz.
- *destUrl*: Die URL der Zielinstanz.
- *destLogin, destPassword*: Die Anmeldedaten des Benutzers, über den die CMS-Dateien in die Zielinstanz übernommen werden sollen.
- *destPath*: Der Pfad des Ordners in der Zielinstanz, in den die CMS-Dateien, einschließlich *sourcePath*, übernommen werden sollen.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der angegebene Benutzer der Quellinstanz muss das Leserecht für alle zu übertragenden Dateien haben. Der Benutzer der Zielinstanz muss das Schreibrecht für diese CMS-Dateien haben, falls sie bereits existieren. Um Dateien in der Zielinstanz anzulegen, benötigt der angegebene Benutzer im jeweils darüberliegenden Ordner das Dateianlegerecht.

Beispiel:

```
contentService
    http://localhost:8080/sourceInstance myUser myPassword /news
    http://otherServer:8080/destinationInstance otherUser otherPassword /intranet/content
```

4.25.2 cp

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur kopiert eine Datei in einen Ordner. Ist die zu kopierende Datei ein Ordner, so werden die darin enthaltenen Dateien nicht kopiert.

Syntax:

cp *sourceObjIdOrPath destObjIdOrPath*

Funktionsparameter:

- *sourceObjIdOrPath* ist die ID der Quelldatei oder sein Pfad. Der Pfad muss absolut sein, d. h. mit einem Schrägstrich beginnen.
- *destObjIdOrPath* ist die ID des Zielordners oder ihr Pfad. Der Pfad muss absolut sein, d. h. mit einem Schrägstrich beginnen.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Leserecht für die Quelldatei und in dem Zielordner das Recht haben, Dateien anzulegen.

Beispiel:

```
CM>cp /internet/presse /resources
```

4.25.3 findObjectId

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur ermittelt die ID einer Datei, sofern sie existiert.

Syntax:

`findObjectId` *`idOrPath`*

Funktionsparameter:

- *`idOrPath`* kann die ID oder der Pfad einer Datei sein.

Rückgabewert bei Erfolg: die ID der Datei (string).

Erforderliche Rechte: keine.

Beispiel:

```
CM>findObjectId /myFolder
2145
```

4.25.4 grep

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur ermittelt aus einer Liste von CMS-Dateien diejenigen, die eine angegebene Zeichenkette in einem angegebenen Feld enthalten. Zusätzlich kann der Status der zu findenden Dateien eingeschränkt werden.

Syntax:

`grep` *`expression objects`* [*`attribute`*] [*`args`*]

Funktionsparameter:

- *`expression`*: der anzuwendende reguläre Ausdruck
- *`objects`*: die Liste der CMS-Dateien, die durchsucht werden sollen
- *`attribute`*: das zu durchsuchende Feld. Diese Angabe ist optional, Voreinstellung: `blob`
- *`args`*: optionale Angabe zusätzlicher Einschränkungen. *`args`* kann sein:
  - `-states` *`state`*: Dies schränkt die Treffermenge auf CMS-Dateien ein, die einen bestimmten Bearbeitungsstatus oder einen aus einer Menge von Status haben. *`state`* kann sein:
    - Eine Liste mit einem oder mehreren der folgenden folgenden Werte: `edited`, `committed`, `released`, `archived`.
    - `all` (Voreinstellung) steht für einen beliebigen Status.
    - `active` steht für einen beliebigen Status außer `archived`.

Rückgabewert bei Erfolg: je gefundene Datei deren Pfad, Angaben zur Fundstelle sowie ein Auszug aus dem Text, in dem der Suchausdruck vorkommt (string). Die Fundstelle wird mit den Kürzeln `o` (Datei-ID), `c` (Versions-ID) und `l` (Zeilennummer) näher spezifiziert (siehe das Beispiel).

Erforderliche Rechte: Der Benutzer muss das Leserecht für die gefundenen Dateien haben.

Beispiel:

```
CM>grep "modifyvar" [obj where condition {objType is template}]
/mastertemplate (o:2011,c:2015,l:4) <npsobj modifyvar="set"
varname="contentTypeAndCharset">text/html;
charset=<npsobj insertvalue="var" name="exportCharset" /></npsobj>
```

4.25.5 incrNpsDate

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur addiert eine Zeitspanne zu einem Datum im kanonischen Format.

Syntax:

`incrNpsDate` *`date offset`*

Funktionsparameter:

- *`date`* ist die Datum-Zeichenkette als 14-stellige Ziffernfolge.
- *`offset`* ist eine Zeichenkette, die eine relative Zeitangabe enthält. Diese besteht aus einer ganzen Zahl, auf die eines der folgenden Wörter folgt: `year`, `fortnight`, `month`, `week`, `day`, `hour`, `minute` (oder `min`), `second` (oder `sec`). Das Wort kann im Singular oder im Plural angegeben werden.

Rückgabewert bei Erfolg: das berechnete absolute Datum in kanonischer Form (string).

Erforderliche Rechte: keine.

Beispiel:

```
CM>incrNpsDate 20110402091300 "-3 months"
20110102091300
```

4.25.6 interactiveLoadSubtree

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur hat die gleiche Funktion wie `loadSubtree` . Lässt sich die Dateiendung nicht eindeutig einer Dateivorlage zuordnen, erfragt die Prozedur die fehlende Information. Ferner kann der Benutzer angeben, ob seine Angaben für alle Dateien mit der betreffenden Dateiendung gelten sollen.

Syntax:

`interactiveLoadSubtree` *`parent startDir startName pubObjClass`* [*`mapping`*]

4.25.7 l

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur gibt eine Tabelle mit ausführlichen Informationen über die Dateien in einem Ordner zurück. Aufgeführt werden die Datei-ID, -vorlage, der Dateityp sowie der -name und -titel.

Syntax:

`l` *`objIdOrPath`*

Funktionsparameter:

- *`objIdOrPath`* ist die ID des Ordners oder ihr Pfad. Der Pfad muss absolut sein, d. h. mit einem Schrägstrich beginnen.

Rückgabewert bei Erfolg: String.

Erforderliche Rechte: Es werden nur Dateien aufgeführt, auf die der Benutzer das Leserecht hat.

Beispiel:

```
CM>l /
/ (2001)  total (2)
objId   class               type        name                title
-------------------------------------------------------------------
2006    template            template    mastertemplate      mastertemplate
2050    document            document    test
```

4.25.8 listObjectsWithoutSuperLinks

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur gibt bei einem Ordner die IDs sämtlicher darin enthaltener Dateien zurück, auf die nicht verwiesen wird.

Syntax:

`listObjectsWithoutSuperLinks` *`objIdOrPath`*

Funktionsparameter:

- *`objIdOrPath`* ist die ID des Ordners oder ihr Pfad. Der Pfad muss absolut sein, d. h. mit einem Schrägstrich beginnen.

Rückgabewert bei Erfolg: Liste von Datei-IDs.

Erforderliche Rechte: Der Benutzer muss das Leserecht für den Ordner haben. Es werden nur die IDs der Dateien zurückgegeben, für die der Benutzer das Leserecht hat.

Beispiel:

```
CM>listObjectsWithoutSuperLinks /
objects without superlinks in path /
/test {Test}
/mastertemplate {mastertemplate}
```

4.25.9 listSubtree

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur gibt die IDs sämtlicher Dateien zurück, die sich in der Hierarchie unterhalb eines Ordners befinden, einschließlich des Ursprungsordners.

Syntax:

`listSubtree` *`objIdOrPath`*

Funktionsparameter:

- *`objIdOrPath`* ist die ID des Ordners oder ihr Pfad. Der Pfad muss absolut sein, d. h. mit einem Schrägstrich beginnen.

Rückgabewert bei Erfolg: Liste von Datei-IDs.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionRead` für den Ordner haben. Es werden nur die IDs der Dateien ausgegeben, für die der Benutzer das Recht `permissionRead` hat.

Beispiel:

```
CM>listSubtree /
2001 2050 2006
```

4.25.10 loadSubtree

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur importiert die Dateien in einem Verzeichnis in den Content Manager, wobei auch Unterverzeichnisse berücksichtigt werden.

Zusatzinformationen: Für das Ausgangsverzeichnis wird ein Ordner mit dem Namen *`startName`* in dem Ordner *`parent`* angelegt. Für jedes weitere Verzeichnis wird ein Ordner mit dem Namen des Verzeichnisses angelegt. Befindet sich in einem Verzeichnis eine Datei mit dem Namen `index.html`, so wird sie als Hauptinhalt des korrespondierenden erzeugten Ordners importiert. Die Prozedur gibt abschließend eine Liste der Dateien aus, die nicht importiert werden konnten (weil beispielsweise die Dateivorlage nicht eindeutig ermittelt werden konnte).

Syntax:

`loadSubtree` *`parent startDir startName pubObjClass`* [*`mapping`*]

Funktionsparameter:

- *`parent`* ist die ID des Ordners, in dem die Teilhierarchie erzeugt wird.
- *`startDir`* ist der Pfad zum Verzeichnis, das importiert werden soll.
- *`startName`* ist der Name des Ausgangsordners.
- *`pubObjClass`* ist der Name der Dateivorlage, das der erzeugte Ordner haben soll.
- *`mapping`* ist eine Liste mit Zuordnungen. Jede Festlegung ist wiederum eine Liste, die die folgenden Parameter (in der hier aufgeführten Reihenfolge) enthält:

- *contentType*: die Dateiendung, für die eine Dateivorlage festgelegt wird.
- *objClass*: die Dateivorlage, die die Dateien haben sollen, die für Dateien mit der Namensendung *contentType* erzeugt werden.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionCreateChildren` für den Ausgangsordner haben. Unter Umständen sind weitere Rechte erforderlich, beispielsweise das Recht, eine Dateivorlage zu verwenden.

Beispiel:

```
CM>loadSubtree /internet /tmp/import press presse_ordner mapping {{html presse_doc} {pdf
 pdf_doc}}
```

4.25.11 ls

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur gibt eine Tabelle mit Informationen über die Dateien in einem Ordner zurück.

Syntax:

`ls` *objIdOrPath*

Funktionsparameter:

- *objIdOrPath* ist die ID des Ordners oder ihr Pfad. Der Pfad muss absolut sein, d. h. mit einem Schrägstrich beginnen.

Rückgabewert bei Erfolg: Tabellenstring.

Erforderliche Rechte: Es werden nur Dateien aufgeführt, für die der Benutzer das Leserecht hat.

Beispiel:

```
CM>ls /
/ (2001)  total (2)
objId   class               type        name
-----------------------------------------
2006    template            template    mastertemplate
2050    document            document    test
```

4.25.12 mkpubs

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur erzeugt für jedes Element im angegebenen Pfad einen Ordner mit der Vorlage *objClass*.

Syntax:

`mkpubs` *path objClass*

Funktionsparameter:

- *path* ist der Pfad, der angelegt werden soll. Der Pfad muss absolut sein.
- *objClass* ist die Dateivorlage, die die angelegten Ordner haben sollen.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss für die Ordner, in denen Dateien angelegt werden sollen, das Recht `permissionCreateChildren` haben.

Beispiel:

```
CM>mkpubs /internet/playland/de publication
```

4.25.13 modifyPermissions

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur modifiziert die Zugriffsrechte der angegebenen Gruppen für die angegebenen Dateien.

Zusatzinformationen: Je nach Modifikator *modifier* werden den angegebenen Gruppen Rechte erteilt oder entzogen.

Syntax:

`modifyPermissions` *objIdsOrPaths modifier groups permissions*

Funktionsparameter:

- *objIdsOrPaths* ist die Liste der Dateien, auf die sich die zu ändernden Zugriffsrechte beziehen. Die Liste kann sowohl Datei-IDs als auch Pfade enthalten. Pfade müssen absolut sein, d.h. sie müssen mit einem Schrägstrich beginnen.
- *modifier* ist einer der folgenden Modifikatoren:
  - `grantTo`: den Gruppen werden die Rechte *permissions* erteilt.
  - `set`: den Gruppen werden zunächst sämtliche dateispezifischen Rechte entzogen. Anschließend werden den Gruppen die Rechte *permissions* erteilt.
  - `revokeFrom`: den Gruppen werden die Rechte *permissions* entzogen.
- *groups* ist die Liste der Gruppen, bei denen die Zugriffsrechte geändert werden sollen.
- *permissions* ist die Liste der dateispezifischen Rechte (wie `permissionRead`).

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionRoot` für die angegebenen Dateien haben.

Beispiel:

```
CM>modifyPermissions [obj list] set {admins rootusers} permissionRoot
modifying permissions
done.
```

4.25.14 objWherePath

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur listet die IDs aller Dateien auf, die sich in dem angegebenen Pfad befinden, also Dateien, deren Pfad mit dem Pfad der angegebenen Datei beginnt. Das Ergebnis kann auf bestimmte Dateitypen und -vorlagen eingeschränkt werden.

Syntax:

`objWherePath` *`idOrPath`* {*`parameter value`*}

Funktionsparameter:

- *`parameter`* ist einer der folgenden optionalen Werte:
  - `objTypes`: gibt an, dass *`value`* eine Liste der Dateitypen enthält, auf die die Ergebnisliste beschränkt werden soll.
  - `objClasses`: gibt an, dass *value* eine Liste der Dateivorlagen enthält, auf die die Ergebnisliste eingeschränkt werden soll.
- *`value`* ist der zum jeweiligen Parameter gehörende Wert.

Zusatzinformationen: Es ist möglich, jedoch nicht sinnvoll, beide Parameter anzugeben, weil sie miteinander UND-verknüpft werden und die Vorlage bereits den Dateityp festlegt. Wird `objClasses` angegeben, werden implizite Spiegeldateien nicht berücksichtigt.

Rückgabewert bei Erfolg: Die Liste der Datei-IDs, die in dem angegebenen Pfad liegen und zu den angegebenen Parameterwerten passen.

Erforderliche Rechte: Es werden nur Dateien aufgeführt, auf die der Benutzer lesend zugreifen darf.

Beispiel:

```
CM>objWherePath /internet objClasses {publication}
2108 2116 2124
```

4.25.15 performWithEditedContentOfObjects

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur führt für jede Datei in der angegebenen Liste *`objIds`* das Kommando `obj withId` *`id`* `editedContent` *`command`* aus. Hat eine Datei keine Arbeitsversion, so wird sie stillschweigend übergangen.

Syntax:

`performWithEditedContentOfObjects` *`command objIds`*

Funktionsparameter:

- *`command`* ist eines der verfügbaren `content`-Kommandos einschließlich der Parameter, die diesem Kommando übergeben werden sollen.

- *objIds* ist die Liste der Dateien, für die das Kommando ausgeführt werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Es werden nur die Dateien berücksichtigt, für die der Benutzer das Recht `permissionRead` hat. Um das angegebene Kommando ausführen zu können, können weitere Rechte erforderlich sein.

Beispiel:

```
CM>performWithEditedContentOfObjects {set validFrom [now]} [obj list]
performing set validFrom [now] with edited Content
done.
```

4.25.16 performWithObjects

Aufgabe: Die Prozedur führt für jede Datei in der angegebenen Liste *objIds* das Kommando `obj withId` *id command* aus.

Syntax:

`performWithObjects` *command objIds*

Funktionsparameter:

- *command* ist eines der verfügbaren `obj`-Kommandos einschließlich der Parameter, die diesem Kommando übergeben werden sollen.
- *objIds* ist die Liste der Dateien, für die das Kommando *command* ausgeführt werden soll.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Es werden nur die Dateien berücksichtigt, für die der Benutzer das Recht `permissionRead` hat. Um das angegebene Kommando ausführen zu können, können weitere Rechte erforderlich sein.

Beispiel:

```
CM>performWithObjects {set suppressExport 0} {2108 2116 2124}
performing set suppressExport 0
done.
```

4.25.17 removeBlobAndFreeLinks

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur löscht die Arbeitsversion und die freigegebene Version einer Datei sowie alle in diesen Versionen enthaltenen Links.

Syntax:

`removeBlobAndFreeLinks` *objId*

Funktionsparameter:

- *objId* ist die ID der Datei.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss für die Datei das Recht permissionRoot haben.

Beispiel:

```
CM>removeBlobAndFreeLinks [obj withPath /internet get id]
```

4.25.18 removeSubtree

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur löscht den angegebenen Ordner und alle darin enthaltenen Dateien.

Syntax:

removeSubtree [-verbose {0|1}] [-check {0|1}] [-removeExternalMirrors {0|1}] [-force {0|1}] *objectIdOrPath*

Funktionsparameter:

- *-verbose* legt fest, ob Fortschrittsinformationen ausgegeben werden sollen (1, Voreinstellung) oder nicht (0). Wenn der Befehl im Tcl-Client aufgerufen wird und *-verbose* den Wert 0 hat, wird der Befehl im Server ausgeführt.
- *-check* legt fest, ob Zugriffsrechte und Links geprüft werden sollen (1, Voreinstellung) oder nicht (0). Die Voreinstellung bewirkt, dass vor der Ausführung des Befehls alle Dateien daraufhin geprüft werden, ob sie gelöscht werden können, d. h. keine Linkziele sind und die Benutzerrechte ausreichend sind. Nur wenn diese Bedingungen erfüllt sind, wird die Ausführung des Kommandos fortgesetzt. Der Wert 0 dagegen führt diese Prüfung nicht durch, wodurch die Teilhierarchie gegebenenfalls unvollständig gelöscht wird.
- *-removeExternalMirrors* (ab Version 6.5.0 PP1) bewirkt, dass Spiegeldateien außerhalb des zu löschenden Teilbaums, deren jeweiliges Original jedoch im Teilbaum liegt, ebenfalls gelöscht werden.
- *-force* erzwingt, dass das Kommando weiter ausgeführt wird (1), auch wenn beim Löschen von Dateien Fehler auftreten. Fehler können dann auftreten, wenn zu löschende Dateien Linkziele sind oder aufgrund fehlender Zugriffsrechte nicht gelöscht werden können. Die Voreinstellung (0) bewirkt, dass das Kommando bei Fehlern abgebrochen wird.
- *objIdOrPath* ist die ID des Ordners oder ihr Pfad. Der Pfad muss absolut sein, d. h. mit einem Schrägstrich beginnen.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht permissionRoot für den Ordner und alle darin enthaltenen Dateien haben, andernfalls können die betreffenden Dateien nicht gelöscht werden.

Beispiel:

```
CM>removeSubtree -verbose 1 -check 0 -force 1 /internet
```

4.25.19 renameObjClass

Verfügbar für: Content Management Server (ab Version 6.6.1)

Aufgabe: Die Prozedur benennt eine Vorlage um und indiziert die Dateien neu, in denen die Vorlage eingetragen ist. Ferner listet das Kommando die Dateien auf, in denen der Name der Vorlage vorkommt. Das Kommando ändert diese Namen nicht.

Syntax:

`renameObjClass` *`oldName newName`*

Funktionsparameter:

- *`oldName`* ist der bisherige Name der Vorlage.
- *`newName`* ist der neue Name der Vorlage.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionGlobalRTCEdit` haben.

Beispiel:

```
root@localhost:3001 default(CM)> renameObjClass publication Folder

Updating index
-------------------------------------------------------------------------------
performing updateInIndex
done.

Searching for config files that contain references to the original object class name
 'publication'.
-------------------------------------------------------------------------------
/Fiona/instance/default/config/content.xml:28  <defaultPublication>publication</
defaultPublication>
/Fiona/instance/default/config/webDav.xml:4    via WebDAV. Default is "publication". -->
/Fiona/instance/default/config/webDav.xml:5    <defaultPublicationClass>publication</
defaultPublicationClass>
...

Searching for templates that contain references to the original object class name
 'publication'.
-------------------------------------------------------------------------------
/mastertemplate (o:2010,c:2084,l:2) <npsobj caseCond="isEqual" value="publication">


Commands to search again for occurrences
-------------------------------------------------------------------------------
Config files:
  grepInConfig {\ypublication\y}

Templates:
  grepInTemplates {\ypublication\y}
```

4.25.20 rm

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur löscht eine Datei.

Zusatzinformationen: Der Befehl basiert auf dem `obj delete`-Befehl, und es gelten die dort genannten Einschränkungen.

Syntax:

`rm` *`objIdOrPath`*

Funktionsparameter:

- *`objIdOrPath`* ist die ID der Datei oder ihr Pfad. Der Pfad muss absolut sein, d. h. mit einem Schrägstrich beginnen.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss für die zu löschende Datei das Recht `permissionRoot` haben.

Beispiel:

```
CM>rm /cats
```

4.25.21 showLinksOfObjects

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur gibt für jede Datei eine Liste der darin enthaltenen Links aus. Es werden nur Dateien berücksichtigt, die Links enthalten.

Syntax:

`showLinksOfObjects` *`objIdsOrPaths`* [*`selector`*]

Funktionsparameter:

- *`objIdsOrPaths`* ist die Liste der Dateien, die untersucht werden sollen. Die Liste kann sowohl Datei-IDs als auch Pfade enthalten. Pfade müssen absolut sein, d. h. sie müssen mit einem Schrägstrich beginnen.
- *`selector`* ist einer der folgenden Selektoren:
  - `all`: alle Links werden berücksichtigt.
  - `dead`: nur die nicht aufgelösten Links werden berücksichtigt.

Rückgabewert bei Erfolg: die Liste der Dateien, die Links enthalten. Für jeden Link werden dessen ID und Ziel ausgegeben.

Erforderliche Rechte: Es werden nur die Dateien berücksichtigt, für die der Benutzer das Recht `permissionRead` hat.

Beispiel:

```
CM>showLinksOfObjects [obj list] all
showing links
/ (2001)
link 2193 /test.html
done.
```

4.25.22 showPermissions

Verfügbar für: Content Management Server

Aufgabe: Die Prozedur listet für jede Datei auf, welche Gruppen welche Zugriffsrechte haben.

Syntax:

`showPermissions` *`objIdsOrPaths`*

Funktionsparameter:

- *`objIdsOrPaths`* ist die Liste der Dateien, die untersucht werden sollen. Die Liste kann sowohl Datei-IDs als auch Pfade enthalten. Pfade müssen absolut sein, d. h. sie müssen mit einem Schrägstrich beginnen.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Der Benutzer muss das Recht `permissionRead` für die angegebenen Dateien haben.

Beispiel:

```
CM>showPermissions {/ 2050}
/ (2001)
        permissionRoot: rootusers admins
/test (2050)
        permissionRead: admins
        permissionWrite: admins
        permissionRoot: rootusers admins
```

4.25.23 streamFromServer

Verfügbar für: Content Management Server

Aufgabe: Liest die Daten, die unter einer Streaming-Ticket-ID gespeichert wurden, vom angegebenen Server.

Syntax:

`streamFromServer` *`host port ticketId fileName`*

Zusatzinformationen:

- Die Routine verwendet das Streaming-Interface, mit dem auch große Datenmengen speicherschonend übertragen werden können.
- Die maximale Größe eines übertragenen Datenblocks überschreitet 64 KB nicht. Dies bedeutet nicht, dass die Datenmenge, die mit dieser Routine übertragen werden kann, auf 64 KB beschränkt ist.

Funktionsparameter:

- *`host`* ist der Name oder die IP-Adresse des Servers.
- *`port`* ist der Port über den auf die Daten zugegriffen werden kann.

- *ticketId* ist die ID, die zuvor von dem Server zurückgegeben wurde, nachdem die Daten zu ihm übertragen wurden.
- *fileName* ist der Name oder Pfad der Datei, in der die Daten gespeichert werden.

Rückgabewert bei Erfolg: keiner.

Erforderliche Rechte: Keine Einschränkungen.

Beispiel:

```
CM>streamFromServer localhost 3002 [obj withId 53891 \
editedContent get blob.stream] ~/meinDokument.pdf
```

4.25.24 streamToServer

Verfügbar für: Content Management Server

Aufgabe: Sendet den Inhalt einer Datei zu einem Server und gibt als Referenz eine ID zurück.

Zusatzinformationen:

- Die Routine verwendet das Streaming-Interface, mit dem auch große Datenmengen speicherschonend übertragen werden können.
- Die maximale Größe eines übertragenen Datenblocks überschreitet 64 KB nicht. Dies bedeutet nicht, dass die Datenmenge, die mit dieser Routine übertragen werden kann, auf 64 KB beschränkt ist.
- Die Gültigkeit der Ticket-ID ist auf die Zeitspanne beschränkt, die im Systemkonfigurationseintrag *streamingTicketValidityTimeInterval* angegeben wurde.

Syntax:

`streamToServer` *host port fileName*

Funktionsparameter:

- *host* ist der Name oder die IP-Adresse des Servers.
- *port* ist der Port, zu dem die Daten gesendet werden.
- *fileName* ist der Name oder Pfad der Datei, deren Inhalt zum Server gesendet werden soll.

Rückgabewert bei Erfolg: Die Ticket-ID, unter der auf die Daten zugegriffen werden kann (string).

Erforderliche Rechte: Keine Einschränkungen.

Beispiel:

```
CM>obj withId 53891 editedContent set blob.stream \
[streamToServer localhost 3002 ~/meinDokument.pdf]
```

5

5 Mit Tcl Standardaufgaben lösen

5.1 Vorlagen kopieren

Die folgende Prozedur, `copyObjClass`, erstellt eine neue Vorlage aus einer bestehenden, indem die Werte der `setKeys` der bestehenden Vorlage in die neue Vorlage geschrieben werden. Die Prozedur hat zwei Argumente, die Namen der bestehenden und der zu erzeugenden Vorlagen (in dieser Reihenfolge).

```
# $Id: copyObjClass.tcl, v. 0.9 2010-11-23 13:51:46
#
# (c) 2010 Infopark AG
# Last changed: 2010-11-23
# Use at your own risk!
#
# Purpose:
#   Creates a new file fromat from an existing one
# Parameters:
#   Name of source file format,
#   Name of file format to create

proc copyObjClass {sourceClassName destClassName} {
  set sourceClassType [objClass withName $sourceClassName get objType]

  set setKeys [objClass withName $sourceClassName get setKeys]
  set nameIndex [lsearch $setKeys {name}]
  set setKeys [lreplace $setKeys $nameIndex $nameIndex]
  set values [eval [list objClass withName $sourceClassName mget] $setKeys]

  set setValues [list]
  foreach key $setKeys value $values {
    lappend setValues $key $value
  }

  eval [list objClass create name $destClassName objType $sourceClassType] $setValues
}
```

Speichern Sie bitte das Skript in einer Tcl-Datei ab. Öffnen Sie eine Tcl-Shell, stellen Sie eine Verbindung zum Content Manager her und lesen Sie die Tcl-Datei mit dem Befehl `source` ein. Nun können Sie die Prozedur verwenden. Beispiel:

```
copyObjClass articleDocument newsDocument
```

5.2 Teilhierarchien kopieren

Die folgenden beiden Prozeduren, `copySubtree` und `copySubtreeAndLinks`, kopieren eine Ordnerhierarchie an eine andere Stelle. Im Unterschied zu `copySubtree`, passt `copySubtreeAndLinks` nach dem Kopiervorgang die Links, die auf Dateien in der ursprünglichen Hierarchie zeigen, so an, dass sie auf die neuen, kopierten Dateien zeigen.

```
# $Id: copySubtree.tcl,v 1.2 2002/03/27 15:51:06
#
# (c) 2001-2009 Infopark AG
# Last changed: 2009-09-25
# Purpose:
#   copySubtreeAndLinks: Copy a file tree and re-generate the links
#   copySubtree: Simple tree-copying without linkage
# Parameters:
#   File ID or name of the folder to copy,
#   File ID or name of the destination folder
#   [keepStatus]
# keepStatus (optional) tries to assign the same status to the copied files
# as they had before and prints a warning if a file has a released and a
# draft/committed version. In this case only the released version will be copied.

proc copySubtreeAndLinks {{objId ""} {destination ""} {option ""} } {
  if {$destination==""} {
    global copySubtree_usage
    puts $copySubtree_usage
    return
  }
  if {($option!="") && ($option!="keepStatus")} {
    puts "$option is a not allowed option!"
    return
  }
  set to_release 0
  set objId [findObjectId $objId]
  set destination [findObjectId $destination]
  set oldPath [obj withId $objId get path]
  set newPath [obj withId $destination get path]
  set newPath $newPath/[obj withId $objId get name]
  if {[catch {findObjectId $newPath}] == 0} {
    set newPath [obj withId [obj withId $destination create objClass \
      [obj withId $objId get objClass] name [obj withId $objId get name]] get path]
    obj withPath $newPath delete
  }
  puts "Copying files ..."
  copySubtree $objId $destination $option
  puts "\nSubtree successfully copied, now links will be adapted ..."
  set newId [obj withPath $newPath get id]
  foreach i [listSubtree $newId] {
     if !{[obj withId $i get isMirror]} {
        if {[obj withId $i get isReleased] == 1} {
          obj withId $i unrelease
          set to_release 1
        }
        foreach j [obj withId $i get subLinks] {
          set dest [link withId $j get destinationUrl]
          if {[regexp "^$oldPath" $dest]} {
            regsub $oldPath $dest $newPath dest
            link withId $j set destinationUrl $dest
          }
        }
        if {$to_release == 1} {
          obj withId $i release
          set to_release 0
        }
     }
  }
  puts "done"
}
```

```
proc copySubtree {{objId ""} {destination ""} {option ""} } {
  if {$destination==""} {
    global copySubtree_usage
    puts $copySubtree_usage
    return
  }
  set objId [findObjectId $objId]
  set destination [findObjectId $destination]
  if {$objId == $destination} {
    puts "You can't copy a file to itself"
    return
  }
  set newId [obj withId $objId copy parent $destination]
  puts -nonewline "\r$objId"
  flush stdout
  if {$option=="keepStatus"} {
    if {[obj withId $objId get isCommitted]=="1"} {
      obj withId $newId commit
    }
    if {[obj withId $objId get isReleased]=="1"} {
      obj withId $newId release
      if {([obj withId $objId get isCommitted]=="1") ||
          ([obj withId $objId get isEdited]=="1")} {
        puts "\nFile $objId has a released and a draft version.\
          Only the released version was copied"
      }
    }
  }
  if {[obj withId $objId get objType] == "publication" &&
      ![obj withId $objId get isMirror]} {
    foreach i [obj withId $objId get children] {
      copySubtree $i $newId $option
    }
  }
}
```

Speichern Sie bitte beide Skripte in einer einzigen Tcl-Datei ab. Öffnen Sie eine Tcl-Shell, stellen Sie eine Verbindung zum Content Manager her und lesen Sie die Tcl-Datei mit dem Befehl `source` ein. Nun können Sie die beiden Prozeduren verwenden. Das folgende Beispiel kopiert den Ordner `/de/news` komplett nach `/internet/de/news`:

```
copySubtreeAndLinks /de/news /internet/de
```

5.3 Alte Versionen und Protokolleinträge löschen

Alte archivierte Versionen von Dateien verbrauchen oft viel Speicherplatz und können die Performance des CMS verringern. Mit Hilfe des folgenden Skripts können diese Versionen gelöscht werden. Dieses Skript löscht ferner die zu den gelöschten Versionen gehörenden Protokoll-Einträge.

Legen Sie das Skript bitte in einer Datei mit dem Namen `cutHistory.tcl` ab. Anschließend können Sie sie mit dem Tcl-Client Ihres CMS einlesen und die Prozedur `cutHistory` ausführen:

```
CM>source /pfad/zu/cutHistory.tcl
CM>cutHistory Datei_ID Verbleibende_Einträge
```

Hier nun das Skript:

```
# (c) 2001 Infopark AG
# Modified: 2010-04-13
```

```
# cutHistory.tcl for Fiona 6.x
#
# Purpose:
# Reduce (cut) the number of archived (i.e. previously released) versions
# to the given number. Furthermore, remove the associated log enries.
#
# Parameters:
# - File ID or path
# - Number of released versions to remain in the system

proc cutHistory {id cutLength} {
    set objectId [findObjectId $id]
    # CMS Fiona 6.5 or later: exclude mirrors
    if {[obj withId $objectId get isMirror] eq "1"} {
        puts "\nMirror files ($objectId) don't have versions..."
    } else {
        set contents [lsort -integer -decreasing \
            [obj withId $objectId get archivedContentIds]]
        set versionLength [llength $contents]
        set logLength [llength [logEntry where logTypes action_release_obj \
            objectId $objectId]]
        puts "Cut by Length:\t$cutLength"
        puts "Log-Size:\t$logLength"
        puts "Versions:\t$versionLength"
        if {[catch {set cutByDate [content withId [lindex $contents $cutLength] \
            get lastChanged]}]} {
            puts "\nFile $objectId has no archived versions..."
        } else {
            foreach i [lrange $contents $cutLength $versionLength] {
                content withId $i delete
                puts "content $i deleted"
            }
            if {[catch {logEntry deleteWhere objectId $objectId \
                    untilDate $cutByDate} msg]} {
                puts "\nCouldn't delete log entries of file $objectId: \n$msg"
            }
        }
    }
}
```

5.4 Dateien finden

Der Content Navigator, d.h. die Weboberfläche von Infopark CMS Fiona bietet mit ihrer Suchseite einen komfortablen Weg, eine bestimmte Datei zu finden. Wenn es nur darum geht, Dateien mit bestimmten Namen via Tcl ausfindig zu machen, hilft ein kleines Skript.

Das Skript braucht lediglich die Namen aller Dateien in einer Teilhierarchie mit einem Suchbegriff zu vergleichen und die Pfade der Treffer auszugeben:

```
proc locate {objId searchString} {
  puts "Getting Ids..."
  set counter 0
  set objects [listSubtree [findObjectId $objId]]
  foreach obj $objects {
    set name [obj withId $obj get name]
    if {[string match *$searchString* $name] == 1} {
      puts "[obj withId $obj get path]"
      incr counter
    }
  }
  puts "\n$counter files found"
}
```

Speichern Sie diese Prozedur in einer Skript-Datei ab, beispielsweise in `locate.tcl`. Starten Sie anschließend den Tcl-Client des Content Managers und machen Sie die Prozedur `locate` in der Skript-Datei mit dem Befehl `source locate.tcl` verfügbar. Sie können die Prozedur nun aufrufen:

```
CM>locate /internet/news business
```

5.5 Dateien verschieben

Einzelne Dateien oder auch Teilbäume können mit Tcl auf einfache Weise verschoben werden. Hierfür braucht lediglich das Feld `parent` der betreffenden Datei mit der ID des Ordners belegt zu werden, in den die Datei verschoben werden soll:

```
CM>obj withId 4592 set parent 2001
```

Dieser Befehl verschiebt die Datei mit der ID 4592 in den Ordner mit der ID 2001 (dies ist der Basisordner). Selbstverständlich können Sie statt der IDs auch Pfade angeben:

```
CM>obj withPath /internet/news set parent [obj root get id]
```

Wie beim Anlegen von Dateien müssen auch beim Verschieben die Restriktionen beachtet werden, die in dem Zielordner gelten:

- Die Vorlage der zu verschiebenden Datei muss in dem Zielordner als Vorlage von darin enthaltenen Dateien erlaubt sein.
- Der Benutzer, der die Aktion durchführt, muss für die zu verschiebende Datei das Administrationsrecht haben.
- In dem Zielordner muss der Benutzer das Recht haben, Dateien anzulegen.

5.6 Tcl-Skripte in mehreren Callback-Funktionen verwenden

Manchmal müssen in mehreren Callback-Funktionen die gleichen Tcl-Prozeduren verwendet werden. Um den gleichen Code nicht an mehreren Stellen pflegen zu müssen, können Sie ihn in eine Datei auslagern, die beim Start des Content Managers eingelesen wird. Gehen Sie hierzu folgendermaßen vor:

1. Legen Sie eine Datei mit der Prozedur in ein Verzeichnis, dessen Inhalt beim Start des Content Managers automatisch eingelesen wird, beispielsweise

   `instance/`*`myInstance`*`/script/cm/serverCmds/callbackCommand.tcl`
2. Die Prozedur sollte alle Argumente erhalten, die auch der Funktion oder den Funktionen übergeben werden. Ferner muss sie die Werte zurückgeben, die für die jeweiligen aufrufenden Funktionen relevant sind. Der folgende Code zeigt dies am Beispiel eines Vollständigkeitschecks:

   ```
   proc callbackCommand {contentId} {
     set messages {"Nachricht"}
     set result 0
     return "$messages $result"
   }
   ```

3. Schließlich muss die Prozedur im sicheren Interpreter registriert werden:

```
safeInterp alias callbackCommand callbackCommand
```

4. Starten Sie den Content Manager neu, damit die Skriptdatei eingelesen und die Prozedur verfügbar gemacht wird:

```
instance/myInstance/bin/rc.npsd restart CM
```

5. Nun können Sie in der jeweiligen Funktion die Prozedur folgendermaßen aufrufen und die Rückgabewerte entsprechend verwenden:

```
CM> objClass withName document get completionCheck
set returnValue [callbackCommand $contentId]
set messages [lindex $returnValue 0]
set result [lindex $returnValue 1]
```

Wenn eine Prozedur aus Funktionen heraus aufgerufen wird, die unterschiedliche Rückgabewerte erwarten, lassen Sie die Prozedur eine Liste mit allen erforderlichen Rückgabewerten zurückgeben. Ermitteln Sie dann in den aufrufenden Funktionen die erforderlichen Werte mit Hilfe von `lindex`. Wenn die Funktionen, die die Prozedur aufrufen, der Prozedur unterschiedliche Argumente übergeben müssen, verwenden Sie ein zusätzliches Argument, das die aufrufende Funktion identifiziert. Machen Sie dann die Berechnungen in der Prozedur vom Wert des zusätzlichen Arguments abhängig.

5.7 Datumsformate umwandeln

Die folgende Tcl-Prozedur wandelt ein 14-stelliges CMS-Datum in einen Unix-Timestamp um:

```
# $Id: nps2unixDate.tcl,v 1.5 2001/12/19 16:09:28 benjamin Exp $
#
# (c) 2001 Infopark AG
# nps2unixDate
#
# Purpose:
#   Converts a CMS timestring (e.g. 20060101123000) to
#   a Unix timestamp (seconds since 01.01.1970).
#
# Parameters:
#   CMS timestring (format: YYYYMMDDHHmmSS)

proc nps2unixDate {timeStamp} {
    # String must contain 14 digits.
    # regular expressions rule!
    if {[regexp "^(1|2)(0|9)\\d\\d(0|1)\\d(0|1|2|3)\\d(0|1|2)\\d\[0-5\]\\d\[0-5\]\\d\$"
 $timeStamp] == "0"} {
        error "nps2unixDate: Parameter is not a valid CMS timestamp"
    }
    set year   [string range $timeStamp 0 3]
    set month  [string range $timeStamp 4 5]
    set day    [string range $timeStamp 6 7]
    set hour   [string range $timeStamp 8 9]
    set minute [string range $timeStamp 10 11]
    set second [string range $timeStamp 12 13]
    set unixTime [clock scan "$month/$day/$year $hour:$minute:$second" -gmt true]
    return $unixTime
}
```

Die umgekehrte Konvertierung kann mit der folgenden Prozedur durchgeführt werden:

```
# $Id: unix2npsDate.tcl,v 1.3 2001/12/19 16:08:28 benjamin Exp $
#
# (c) 2001 Infopark AG
# unix2npsDate
#
# Purpose:
#   Converts a Unix timestamp (seconds since 01.01.1970) to
#   a CMS timestring (e.g. 20060519143000).
#
# Parameters:
#   Unix timestring

proc unix2npsDate {timeStamp} {
    # The string must not contain any non-digits
    if {[regexp "\[^\\d\]" $timeStamp]} {
        error "unix2npsDate: Parameter is not a valid unix timestamp"
    }
    # NOTE
    # The string must not exceed 2.000.000.000 which is
    # Wed, May 18 05:33:20 MET DST 2033
    # Actually, tcl is able to scan timestamps beyond this date
    if {$timeStamp > "2000000000"} {
        error "unix2npsDate: Parameter is too large"
    }
    return [clock format $timeStamp -format "%Y%m%d%H%M%S"]
}
```

Speichern Sie diese Skripte bitte in einer einzigen Tcl-Datei ab. Öffnen Sie eine Tcl-Shell und lesen Sie die Tcl-Datei mit dem Befehl `source` ein. Nun können Sie die Prozeduren verwenden. Beispiel:

```
nps2unixDate 20110519130500
```

6

6 Funktionsweise der Template Engine

Die Template Engine spielt auf dem Live-System eine zentrale Rolle. Sie hat die Aufgabe, aktualisierten Content, den sie vom Content Management Server erhält, zu exportieren, d. h. mittels Layouts Webseiten zu erzeugen. Wenn die Verity Search Cartridge eingebunden ist, so aktualisiert sie gleichzeitig deren Indizes. Den gegebenenfalls installierten Portal Manager versorgt sie mit Informationen über Dokument-Zugriffsbeschränkungen, Channels und News.

Template Engine und Search Engine Server arbeiten mit einer zweiten, offline geschalteten Verzeichnishierarchie, die die Template Engine auf Anweisung des Content Managers live schaltet. Details über diese Verzeichnishierarchien finden Sie im Abschnitt Die Verzeichnishierarchien.

6.1 Import und Export

Die Template Engine arbeitet in Phasen. Abwechselnd importiert sie Informationen über CMS-Datei-Versionen vom Content Management Server (Import-Phase) und generiert Webdokumente, die sie dann live schaltet (Export-Phase). Die Export-Phase ist in mehrere Teilphasen unterteilt. Die einzelnen Phasen haben die im Folgenden beschriebenen Aufgaben:

- `import`: Die Template Engine erhält vom Content Manager so genannte Update-Records, d. h. Informationen über aktualisierte Dateien. Ist der Infopark Portal Manager eingebunden, so erhält sie auch Update-Records, die sich auf Channels und Newsartikel beziehen.
- `importCB` (ab Version 6.7.2): Die Template Engine ruft eine Callback-Funktion auf, die Zugriff auf ein Journal der Import-Aktionen hat und weitere Aktionen auslösen kann.
- `export-init`: Die Exportanforderungen, die sich aus den Update-Records ergeben, werden bearbeitet und gruppiert. Die Gruppierung erlaubt es, mehrere Slaves Dateien exportieren zu lassen.
- `export-fill`: Die Exportanforderungen werden ausgeführt, d. h. Dateien in die offline geschaltete Verzeichnishierarchie exportiert. Diese Phase (und damit der gesamte Export) wird abgebrochen, wenn eine konfigurierbare Anzahl Dateien nicht exportiert werden konnte (Systemkonfigurationseintrag `tuning.export.acceptableFailures`). Ist die Verity Cartridge eingebunden, so werden die erzeugten Webdokumente parallel indiziert.
- `export-fillCB`: Die Template Engine ruft eine Callback-Funktion auf, mit der die aktualisierte Verzeichnishierarchie beliebig repliziert werden kann.
- `export-switch`: Die aktualisierte Verzeichnishierarchie wird live geschaltet, indem die Ziele zweier symbolischer Links (online und offline geschaltete Hierarchien) vertauscht werden.
- `export-switchCB`: Die Template Engine ruft eine weitere Callback-Funktion auf, mit der die Ziele der symbolischen Links auch auf anderen Webserver-Systemen vertauscht werden können.
- `export-sync`: Die jetzt offline geschaltete Verzeichnishierarchie wird auf den aktuellen Stand gebracht.

- `export-syncCB`: Mit dieser Callback-Funktion kann eine entfernte, jetzt offline geschaltete Verzeichnishierarchie auf den aktuellen Stand gebracht werden.
- `export-sync-finished`: Das während der Phase `export-fill` geschriebene und in der Phase `export-sync` abgearbeitete Journal der geänderten Dateien wird rotiert. Damit ist die Export-Phase beendet, und die Template Engine geht wieder in die Import-Phase.

Die Template Engine geht in die Export-Phase, wenn sie vom Content Management Server eine entsprechende Anweisung oder einen der Tcl-Befehle `app publish` und `app export` erhält. Während mit `app publish` ein vollständiger Export- und Veröffentlichungszyklus durchlaufen oder ein unterbrochener Zyklus wieder aufgenommen wird, exportiert `app export` die Dateien und kehrt anschließend in die Import-Phase zurück. Da die Template Engine Dateien inkrementell exportiert, kann man `app export` nutzen, um den Veröffentlichungsvorgang (`app publish`) auf die Umschaltung und Synchronisation der Verzeichnisse zu reduzieren und dadurch schneller zu machen. Diese Befehle der Template Engine zu verwenden, sollte jedoch nur im Fehlerfall erforderlich sein, weil der Server normalerweise entsprechende Anweisungen als Update-Records vom Content Manager (mittels System-Jobs) erhält. Einzelheiten zu diesen Jobs können Sie der Dokumentation zu Administration und Layout entnehmen.

Schlägt eine der Export-Phasen bis einschließlich `export-switch` fehl, so kehrt die Template Engine zur Import-Phase zurück. Schlägt dagegen eine der darauf folgenden Phasen fehl, so bleibt der Fehlerzustand bestehen. In diesem Fall ist es zwingend erforderlich, die Fehlerursache zu beseitigen, um anschließend mit `app publish` den Export-Vorgang fortzusetzen, d. h. die Verzeichnishierarchien umzuschalten und die erforderlichen Synchronisationsvorgänge auszuführen. Kann die Fehlerursache nicht beseitigt werden (weil etwa ein entfernter Rechner nicht erreichbar ist), so können notfalls die Callback-Funktionen außer Kraft gesetzt werden, indem beispielsweise ihr Code auskommentiert wird (erfordert, dass der Server neu gestartet wird).

Die Callback-Funktionen

Die Template Engine liefert normalerweise die Inhalte vom selben System aus, auf dem sie installiert ist, d. h. sie exportiert die Dateien in eine lokale, offline geschaltete Verzeichnishierarchie, die dann online geschaltet wird. Die ehemalige Online-Hierarchie wird abschließend mit der aktuellen Onlinehierarchie synchronisiert, damit dem folgenden Export die aktuellen Daten zugrunde liegen.

Bei verteilten Systemen lassen sich diese Vorgänge mit den oben beschriebenen Callback-Funktionen auch auf einem oder mehreren entfernten Rechnern durchführen. Die Callback-Funktionen rufen die unten beschriebenen kundenspezifischen Prozeduren auf. Bitte beachten Sie, dass die Callback-Funktionen einen Fehler mit dem Befehl `error` anzeigen müssen. Wird eine Callback-Funktion nochmals aufgerufen, nachdem sie fehlgeschlagen ist, so muss das Ergebnis das gleiche wie beim ersten Aufruf sein, wenn dieser erfolgreich gewesen wäre. Callback-Funktionen dürfen den Inhalt des Export-Verzeichnisbaums nicht modifizieren.

Die Callback-Funktionen müssen als Serverkommandos in einer Tcl-Skript-Datei definiert werden, die im instanzenspezifischen Verzeichnis `script/cm/serverCmds` abgelegt wird.

Phase importCB (ab Version 6.7.2), Callback-Funktion *importCallback*

Beim Importieren von Update-Records erstellt die Template Engine ein Journal der Änderungen an CMS-Dateien. Diese Callback-Funktion kann das Journal auswerten und weitere Aktionen auslösen.

Phase export-fillCB, Callback-Funktion *exportFillCallback*

Vor dem Aufruf der Callback-Funktion schließt die Template Engine den Export in die offline geschaltete lokale Verzeichnishierarchie vollständig ab. Nach der Ausführung muss es eine Offline-

Verzeichnishierarchie auf dem entfernten Rechner geben, die ebenfalls vollständig ist. Die Callback-Funktion muss gefahrlos mehrmals ausgeführt werden können.

Phase export-switchCB, Callback-Funktion *exportSwitchCallback*

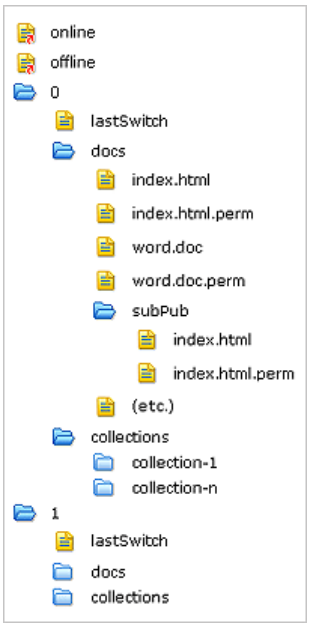
Aufgabe der Callback-Funktion ist es, auf einem entfernten Rechner die online und offline geschalteten Verzeichnishierarchien zu vertauschen. Beim Aufruf der Callback-Funktion enthält die lokale, online geschaltete Verzeichnishierarchie die neuen Daten, die offline geschaltete Hierarchie die bisherigen Online-Daten. Ferner muss auf dem entfernten Rechner der Zustand nach erfolgreicher Ausführung des `exportFillCallback` herrschen, d. h. die Offline-Hierarchie enthält die neuen, die Online-Hierarchie noch die alten Daten. Nach erfolgreicher Ausführung muss es auf dem entfernten Rechner je eine Online- und eine Offline-Verzeichnishierarchie geben, die vertauscht wurden, so dass deren Inhalte denen des lokalen Systems entsprechen. Im Fehlerfall dürfen die entfernten Verzeichnishierarchien nicht vertauscht worden sein.

Phase export-syncCB, Callback-Funktion *exportSyncCallback*

Diese Callback-Funktion hat die Aufgabe, auf dem entfernten Rechner die Offline-Hierarchie auf den Stand der Online-Hierarchie zu bringen. Bevor die Callback-Funktion aufgerufen wird, hat die Template Engine die lokalen Hierarchien abgeglichen. Die Callback-Funktion wird nur aufgerufen, wenn der `exportSwitchCallback` erfolgreich ausgeführt wurde, d. h. die aktuellen Daten online und die veralteten offline sind. Nachdem die Callback-Funktion erfolgreich ausgeführt wurde, müssen die beiden entfernten Hierarchien die gleichen Inhalte haben.

6.2 Die Verzeichnishierarchien

Die Template Engine legt alle Daten, die für das Live-System relevant sind, in einer Verzeichnishierarchie ab. Ausgangspunkt dieser Hierarchie ist das jeweils instanzenspezifische Verzeichnis `export`. Die Hierarchie hat die folgende Struktur:



`online` und `offline` sind symbolische Links, von denen einer auf das Verzeichnis `0` und der andere auf `1` zeigt. Die Template Engine vertauscht die Ziele dieser Links, um die jeweils offline geschaltete Hierarchie live zu schalten und umgekehrt.

Die beim Export erzeugten Webdokumente legt die Template Engine im Verzeichnis `docs` ab. Parallel zu jedem Dokument wird eine zusätzliche, bis auf die Endung `.perm` gleichnamige Datei erzeugt, in der die Namen der zugriffsberechtigten Gruppen abgelegt werden, also der Gruppen, denen im Content Management Server das dateispezifische Liveserver-Leserecht (`permissionLiveServerRead`) zugewiesen wurde. Diese Informationen können vom Portal Manager ausgewertet werden, sobald die Verzeichnishierarchie live geschaltet wurde. `collections` enthält bei eingebundener Infopark Search Cartridge die Collections, die für die Suche auf dem Liveserver benötigt werden. In diese Collections lässt die Template Engine den Search Engine Server die unter `docs` abgelegten Dokumente indizieren.

Die Datei `lastSwitch` schließlich dient dazu, den Zeitpunkt der letzten Aktivierung einer Verzeichnishierarchie aufzuzeichnen. Indem Applikationen den Inhalt dieser Datei in regelmäßigen Abständen auswerten, können sie feststellen, ob sich der Inhalt der betreffenden Verzeichnishierarchie geändert hat.

6.3 Abhängigkeiten

Die Template Engine hat unter anderem die Aufgabe, den Export von CMS-Dateien zu optimieren. Sie einzusetzen schont die Ressourcen des betreffenden Rechners, und Websites können schneller publiziert werden.

Das größte Optimierungspotenzial liegt darin, Dateien nur dann zu exportieren, wenn sich die ihnen entsprechenden Webdokumente gegenüber der aktuellen Version ändern. Um dies vorhersagen zu können, zeichnet die Template Engine Abhängigkeiten zwischen Dateien auf, die im Folgenden erläutert werden.

Eine Datei muss nicht nur dann neu exportiert werden, wenn sie selbst geändert wurde. Bezieht eine Datei Daten aus einer anderen Datei ein (über Links oder NPSOBJ-Anweisungen), so ist sie von dieser abhängig und muss auch dann exportiert werden, wenn sich nur Bestandteile der einbezogenen Datei geändert haben.

Die Template Engine zeichnet solche Abhängigkeiten von CMS-Dateien auf, während sie sie exportiert. Da beim ersten Export einer Ordnerhierarchie immer alle Dateien exportiert werden, ist sichergestellt, dass alle Abhängigkeiten aufgezeichnet werden. Später, wenn der Datenbestand mit Hilfe von Update-Records aktualisiert wird, prüft die Template Engine mit Hilfe der Aufzeichnungen, welche exportierten Dokumente veraltet sind und löscht diese. Wird die Ordnerhierarchie schließlich publiziert, erzeugt die Template Engine die zu aktualisierenden Dokumente.

Infopark CMS Fiona kennt die folgenden Abhängigkeitstypen. Die Details sind in der rechten Spalte aufgeführt. Die Datei, von der eine Datei abhängig ist, wird *Zieldatei* genannt.

Abhängigkeitstyp	Details
`object (1)`	**Beschreibung**: Die Datei hängt vom Namen oder der Position der Zieldatei in der Ordnerhierarchie ab. Das ist auch der Fall, wenn der Name eine Komponente in einem Linkzielpfad ist. **Auslösung**: Parameter `name, objectsToRoot, object, parent, visibleName/visiblePath` (erzeugt Abhängigkeit für alle `objectsToRoot` einer Linkzieldatei). **Auswirkung**: Zur Datei gehörende Dateien werden gelöscht, wenn `parent` oder `name` der Zieldatei geändert oder die Zieldatei gelöscht wird.

`content (2)`	**Beschreibung**: Die Datei hängt von einer anderen Datei oder von anderen Feldern ab als bei *object*-, *reference*- und *children*-Abhängigkeiten. **Auslösung**: Abfrage von Parametern, die keine der oben genannten Abhängigkeiten erzeugen. **Auswirkung**: Wenn die Zieldatei freigegeben wird, wird die Datei aus dem Cache gelöscht.
`reference (3)`	**Beschreibung**: Die Datei hängt nur von sich selten ändernden Daten der Zieldatei ab, wie sie etwa für Links und Inhaltsübersichten benötigt werden. Wenn die Zieldatei erneut freigegeben wird und diese Daten dabei unverändert bleiben und keine anderen (insbesondere: Versions-)Abhängigkeiten erzeugt werden, werden die exportierten Dateien nicht aus dem Cache gelöscht. **Auslösung**: Abfrage des Zustands der Zieldatei. Referenz-Parameter: `title`, `contentType`, `objClass` bei Dateien, `destinationUrl`, `displayTitle` bei Links sowie die im Systemkonfigurationseintrag `export.referenceAttributes` aufgeführten Felder (geben Sie keine Felder vom Typ `linklist` an). **Auswirkung**: Die Datei wird nur dann gelöscht, wenn sich der Wert eines Referenz-Parameters geändert hat.
`children (4)`	**Beschreibung**: Datei hängt von der Liste der (freigegebenen) Dateien in einem Ordner ab. **Auslösung**: Abfrage von `children`, `prev`, `next`. **Auswirkung**: Die zur Datei gehörenden Dateien werden gelöscht, wenn Dateien im Ordner (neu) freigegeben oder zurückgezogen werden (oder auf andere Art ihre freigegebene Version verlieren).
`usesAll (5)`	**Beschreibung**: Datei hängt von den Daten einer großen Zahl oder unbestimmten Menge von Dateien ab. **Auslösung**: automatisch bei Überschreitung von `usesAllThreshold` Abhängigkeiten oder Abfrage der Parameter: `superObjects`, `superLinks`, `hasSuperLinks`. **Auswirkung**: Zur Datei gehörende Dateien werden bei jeder beliebigen Änderung des Datenbestandes der Template Engine gelöscht.

In systemExecute-Prozeduren können während des Exports bei Bedarf Abhängigkeiten erzeugt werden. Dies kann erforderlich sein, wenn in der Prozedur Felder von Dateien ausgelesen werden, die ansonsten nicht referenziert werden, so dass für sie keine Abhängigkeit erzeugt wird.

6.4 Den Exportvorgang beschleunigen

Was den Export verlangsamt

Insbesondere bei großen Websites bewirken manche Änderungen an Dateien, dass die Template Engine viele andere Dateien als veraltet betrachtet, weil sie von der geänderten Datei abhängig sind. Dies führt dazu, dass der Veröffentlichungsvorgang länger dauert, weil sehr viele Dateien neu exportiert werden müssen.

Große Auswirkungen haben:

- Änderungen an Layoutdateien. Sie führen dazu, dass der Teilbaum, in dem das Layout wirksam ist, veraltet ist und neu exportiert werden muss. Dies gilt unabhängig davon, ob das Layout während des Exports tatsächlich verwendet wird oder nicht.
- Namensänderungen bei Ordnern. Sie führen ebenfalls dazu, dass alle Dateien in diesem Teilbaum und darüber hinaus alle Dateien mit Links, die auf eine Datei in diesem Teilbaum verweisen, neu exportiert werden müssen, weil sich der Pfad der im Ordner enthaltenen Dateien geändert hat.

Geringere Auswirkungen haben:

- Änderungen der Dateiendung. Sie haben zur Folge, dass alle Dateien mit Links auf die betreffende Datei neu exportiert werden müssen.
- Größenänderungen bei Bildern. Sie führen ebenfalls dazu, dass alle Datein mit Links auf die betreffende Datei neu exportiert werden müssen, weil die Links Größenangaben enthalten können.
- Änderungen am Feld `suppressExport` . Referenzen (wie Links) auf die betreffende Datei werden beim Export sichtbar gemacht oder nicht, abhängig vom Wert dieses Feldes. Seinen Wert zu ändern, bedeutet deshalb, dass die Dateien, die die Referenzen enthalten, neu exportiert werden müssen.
- Werden Dateien angelegt oder Dateien gelöscht, so veralten alle Dateien, die Inhaltsübersichten (`toclist`) der betreffenden Ordner enthalten.
- Änderungen an den meisten Versionsfeldern führen dazu, dass neben der betreffenden Datei alle Dateien veralten, bei deren Export Feldwerte aus der ursprünglichen Datei abgefragt werden.

Maßnahmen zur Verkürzung der Exportzeit

Wenn Sie Inhalte statisch mit `includetext` einbinden, ist es empfehlenswert, die Inhalte stattdessen dynamisch mit `insertvalue = "dynamiclink"` einzubinden. Ferner sollten In Layoutdateien die Werte `hasSuperLinks`, `superLinks` und `superObjects` sehr vorsichtig eingesetzt werden, da sie eine Abhängigkeit der gefragten Datei von allen anderen Dateien erzeugen.

Die Exportzeit resultiert demnach nicht allein aus der Anzahl der exportierten Dateien, sondern auch aus der Komplexität der Layoutstruktur und dem Berechnungsaufwand, der je exportierter Datei entsteht. Dieser Aufwand ergibt sich vor allem aus der Häufigkeit und der Art der Werte, die in Layoutdateien abgefragt werden. So ist es beispielsweise wesentlich zeitaufwändiger, Datei- oder Linklisten zu generieren als einzelne Feldwerte wie `title` zu ermitteln.

Abhängigkeiten analysieren

Ob die Abhängigkeiten die Ursache eines langsam laufenden Exports sind, lässt sich folgendermaßen ermitteln.

- Führen Sie im Content Manager den Befehl `incrExport reset` aus und übertragen Sie Ihren Content mit `job withName systemTransferUpdates exec` zur Template Engine.

  Achtung: Während dieser Befehl ausgeführt wird, dürfen keine Update-Records erzeugt werden. Der Befehl incrExport reset darf nur im Single-Modus verwendet werden und nur ausgeführt werden, wenn mit Sicherheit keine schreibenden Zugriffe auf den Datenbestand stattfinden, da andernfalls die exportierten Daten unvollständig sind oder zerstört werden. Stellen Sie daher sicher, dass während der Ausführung des Befehls der CM nicht als Server läuft und keine anderen CMs im Single-Modus gestartet werden oder laufen.
- Nachdem alle Update Records übertragen wurden, führen Sie in der Template Engine den Befehl `app export` aus.
- Untersuchen Sie unmittelbar nach dem Export die erzeugten Abhängigkeiten vom Typ `usesAll`. Sie können hierfür den Befehl `statistics compute name all` verwenden sowie die Protokolldatei der Template Engine untersuchen (jedes Mal, wenn eine Abhängigkeit dieses Typs erzeugt wird, wird eine Warnung ins Protokoll geschrieben).
- Stellen Sie fest, weshalb die betreffenden Dateien von sehr vielen anderen Dateien abhängen. Zu diesem Zweck kann der Wert des Systemkonfigurationseintrags `tuning.export.usesAllThreshold` vergrößert (die Template Engine muss danach neu gestartet werden) und können anschließend einige der betreffenden Dateien aktualisiert werden (`obj touch` und `obj updateCache`). Werden für die Dateien nun keine `usesAll`-Abhängigkeiten mehr erzeugt, so erhalten Sie mit `obj withId` *`id`* `get dependencies` auch die IDs der Dateien, von denen die betreffenden Dateien abhängen.

  Möchten Sie wissen, welche Layoutdatei und welche NPSOBJ-Anweisung daraus für die Erzeugung einer bestimmten Abhängigkeit verantwortlich ist, so erhöhen Sie im Systemkonfigurationseintrag `server.log.logger` mit dem Namen `info` den Wert für das Attribut `level` auf 3. Starten Sie anschließend die Template Engine neu. Aktualisieren Sie eine der betreffenden Dateien mittels `obj withPath` *`path`* `updateCache`). Sie können nun dem `info.log` die oben genannten Details zum Export dieser Datei entnehmen.

Um die Anzahl der Abhängigkeiten zu reduzieren, gehen Sie bitte wie oben beschrieben vor, d. h. verringern Sie die Komplexität Ihrer Layoutstruktur und den Berechnungsaufwand in den Vorlagen.

7

7 Fehlermeldungen

In diesem Abschnitt werden Fehlermeldungen erläutert, deren Ursache nicht immer klar zu erkennen ist.

Fehlercode	Erläuterung
10005	Die Anzahl der Response-Elemente entspricht nicht der Anzahl der Request-Elemente in der Anfrage.
20005	Um den Verwalter eines Benutzers oder einer Benutzergruppe zu setzen, benötigt der angemeldete Benutzer das globale Recht `permissionGlobalUserEdit`.
20006	Der angemeldete Benutzer kann andere Benutzer nicht in Gruppen aufnehmen oder aus Gruppen entfernen, da er das globale Recht `permissionGlobalUserEdit` nicht hat.
20007	Der angemeldete Benutzer kann keine globalen oder dateispezifischen Rechte erteilen, da er die hierfür erforderlichen Rechte (`permissionGlobalRoot` bzw. `permissionRoot`) nicht hat.
20008	Der angemeldete Benutzer kann keine globalen oder dateispezifischen Rechte entziehen, da er die hierfür erforderlichen Rechte (`permissionGlobalRoot` bzw. `permissionRoot`) nicht hat.
20009	Der angemeldete Benutzer hat nicht das erforderliche Recht (`permissionGlobalUserEdit`), um einen anderen Benutzer zu sperren oder zu entsperren.
20012	Der Benutzer darf die Transaktion nicht ausführen.
20013	Der angemeldete Benutzer benötigt das globale Recht `permissionGlobalExport`, um die Versionsfunktion `debugExport` verwenden oder auf eines der Versionsfelder `encodedExportBlob`, `exportBlob` und `exportFiles` zugreifen zu können.
20014	Der angemeldete Benutzer darf die Felder der Version nicht auslesen, da er das Leserecht (`permissonRead`) für die Datei, zu der die Version gehört, nicht hat.
20015	Der angemeldete Benutzer darf die Parameter der Datei nicht auslesen, da er das Leserecht (`permissonRead`) für die Datei nicht hat.
20016	Der Benutzer hat nicht die Rechte, die erforderlich sind, um eine Datei mit der angegebenen Vorlage in diesem Ordner anzulegen. Diese Rechte sind das globale Anlegerecht der Vorlage und das Dateianlegerecht in dem betreffenden Ordner.
20020	Der angemeldete Benutzer hat versucht, die Arbeitsversion einer Datei freizugeben, ohne dazu berechtigt zu sein.
20021	Der eingeloggte Benutzer hat versucht, den Workflow einer Datei abzubrechen und einen neuen zu beginnen, ohne dazu berechtigt zu sein.

33008	Der Wert des angegebenen Bezeichners ist keine URL und kann auch nicht in eine URL umgewandelt werden.
36001	Beim Aufruf eines zusätzlichen (kundenspezifischen) Befehls wurde die konfigurierte, damit assoziierte Tcl-Prozedur nicht gefunden.
36024	Diese Meldung betrifft den Infopark Portal Manager. Bei einem Kommando, das genau einen der genannten Werte erwartet, wurde keiner dieser Werte angegeben.
36025	Diese Meldung betrifft den Infopark Portal Manager. Bei einem Kommando, das genau einen der genannten Werte erwartet, wurde keiner dieser Werte angegeben.
40005	Der Benutzer ist nicht der Administrator der Datei und muss deshalb der Bearbeiter sein, um die Workflowaktion ausführen zu können.
40009	Der Benutzer hat die Datei bereits unterschrieben. Da er nicht der Administrator der Datei ist und der Workflow Mehrfachunterzeichnungen verbietet, darf er kein zweites Mal unterschreiben.
40010	Die Datei kann nicht eingereicht oder abgezeichnet, sondern nur freigegeben werden.
40011	Die `commit` -Operation kann nicht ausgeführt werden, da die Datei noch nicht von allen Bearbeitergruppen im Workflow bearbeitet wurde. Nur der Dateiadministrator und ein Superuser hat das Recht, die Datei trotzdem einzureichen oder freizugeben.
40012	Die Datei kann nicht eingereicht oder freigegeben werden, da der Arbeitscontent unvollständig ist. Dies bedeutet, dass die Arbeitsversion defekte Links oder Felder mit nicht zulässigen Inhalten enthält.
40013	Der Benutzer ist weder Administrator der Datei noch Superuser und kann die Datei nicht vorzeitig freigeben.
40014	Der Benutzer ist weder Administrator der Datei noch Superuser und kann sie nicht vorzeitig freigeben.
40015	Der Benutzer ist weder Administrator der Datei noch Superuser, sodass er Mitglied in der letzten Prüfergruppe sein muss, um die Datei freigeben zu können.
40016	Es gibt eine eingereichte Version, sodass nur Mitglieder der verbleibenden unterschriftsberechtigten Gruppen die Datei ablehnen dürfen.
40017	Es wurde versucht, ein Unterschriftenfeld mehrfach zu verwenden. Dies ist nicht zulässig.
50009	Der angegebene Name enthält unzulässige Zeichen. Folgende Zeichen können in Namen verwendet werden: A-Z, a-z, 0-9 und _.
50014	Bei einem `sortKey`-Feld wurde ein Feld angegeben, das nicht existiert.
50015	Für das `sortOrder`-Feld wurde ein unzulässiger Wert angegeben.
50024	Die angegebene Spezifikation der `presetAttributes` (vorbelegte Felder) ist nicht korrekt. Es konnte daraus keine gültige Liste gebildet werden.
50025	Der angegebene Name enthält unzulässige Zeichen. Folgende Zeichen können in Namen verwendet werden: A-Z, a-z, 0-9 und _.
50034	Der angegebene Name enthält unzulässige Zeichen. Folgende Zeichen können in Namen verwendet werden: A-Z, a-z, 0-9 und _.
50037	Der angegebene Name enthält unzulässige Zeichen. Folgende Zeichen können in Namen verwendet werden: A-Z, a-z, 0-9 und _.
50038	Beim angegebenen Feld wurde in der Liste der `presetAttributes` der genannte Wert angegeben, der jedoch für dieses Feld unzulässig ist.

50040	Das angegebene Tag enthält unzulässige Zeichen.
50042	Die erlaubten Dateinamenserweiterungen können im Systemkonfigurationseintrag `mimeTypes` konfiguriert werden.
50043	Die angegebene Definition enthält ein nicht existierendes Feld oder eine nicht existierende Benutzergruppe.
50044	Der angegebene Name enthält unzulässige Zeichen. Folgende Zeichen können in Namen verwendet werden: A-Z, a-z, 0-9 und _.
50045	In einem Update-Record wurde ein unbekannter Update-Typ angegeben.
50063	Job-Namen dürfen nicht mit `system` und nicht mit dem Unterstrich beginnen. Erlaubte Zeichen in Job-Namen sind a - z, A - Z, 0 - 9 und der Unterstrich.
55000	Die Zeichenfolge besteht aus zwei Zeichen, die sich jeweils im Bereich von 0 bis 9, a bis z oder A bis Z befinden. Die Zeichen werden zur Verschlüsselung verwendet.
55001	Mit diesem Fehler werden Payloads beantwortet, die in einer Version des XML-Schnittstellenprotokolls formuliert sind, die der Server nicht unterstützt. Zum Verhalten des Servers in diesen Fällen siehe die XML-Referenz.
55003	Der Link gehört zu einem Tag, in dem es nicht erlaubt ist, einen Anker anzugeben.
55004	Der Link gehört zu einem Tag, das Frame-Targets nicht unterstützt.
55005	Die beim Kopieren oder Verschieben angegebene Datei kann nicht als Ziel akzeptiert werden. Ursache kann sein: eine Datei mit dem Namen der betreffenden Datei existiert bereits in dem Ordner; die Datei ist kein Ordner, liegt innerhalb des zu verschiebenden Hierarchiezweiges oder darf keine Unterordner mit der Vorlage der Quelldatei haben. Möglicherweise haben Sie auch nicht die erforderlichen Rechte.
55006	Eine Datei mit dem gewünschten Namen existiert bereits im Verzeichnis.
55011	Der angegebene Name enthält unzulässige Zeichen. Folgende Zeichen können in Namen verwendet werden: A-Z, a-z, 0-9 und _.
55015	Die angegebene oder bei der Operation verwendete Namenserweiterung ist nicht in den `validContentTypes` der Vorlage enthalten, das der Datei zugeordnet ist.
55023	Die angegebene Vorlage kann bei dieser Datei nicht eingetragen werden. Es existiert nicht oder ist nicht in den `validSubObjClasses` der Vorlage des darüber liegenden Ordners enthalten.
55036	Einem Link konnte der angegebene Feldname nicht zugewiesen werden, weil es das Feld in der Version, zu der der Link gehört, nicht gibt.
55046	Dieser Fehler tritt auf, wenn man versucht, ein Miniaturbild (Thumbnail) für eine Datei zu erzeugen, die nicht vom Typ *Bild* oder *Ressource* ist.
55047	Der angegebene Zeichensatz existiert nicht oder wurde nicht definiert.
55054	Bei dem Dateibefehl `createAndLoad` wurde als `file` ein Pfad mit mehr als zwei Komponenten angegeben, oder der Pfad ist absolut oder verweist auf das übergeordnete Verzeichnis.
55055	Bei dem Versuch, auf ein temporäres Verzeichnis zuzugreifen, nachdem der Upload Wizard aufgerufen wurde, wurde ein Pfad angegeben, der aus mehreren Teilen besteht. Dies ist nicht zulässig.
60014	Dies ist ein interner Fehler. Bitte wenden Sie sich an den Support (support@infopark.de).
60015	Dies ist ein interner Fehler. Bitte wenden Sie sich an den Support (support@infopark.de).

60016	Diese Meldung betrifft den Infopark Portal Manager. Der angegebene Pfad existiert nicht.
60033	Das angegebene Layout wird beim Export benötigt, ist jedoch nicht vorhanden oder nicht freigegeben.
80001	Bevor ein Ordner gelöscht werden kann, müssen die darin enthaltenen Dateien gelöscht werden.
80009	Das Feld kann nicht gelöscht werden, da es noch in den angegebenen Vorlagen verwendet wird.
80010	Das angegebene Feld kann nicht gelöscht werden, da es noch in den angegebenen Workflows für Unterschriften verwendet wird.
80011	Der Workflow kann nicht gelöscht werden, da er in den angegebenen Vorlagen im vorbelegten `workflowName` -Feld referenziert wird.
80014	Bei den genannten Benutzern muss der Wert des Feldes geändert werden, um den Aufzählungswert löschen zu können.
85013	Requests in der Request-Warteschlange des Search Engine Servers werden temporär nicht abgearbeitet, während die exportierten Dokumente live geschaltet werden.
85015	Der Request an die XML-Schnittstelle wurde nicht bearbeitet, weil ein vorhergehender Request, der als `preclusive` markiert war, fehlgeschlagen ist.
85020	Es wurde versucht, den Hauptinhalt einer Version zu bearbeiten. Dies ist jedoch nicht möglich, wenn die Datei, zu der die Version gehört, auf einer Vorlage basiert, bei der ein Hauptlayout angegeben wurde.
85028	Felder werden automatisch zur Basisgruppe hinzugefügt, wenn sie aus anderen Feldergruppen gelöscht oder in die Vorlage aufgenommen werden.
85031	Der Content ist ein eingereichter oder freigegebener Content und damit nicht änderbar.
85033	Die aufgeführten Felder können nicht in die Liste `mandatoryAttributes` aufgenommen werden, weil sie nicht in der Felderliste der Vorlage enthalten sind.
85034	Die aufgeführten Felder können zur Liste `presetAttributes` nicht hinzugefügt werden, weil sie nicht in der Felderliste der Vorlage enthalten sind.
85035	Die aufgeführten Felder können zur Liste `presetFromParentAttributes` nicht hinzugefügt werden, weil sie nicht in der Felderliste der Vorlage enthalten sind.
85036	Der in der Vorlage angegebene Dateityp lässt keine Unterdateien zu. Der Fehler tritt auf, wenn man einer Vorlage, mit der kein Ordner (sondern ein anderer Dateityp) definiert wird, eine erlaubte Vorlage für darin enthaltene Dateien zuzuweisen versucht.
85037	Es wurde versucht, bei einer Vorlage, die Layouts, Bilder oder Ressourcen definiert, ein Hauptinhaltslayout anzugeben. Bei solchen Vorlagen kann kein Hauptinhaltslayout angegeben werden.
85038	In der Definition einer Vorlage wurde versucht, den Hauptinhalt vorzubelegen. Dies ist jedoch nicht möglich, wenn ein Hauptinhaltslayout angegeben ist.
85039	In der Definition einer Vorlage wurde versucht, den Hauptinhalt mit dem Wert des Hauptinhalts des darüber liegenden Ordners vorzubelegen. Dies ist jedoch nicht möglich, wenn ein Hauptinhaltslayout angegeben ist.
85041	Kontextlinks müssen auf CMS-Dateien verweisen.
90007	Das angegebene Feld kommt im XML-Dokument vor, in der zum Dokument gehörenden Vorlage ist es jedoch nicht enthalten.

100000	Der angegebene kundenspezifische Befehl (*engl.* "custom command") konnte nicht ausgeführt werden.
100001	Bei der Ausführung eines Custom Commands konnte nicht in eine Binärdatei geschrieben werden.
100016	Die angegebenen Werte können nicht verwendet werden. Der Wert für den Konfigurationseintrag `master.maxSlaves` muss immer höher sein als der für `master.minIdleSlaves`.
100029	Dies ist ein Datenbankfehler.
100030	Dies ist ein Datenbankfehler.
100031	Dies ist ein Datenbankfehler.
100032	Dies ist ein Datenbankfehler.
100037	Dies ist ein Datenbankfehler.
100038	Dies ist ein Datenbankfehler.
100039	Eine CMS-Applikation hat versucht, auf eine nicht vorhandene Datei lesend zuzugreifen. Der Fehler kann beispielsweise dann auftreten, wenn das Streaming-Interface eine Datei zu einem Ticket ausliefern möchte, diese Datei jedoch gelöscht wurde.
100040	Dies ist ein Datenbankfehler.
100043	Dies ist ein Datenbankfehler.
100044	Dies ist ein Datenbank- oder ein interner Systemfehler. Bitte wenden Sie sich an den Support (support@infopark.de).
100052	Die Datei, in der die Logins und Passwörter aller Benutzer gespeichert werden, kann nicht erzeugt werden.
100054	Dies ist ein Datenbankfehler.
100055	Dies ist ein Datenbankfehler.
100056	Dies ist ein Datenbankfehler.
100110	Dies ist ein Datenbankfehler.
100111	Die Arbeitsversion konnte nicht angelegt werden, weil es keine voreingestellte Dateinamenserweiterung gibt.
100112	Dies ist ein Datenbankfehler.
100113	Dies ist ein Datenbankfehler.
100114	Dies ist ein Datenbankfehler.
100115	Dies ist ein Datenbankfehler.
100116	Dies ist ein Datenbankfehler.
100119	Dies ist ein interner Systemfehler.
100121	Dies ist ein Datenbankfehler.
100123	Der angegebene Typ entspricht keiner internen Repräsentation eines Tasktyps.
100148	Dies ist ein Datenbankfehler.
100149	Dies ist ein Datenbankfehler.
100150	Dies ist ein Datenbankfehler.
100151	Dies ist ein Datenbankfehler.

100152	Die Applikation konnte auf die angegebene Steuerdatei nicht zugreifen. Entweder existiert die Datei nicht oder der Applikation fehlt das Leserecht.
100153	Dies ist ein Datenbankfehler.
100154	Dies ist ein Datenbankfehler.
100164	Bei der Indizierung durch den Search Engine Server wurde der externe Prozessor nicht gefunden oder die Programmdatei war nicht ausführbar.
100165	Bei der Indizierung durch den Search Engine Server sollten dem externen Prozessor Daten über eine Pipe übergeben werden. Dieser Vorgang ist fehlgeschlagen.
100166	Bei der Indizierung durch den Search Engine Server sollte die Ausgabe des externen Prozessors über eine Pipe gelesen werden. Dieser Vorgang ist fehlgeschlagen.
100167	Dies ist ein Datenbankfehler.
100168	Dieser Fehler kann unterschiedliche Ursachen haben, unter anderem, dass der Pfad nicht existierende Verzeichnisse enthält oder die Zugriffsrechte nicht ausreichen.
100172	Bei der Initialisierung der Verity-Suchmaschine wurden keine Collections gefunden. Diese werden im instanzenspezifischen Verzeichnis *data/collections* gesucht.
100184	Beim SES werden auszuführende Aktionen in die Namen von Requestdateien kodiert. Es gibt nur die beiden Aktionen *Indizieren* und *Löschen*. In Bezug auf eine solche Datei ist es zu einer Inkonsistenz gekommen. Der Fehler wird ins System-Protokoll geschrieben, und die betreffende Datei wird ignoriert.
100185	Dies ist ein Datenbankfehler.
100188	Bei dem Versuch, ein Streaming-Ticket anzulegen, ist ein Fehler aufgetreten. Möglicherweise existiert das Tickets-Verzeichnis nicht oder die Applikation darf nicht darauf zugreifen.
100197	Die Längenbeschränkung ist auf das Dateisystem oder das Betriebssystem zurückzuführen.
100209	Dieser Fehler betrifft die Template Engine. Beim Export sind mehr Fehler aufgetreten, als der Wert des Systemkonfigurationseintrags `acceptableExportFailures` zulässt.
100210	Der Portal Manager oder die Applikation, die diese Meldung ausgegeben hat, ist falsch konfiguriert. Möglicherweise wurde der Portal Manager auch nicht gestartet.
100219	Dies ist ein Datenbankfehler.
109010	Dies ist ein Channel-Check-Callback-Fehler.
109012	Dies ist ein Contentzuweisungscallback-Fehler.
109016	Während `sudo` ausgeführt wurde, traten die aufgeführten Fehler auf.
109017	Während die `generateThumbnail` -Funktion ausgeführt wurde, traten die aufgeführten Fehler auf.
109018	Die Prozedur `usersWhere` in der Datei *usermanAPI.tcl* hat den angegebenen Fehler geliefert.
109021	Die Prozedur `groupsWhere` in der Datei *usermanAPI.tcl* hat den angegebenen Fehler geliefert.
109500	Der Link-Callback kann nicht ausgeführt werden, da die Liste der Felder die falsche Anzahl Elemente enthält.
109501	Der Link-Callback konnte nicht korrekt ausgeführt werden.
109503	Thumbnails müssen im JPEG-Format gespeichert werden.

109511	Dieser Fehler wird ausgegeben und die Fehlerursache wird näher spezifiziert, wenn der Tcl-Code in einer der folgenden Funktionen einen Fehler produziert: Wertzuweisungsfunktion, Wertanzeigefunktion, Vollständigkeitscheck, Versionszuweisungsfunktion, Anlegecheck für Dateien in einem Ordner, Workflowzuweisungsfunktion.
120012	Dieser Fehler kann auftreten, wenn gesicherte Daten gelöscht wurden oder Benutzer während des Dump-Vorgangs auf dem Server gearbeitet haben.
130007	Dies ist ein fataler Datenbankfehler, der dazu führt, dass der Prozess der betreffenden CMS-Applikation terminiert wird. Die betreffende Datenbank-Aktion wurde nicht ausgeführt.
140021	In NPSOBJ-Anweisungen können nur die Alias-Namen von Tcl-Befehlen verwendet werden. Den Namen müssen im entsprechenden Systemkonfigurationseintrag (beispielsweise `tclSystemExecuteCommands` oder `tclFormatterCommands`) die tatsächlichen Prozeduraufrufe zugewiesen werden.