

SAP Knowledge Provider (BC-SRV-KPR)



HELP.BCSRVKPR

Release 4.6C



Copyright

© Copyright 2001 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft[®], WINDOWS[®], NT[®], EXCEL[®], Word[®], PowerPoint[®] and SQL Server[®] are registered trademarks of Microsoft Corporation.

IBM[®], DB2[®], OS/2[®], DB2/6000[®], Parallel Sysplex[®], MVS/ESA[®], RS/6000[®], AIX[®], S/390[®], AS/400[®], OS/390[®], and OS/400[®] are registered trademarks of IBM Corporation.

ORACLE[®] is a registered trademark of ORACLE Corporation.

INFORMIX[®]-OnLine for SAP and Informix[®] Dynamic Server[™] are registered trademarks of Informix Software Incorporated.

UNIX[®], X/Open[®], OSF/1[®], and Motif[®] are registered trademarks of the Open Group.







HTML, DHTML, XML, XHTML are trademarks or registered trademarks of W3C[®], World Wide Web Consortium, Massachusetts Institute of Technology.

JAVA[®] is a registered trademark of Sun Microsystems, Inc.

JAVASCRIPT[®] is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, SAP Logo, R/2, RIVA, R/3, ABAP, SAP ArchiveLink, SAP Business Workflow, WebFlow, SAP EarlyWatch, BAPI, SAPPHIRE, Management Cockpit, mySAP.com Logo and mySAP.com are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other products mentioned are trademarks or registered trademarks of their respective companies.

Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax
	Tip

Contents

SAP Knowledge Provider (BC-SRV-KPR)	6
Document Management with KPro.....	10
Document Management Service (BC-SRV-KPR).....	14
Concepts	15
Attributes	17
Relationships.....	19
Versioning	21
Context Resolution.....	24
Content Models	26
DMS Programming Interface	28
Content Management Service (BC-SRV-KPR)	31
Concepts	33
SAP Content Server	35
KPro and Caching	36
Customizing for Caching.....	37
Cache Server	38
Content and Cache Server Administration	39
Selecting a Server.....	40
Functions.....	42
Informational Overview	43
Detailed Information.....	45
Certificates	46
Settings	47
Statistics.....	48
Creating New Content Repositories	49
Content and Cache Server Monitoring	50
Monitoring for Content Servers	52
Monitoring for Cache Servers	54
Monitoring for KPro Web Servers	56
SAP Content Server HTTP 4.5 Interface	58
Introduction.....	59
Definition of Terms.....	60
Implementation	62
Security	63
secKey	64
Protection / Right of Access	66
Syntax	68
General	69
Coding in the URL.....	70
Coding in the Response Body	72
Functions.....	73
Access Functions.....	75
info	76
get.....	81

docGet.....	83
create.....	87
HTTP-PUT.....	89
HTTP-POST multipart/form-data.....	90
mCreate.....	92
append.....	95
update.....	97
HTTP-PUT.....	99
HTTP-POST multipart/form-data.....	100
delete.....	101
search.....	103
attrSearch.....	106
Administration Functions.....	112
putCert.....	113
serverInfo.....	115
Error Codes.....	118
Appendix.....	119
Parameters and Key Words.....	120
Information on Migrating Existing Archives.....	125
Index Management Service (BC-SRV-KPR).....	128

SAP Knowledge Provider (BC-SRV-KPR)

Purpose

The SAP Knowledge Provider (KPro) is a cross-application and media-neutral information technology infrastructure within the R/3 Basis System. The modular structure and openness on which KPro is based is reflected in its modular services and clearly defined interfaces. Its extensive flexibility means that KPro can be used to process the widest variety of information types within and relating to documents and document-like objects. For example, administration and index data, as well as pure content.

Applications that use the SAP Knowledge Provider involve various end users, who in turn have different requirements. There is therefore no universal interface for accessing KPro services regarding the following points:

- Specific knowledge management functions for end users
- Specific terminology for describing the document-like objects within the context of the individual application
- Specific design of work process flows
- Specific design of user interfaces

Consequently, many different applications can use KPro services by using their own user interfaces. The cross-application exchange of information via the KPro infrastructure on which the interfaces are based is ensured.

KPro is targeting the following application areas (the SAP applications that have already integrated KPro are listed in the section "KPro Client Applications"):

- Context-specific development and distribution of help texts and training material from SAP itself, as well as from SAP customers, within the context of the SAP Knowledge Warehouse
- Management of multi-media information objects for enterprises primarily engaged in the media field, for example, the collection and distribution of information using all possible media and distribution channels, such as print media, the Internet, television and radio
- Development, management and distribution of software and its components
- Administration of report lists in the Business Information Warehouse environment
- Connection of business transactions with additional static information objects for e-commerce solutions
- Extensive support for document-based business processes using SAP Business Workflow
- Publishing solutions for both the Internet and intranet. For example, international enterprises can obtain up to date information at any time using KPro about important issues within the enterprise, for example quality management.

Even though there is likely to be an overlap in the requirements of end users in various areas, significant differences regarding the general aims and requirements of the KPro client applications exist. For example, constructing an intranet is quite different from publishing a newspaper or administrating software components. In spite of these outward differences, what is required of the infrastructure is the same, such as Versioning, context resolution and the

integration of various content servers and search engines. In order to meet these requirements, KPro provides a media-neutral and cross-application general infrastructure.

Implementation Considerations

In order to use KPro, you may, depending on the SAP application, need to make some Customizing settings: In the implementation guide (IMG), choose *Basis* → *Basis Services* → *SAP Knowledge Provider*. Further explanation of the individual activities can be found in the IMG modules.

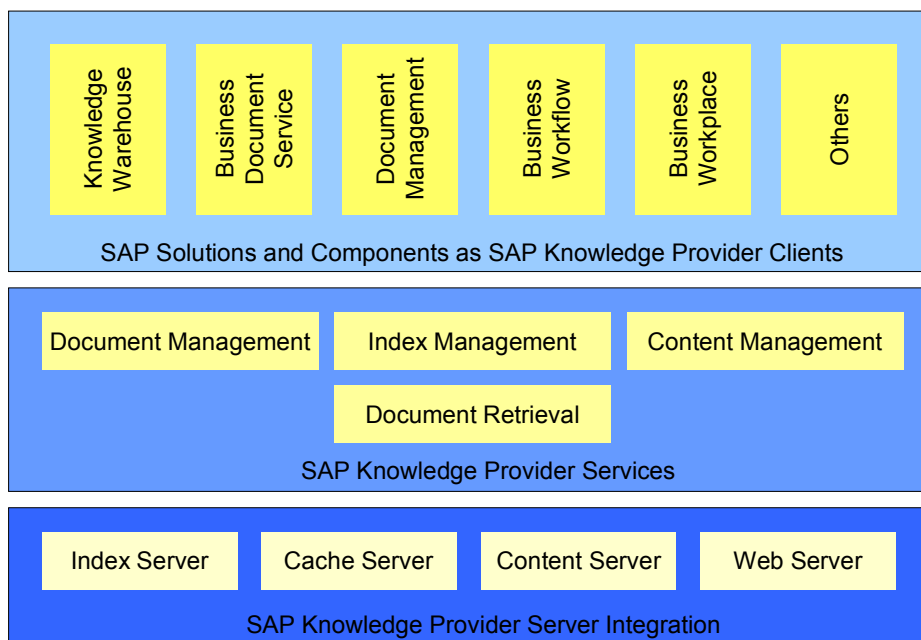
The role of KPro Administrator (SAP_BC_SRV_KPR_ADMIN) is provided for the SAP Knowledge Provider.

Features

The SAP Knowledge Provider provides the following services within its information technology infrastructure:

- Integration of KPro client applications
- Integration of KPro services
- Integration of KPro servers

SAP Knowledge Provider Services und Clients



KPro Client Applications

Client applications can use KPro services by integrating the relevant interfaces. You can select various services in accordance with flexibility and the scope of the required document management functions, for example, the Content Server HTTP interface for integrating storage systems, the IMS interface for integrating search and retrieval machines.

SAP Knowledge Provider (BC-SRV-KPR)

The following R/3 applications use KPro functions:

- SAP Knowledge Warehouse
- Business Document Service (BDS)
see documentation for [BC - Business Document Service \[Ext.\]](#)
- Document Management System (DMS)
see documentation for [Document Management \[Ext.\]](#)
- SAP Business Workflow
see documentation for [SAP Business Workflow \[Ext.\]](#)
- SAP Business Workplace
see documentation for [BC - Business Workplace \[Ext.\]](#)

KPro Services

The SAP Knowledge Provider provides the following services:

- Document Management Service (DMS)
The DMS enables you to edit documents and document-like objects using application-specific content models. The DMS can be used to modify administration data for documents and store document content in content servers.
See [Document Management Service \[Page 14\]](#)
- Document Retrieval Service
The Retrieval Service enables you to use all sorts of different retrieval strategies, alone or in combination with each other.
- Index Management Service
The IMS enables different index servers (search engines) to be integrated according to particular applications and tasks, thereby guaranteeing index server independence.
See [Index Management Service \[Page 128\]](#)
- Content Management Service
See [Content Management Service \[Page 31\]](#)

KPro Server Integration

- Index Servers
Allows search engines to be integrated, see *SAP Knowledge Provider Server Infrastructure Components* installation guide
- Content Servers
Allows the SAP DB or external content servers to be integrated, see [SAP Content Server \[Page 35\]](#)
- Cache Servers
Allows document caching, see [Cache Server \[Page 38\]](#)
- Web Servers

SAP Knowledge Provider (BC-SRV-KPR)

Allows a server to be integrated to display info objects as part of the SAP Knowledge Warehouse, see *SAP Knowledge Provider Server Infrastructure Components* installation guide.

Document Management with KPro

Introduction

The information and knowledge contained in documents can have a lasting effect on the efficiency and productivity within enterprises.

As a general and comprehensive infrastructure, the SAP Knowledge Provider (KPro) provides cross-application and media-neutral document management techniques. It is a component of the R/3 Basis System and can be integrated by all SAP applications.

Concepts

The Different Information Entities

The SAP Knowledge Provider distinguished between the following information entities in relation to documents:

- Content
- Content index
- Administration data
- Model data



A text file presenting the content exists for this unit. The file exists irrespective of whether further information is available about this unit. The author, creation date and the name of the relevant documentation are examples of administration data. The model data contains the document attributes amongst other things. So the fact that a document has the attribute “author” is part of the model data. The primary role of the content index is to be a secondary representation of the content, thereby enabling searches based on content (full text search, for example). This distinction need not always be made in full. For example, it may be advantageous to store part of the administrative data (the author, for example) in the content index as well, or in the content (self-describing documents).

Documents and Document-Like Objects

The term “document” is very broad in the KPro context: Texts, images, video and audio tapes, and also programs, Web pages, controls and so on, are termed “documents” in KPro. This means that the term ‘document’ in KPro extends beyond its classical meaning and encompasses not only those entities usually known as documents, but also all other possible self-contained document-like objects.

Application-Specific Content Models

There is no fixed definition of the term ‘document’ in KPro. Instead, client applications can create their own, specific definitions for their document models. This ensures that the content models are application-specific. Such a definition encompasses the following in particular:

- Classes for relationships
- Classes for logical documents

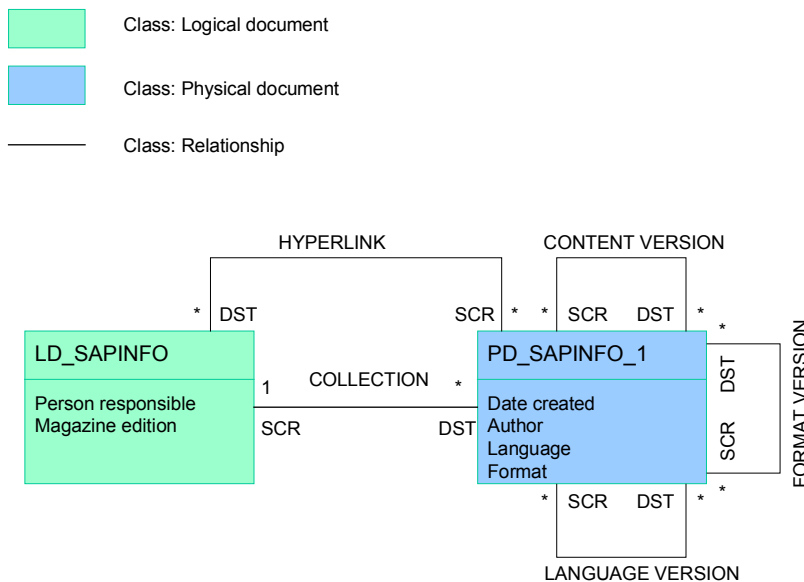
- Classes for physical documents

Content models are the structure and architecture on which the editing of document-like objects within an application-specific context is based. This means that each client application can use their own application-specific meta data description, without having to skimp on exchanging information with other applications.



Application-specific content model for a magazine article:

In this example, there is a logical document class and a physical document class, each with certain properties. There are also various classes for relationships. The relationship class instances CONTENT VERSION, FORMAT VERSION and LANGUAGE VERSION enable you to manage as many different versions as you wish according to various criteria (content changed, translation, format conversion).



Application-specific content model

Intension and Extension

KPro provides functions for simplifying and automating document management in many areas. The division of document management into two levels (intension and extension) is of central importance here.

The intension represents the logical properties of a document, in other words, its sense, purpose and intention.

The extension represents the actual document in concrete terms.

The intension and extension are represented in the SAP Knowledge Provider by different administration objects: logical document and physical document respectively.

Document Management with KPro



This part of the documentation introduces the intension, the conceptual background of the SAP Knowledge Provider, and is purely abstract. This unit is available in both German and English. It is in electronic form (HTML help and plain HTML, for example), but can also be printed. This means that the concrete representations exist in various forms, that is, extensions.

Relationships

Isolated documents are worthless. Document-like objects always have relationships with other objects. These relationships may be of an implicit or explicit nature. You can use KPro to manage not only the document-like objects themselves, but also the relationships between these objects. You can also define relationship classes whose instances explicitly express the relationships between documents.

The relationships and references between documents are dynamic. If a document is changed in KPro, this does not lead to a costly chain reaction of manual changes in other documents. You do not have to worry about such changes, nor the effects of changes, on other documents.

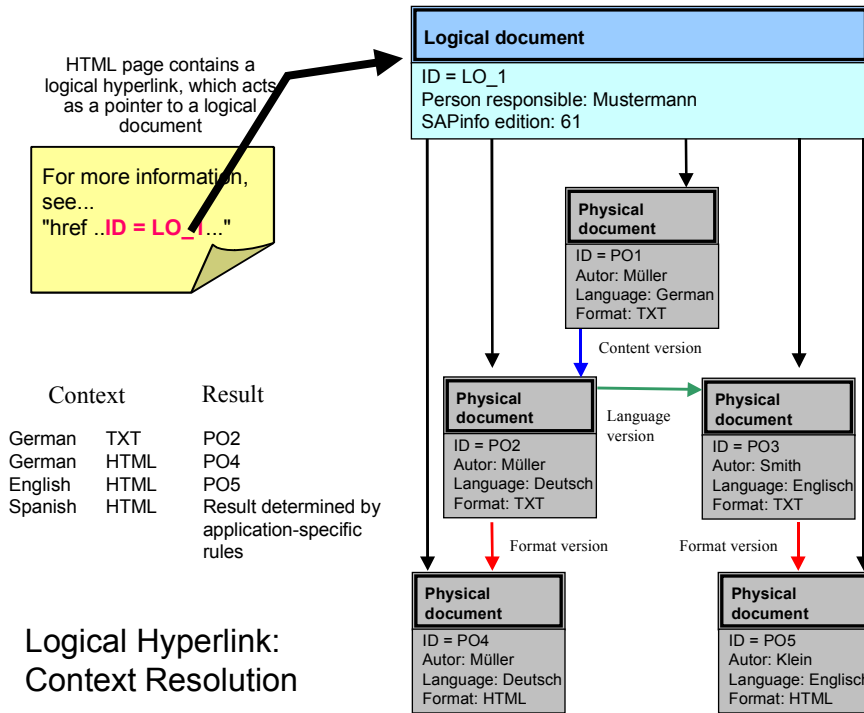


If there is a note stored that relates to a document, this connection between the note and the document can be expressed by a relationship. In the same way, hyperlinks in documents can be expressed by relationships, in order that the dependency between documents linked by hyperlinks, be made transparent.

Late Binding and Context Resolution

If there is a large number of relationships between various documents (hyperlinks, for example), the documents are not longer isolated from one another. This problem applies to all types of relationships, both between different documents and between documents and business objects, and is not restricted to hyperlinks. SAP Knowledge Provider offers a solution in the referencing of logical documents (logical hyperlink, for example) as well as a context resolution process at runtime. This means that the version that best matches the specified context is not selected from the set of physical documents belonging to the specified logical document until runtime. It is selected by considering contextual information (user context or business context). Late binding is therefore an interface between the intension and the extension. During the resolution process, there can be an inflow of both information managed by the SAP Knowledge Provider and information from the SAP Knowledge Provider client application. If documents are translated or converted into a different format, or their contents changed, the relationship network remains intact. This avoids costly and time-consuming extra work.

Document Management with KPro



Versioning

KPro provides an efficient versioning concept, which supports multiple versioning based on various criteria. There may be different language or format versions, for example. A document may exist in German, English and French and in the format PDF or HTML.

Equal Access and Retrieval

KPro allows equal access to all document-like objects via HTTP, irrespective of the storage medium used and the application-specific design of the Web site. Check-in and check-out mechanisms ensure that authors do not make conflicting changes to the same object. Numerous retrieval strategies are supported, one of these being the integration of search engines for full text search.

Storage Media for Content

KPro provides services for document storage. This enables SAP-based document management functions to be combined with a wide range of storage media.

See also [Content Management Service \[Page 31\]](#).

Document Management Service (BC-SRV-KPR)

Purpose

The Document Management Service (DMS) is a subcomponent of the information technology infrastructure provided by the SAP Knowledge Provider as part of the R/3 Basis system. A central element of the DMS is the processing of document-like objects on the basis of application-specific content models. It is only by using the DMS that you can change the administrative data for documents and store it in a storage medium (R/3 database, for example).

Implementation Considerations

Before you use the DMS, you need to make some Customizing settings: In the Implementation Guide (IMG), choose *Basis* → *Basis Services* → *SAP Knowledge Provider* → *Document Management Service*. The individual activities are explained in the IMG modules.

Features

The following functions are available in the DMS:

- Management of [attributes \[Page 17\]](#)
- Management of [relationships \[Page 19\]](#)
- [Versioning \[Page 21\]](#)
- [Context resolution \[Page 24\]](#)
- [Content models \[Page 26\]](#)

A central purpose of the DMS is to **protect** the SAP application from the peculiarities of storing administrative data in tables. The DMS deals with the management and editing of all administration data and the corresponding tables.

It also provides a **lock mechanism**. This ensures that no more than one author can work on the same document at the same time.

A [programming interface \[Page 28\]](#) is also available in the DMS.

Concepts

In the context of the SAP Knowledge Provider, the term "document" is used colloquially to describe all information entities accessible via document-like objects. In particular, these are:

- The actual content
- Administrative data
- Index data
- Meta data

Each application-specific [content model \[Page 26\]](#) represents a document definition.



Logical document "Documentation unit on SAP Knowledge Provider data structures".

Logical documents, physical documents, and components are central components of the SAP Knowledge Provider. A logical document represents a logical compound of a document in all its forms. Each logical document can therefore contain several physical documents. Physical documents consist of meta descriptions of the actual existing documents. Attributes are used to form these meta descriptions.



Physical documents for the logical document "Documentation unit on SAP Knowledge Provider data structures":

"Documentation unit on SAP Knowledge Provider data structures in HTML format, in English, third version, 1.10.1999" and "Documentation unit on SAP Knowledge Provider data structures in WinWord format, in German, second version, 2.10.1999".

References to one or more components exist in physical documents. Components contain administrative data on the actual content, the size of the file, for example.

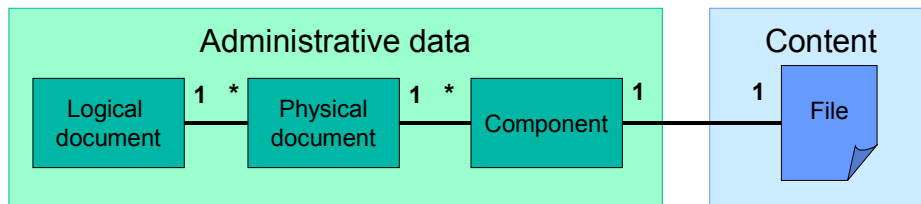


Components for the physical document "Documentation unit on SAP Knowledge Provider data structures in HTML format, in English, third version, 1.10.1999", where this physical document consists of a HTML file and a PowerPoint file on the content server:

administrative counterpart of the HTML file and administrative counterpart of the PowerPoint file on the content server.

On the physical level, **one** file, which contains the actual content, is assigned to each component.

Concepts



For the necessary Customizing steps, see the Implementation Guide (IMG) under *Basis* → *Basis Services* → *SAP Knowledge Provider* → *Document Management Service*.

Attributes

Attributes identify meta descriptions of physical documents. Examples of an attribute are format or language.

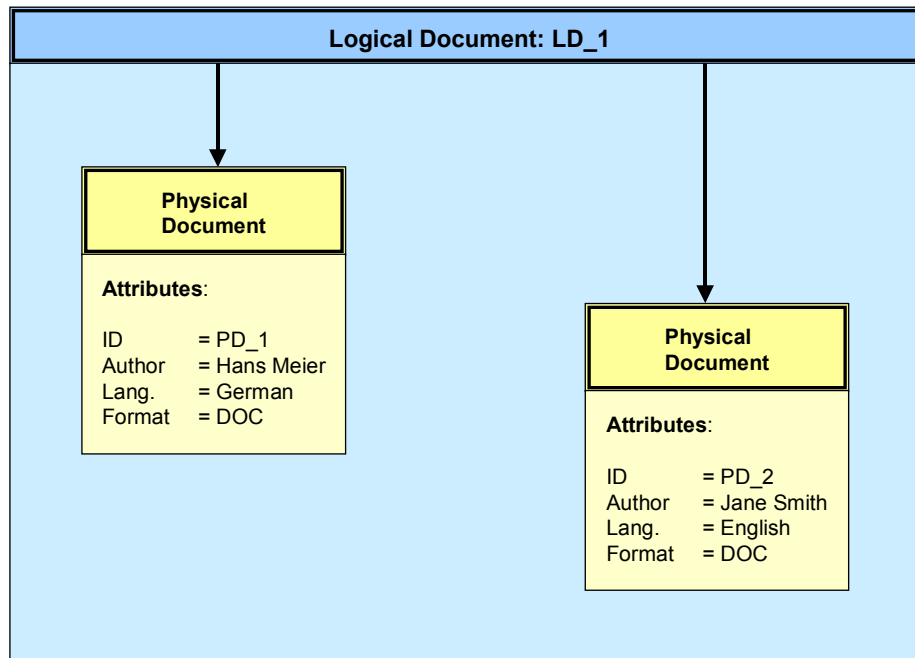
Attributes can be assigned by either the SAP application in which the documents were created or by the SAP Knowledge Provider itself. The following predefined attributes are supplied by the SAP Knowledge Provider:

Attribute name	Description	Relevant for:		
		Logical documents	Phys. docs	Relation-ships
CREATED_AT	Creation time	x	x	x
CREATED_BY	Created by	x	x	x
LAST_CHANGED_AT	Time of last change	x	x	x
LAST_CHANGED_BY	Last changed by	x	x	x
EXPIRYTIME	Expiration time	x	x	x
ORIGINAL_LANGUAGE	Original language	x		
CHECKED_OUT	Flag for documents that have been checked out		x	
CHECKOUT_USER	Name of the user who checked out the document		x	
DOCUMENT_FORMAT	Document format		x	
LANGUAGE	Language		x	
RESERVED	Flag for resubmitting a document		x	
STATE	Status		x	
STORAGE_CATEGORY	ID of the storage category		x	
DESCRIPTION	Brief description	x	x	

Attributes



The logical document with the ID LD_1 comprises 2 physical documents with the IDs PD_1 and PD_2. The first physical document is identified by the following attributes: ID = PD_1, author = Hans Meier, Language = German and format = DOC. The second physical document is identified by the following attributes: ID = PD_2, author = Jane Smith, language = English and format = DOC.



Relationships

The SAP Knowledge Provider uses relationships to represent dependencies between documents. An internal relationship links exactly two KPro administrative documents according to certain criteria. These criteria are used to group relationships into classes. Each relationship is a separate object. In addition to internal relationships, there are also relationships that link KPro administrative objects with documents that are not managed in the KPro. These are referred to as external relationships.



When a document managed as a physical document in the KPro is translated from German into English, a language relationship exists between the two documents that allows the origin of the English document to be traced. The criterion for building the relationship in this case is the "language". Creating and managing language relationships of this kind is referred to as language versioning.

The SAP Knowledge Provider provides the following predefined types of relationships or relationship classes:

- **Collection relationship**

The collection relationship (LOGOBJECT) models the relationship between several physical documents and one logical document. A certain physical document can belong to one single logical document. This relationship allows the logical document to act as a container for physical documents.
- **Template relationship**

A template relationship (EXPORT_MODEL) simply comprises two physical documents. In this way, a certain document can act as a template for creating further documents. This relationship is used when new documents are created.
- **Version relationship**

A version relationship (VERSIONREF, FORMATREF, TRANSLREF) between document A and document B, for example, indicates that document A was created as a version of document B. Document A can be:

 - A translation
 - An update of the content of document B
 - A backup in a format that is different to document B
- **Description relationship**

A description relationship (DESCOBJECT) indicates that an information object represents a description of a different information object. This can be useful, for example, if the content stored in the KPro is an image.
- **Structure relationship**

Relationships

A structure relationship (STRUCTLINK) models a hierarchical relation between the content of documents. This can be the relationships between chapters and paragraphs within a book, for example.

- Hyperlink relationship

Hyperlink relationships (HYPERLINK) are used to represent hyperlinks between documents. If a hyperlink points to a logical document, the context resolution functionality in the KPro is used.

Versioning

The term "versioning" in the KPro is an umbrella term for different versioning subtypes that are mapped by different relationship classes in versioning. Versioning can be regarded as mapping a derivation relationship between two documents. The relationship is directed from document 2 to document 1, whereby document 2 originated from document 1 on the basis of a specific criterion.

The following versioning relationship classes are predefined in the KPro:

- Content versioning
Criterion: content has changed
- Format versioning
Criterion: format has changed
- Language versioning
Criterion: language has changed

Multiple Versioning

Independent of these criteria, the KPro supports **multiple versioning**. The term "multiple versioning" is used to refer to situations where several additional physical documents are derived from one physical document. The derived documents can differ with regard to their format, language, and so on.



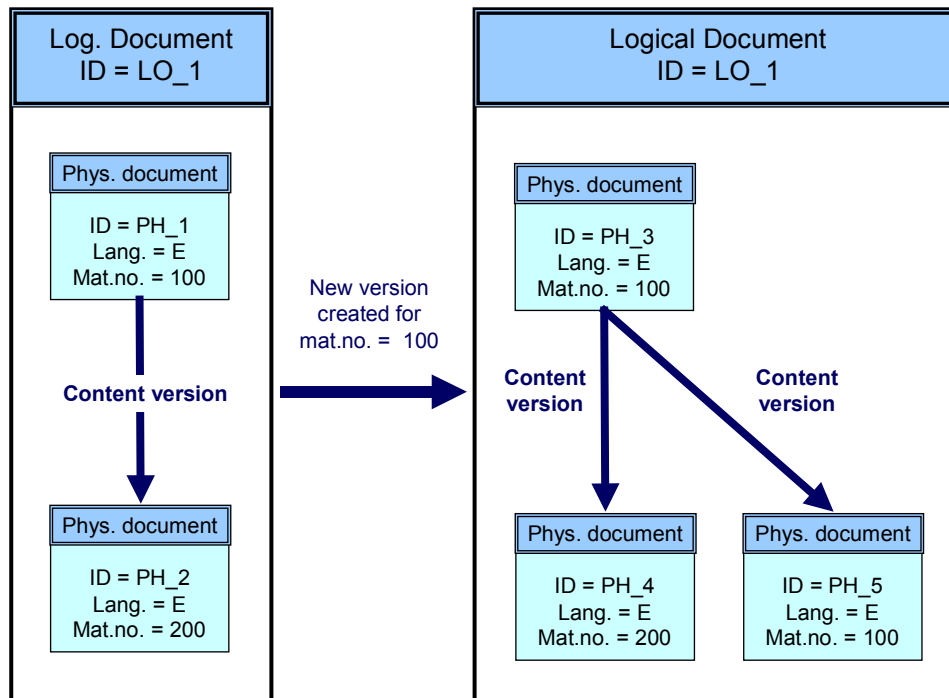
Three different content versions are derived from one physical document. These three versions are equal children of one physical document.

If a German document is translated into English and Spanish, this is also a case of multiple versioning.

The graphic below illustrates the principle of multiple versioning using content versioning as a versioning criterion.

Versioning

Multiple Versioning



The logical document on the left contains two manuals for washing machines in English - one for model 100 and one for model 200. The manual for model 200 originates from the model that was originally written for model 100. Since model 100 has a technical defect, the manual must be modified. In order to enable the name of the person who changed the manual and the date of the change to be traced (for example, as part of the quality assurance measures), a new content version is created without overwriting the original version. In this way, a new physical document is created for model 100 and is shown on the right hand side. As a result, two content versions now exist that were derived from the content originally written for model 100.

Strict Versioning

The **strict versioning** principle protects the content of the components of physical documents from being changed. Every change automatically results in a new version with a new ID. This can lead to an excessive number of versions, particularly in applications where a large number of documents are processed. If you use references to physical documents, you will have to update each individual reference as soon as a new version is created or an existing one deleted. This highly complex procedure, however, is not necessary thanks to the three-level content model (see also [Content Models \[Page 26\]](#)). The three-level content model supports both strict versioning and references between documents: all references refer to logical documents. This means that it is irrelevant whether new versions with new IDs are created for the individual physical documents. Therefore, there is no unnecessary version overhead.

Versioning Types

Content versioning

If a new physical document is created when the contents of the components of an existing physical document are changed, content versioning stores and manages this information in the KPro. Content relationships are used to build content versioning.

Format versioning

The purpose of format versioning is to manage physical documents that differ solely with regard to their technical format. Format versioning uses format relationships for this purpose. Physical documents that are part of format versioning must belong to the same logical document.



When a DOC is converted to a PDF, this dependency is stored as a format relationship in the KPro. Format versioning manages this relationship and allows you to check at any time which DOC was used to create the PDF. In this way, you can answer the question "Is there a format for this physical information object that is suitable for my application?".

Language versioning

Management of language relationships is referred to as language versioning.

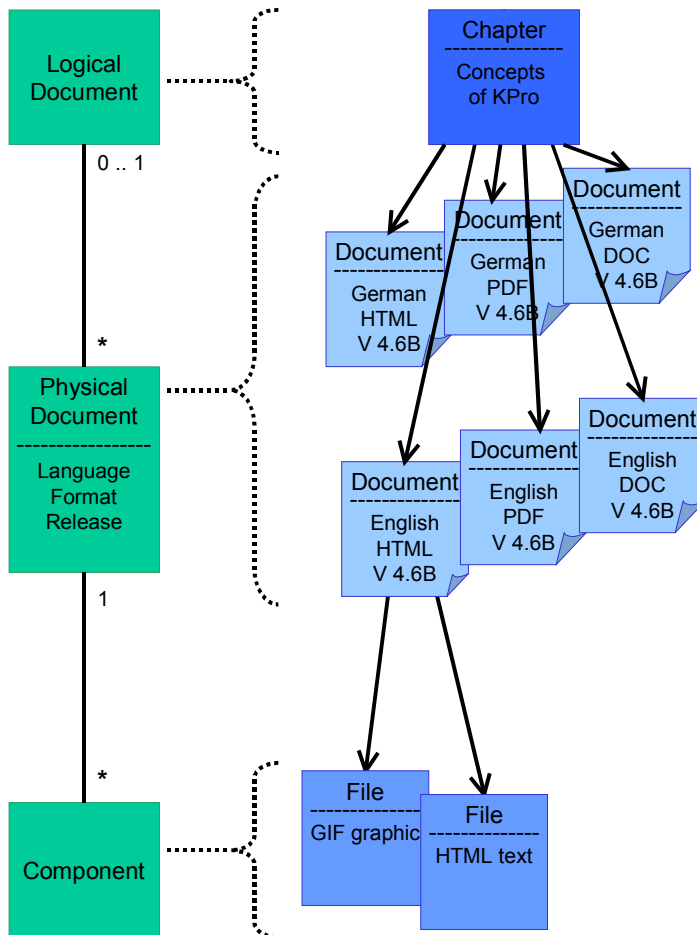


A user wants to translate a document managed in the KPro from German into English. Using the KPro interface, the user can transfer a copy of the German original to his or her application. A language relationship is then created between the copy and the original. Once the user has completed the translation, the translated version is transferred to the KPro using the check-in functionality. The purpose of language versioning is to manage the created language relationship and answer questions such as "Is there a translation of the physical document 4711 in German, English, or French, and so on?".

Context Resolution

Context Resolution

The KPro supports the concept of logical hyperlinks to logical documents. The most important part of a logical hyperlink to a logical document is the ID of the logical document. At runtime, a context string (attribute name1 = attribute value1, attribute name2 = attribute value2, ...) is added dynamically to resolve the logical hyperlink and determine the physical document and its components and content. This process is referred to as context resolution.



A component represents exactly one file on the Content Server. In this way, a document can comprise several files, as is often the case with HTML documents (HTML documents often contain GIF files).

Physical documents, on the other hand, represent individual documents and everything that is associated with them (that is, their components). Since physical documents also contain the properties of documents that represent a specific version, they are evaluated to resolve the context.

Logical documents represent all of the documents that belong to a collection. In other words, they also contain all the physical documents. Logical documents do not contain information that is relevant for resolving the context and, therefore, are not subjected to the versioning process. Logical documents represent a set of those physical documents that logically belong to a group.

Context Resolution

The group definition and its properties depend on the respective client application. References always point to logical documents, which means that their independence of the lifecycle of the individual documents is guaranteed.



Several current versions of a document can exist at the same time (see also: [Versioning \[Page 21\]](#)). A user, who is logged onto an SAP System in English and has installed a Web browser for the KPro at the frontend, wants to display the English HTML version of the documentation "KPro Concepts". A German user logged on in German, on the other hand, wants to display the German version of this documentation. Both users also expect to see the documentation for the current release. The KPro then compares the attributes of the user context with the relevant document properties:

Attributes of the User Context	Document Properties
Logon language	Language
Viewer application	Format
Current release of the SAP System	SAP System release defined for the document

If a document exists in the collection whose properties match all of the context attributes, it is displayed by the KPro. Otherwise, the document that matches the most attributes is displayed.

A client application of the KPro can also define context attributes and document properties (see also: [Attributes \[Page 17\]](#)).

If several documents with the same number of matching properties exist, the application-specific context resolution function is activated in the form of application-specific function modules. These function modules evaluate the additional attributes in order to return a suitable document or they prioritize properties in the context of the application.

Content Models

Content Models

The individual SAP applications create meta data models for the various types of documents that they use. Relationships exist between these documents. These models are called content models.

A content model contains a description of the characteristics of the individual content entities in KPro, at an abstract level. The different types of content are represented on the documents in the content model and managed. End user documentation and scanned documents are possible types of content.

KPro Content Model

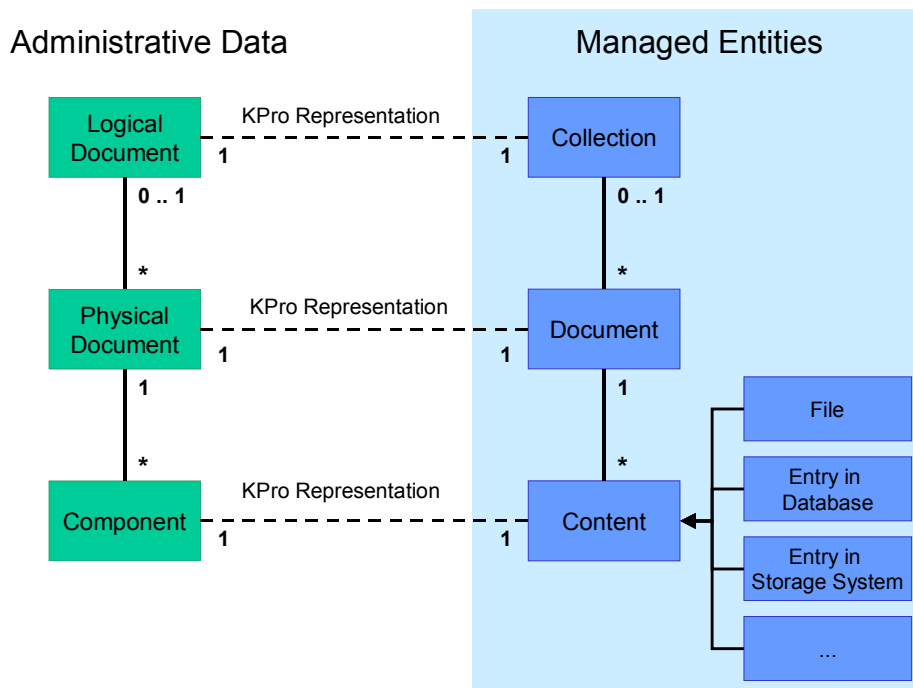
The KPro content model contains three levels for administrative data. The KPro content model is therefore also called a **three-level content model**:

- Components manage content
- Physical documents manage documents
- Logical documents manage collections of documents

Several physical documents are assigned to a logical document. A physical document represents an individual document, while a logical document represents a collection of documents.

A document can consist of various files, meaning that one or more components can exist for each physical document. Each component, in turn, is associated with **one** content unit

KPro Content Model





KPro also allows two-level content models. In this case, an application using KPro does not need any logical documents. Only components and physical documents are used.

Application-Specific Content Model

In order to use the KPro document infrastructure, applications can develop their own content models. KPro accesses these application specific content models at runtime and uses the meta information they contain to execute the required services. Applications can therefore define their own attributes or certain types of versioning, for example.

Firstly, you decide whether to use a two or three-level content model. If it is possible that at some point, the documents will be changed or other versions of the documents will be created, the three-level content model is used. Possible changes are changes to the content, the language and the format. The two-level content model is only intended for documents that are not to be changed and for which no versions are to be created.

The three-level content model is also recommended for scenarios where more than one context-dependent instance of a particular physical document exists simultaneously.

As soon as you have decided whether to use the two or three-level content model, the content model itself can be made by following the following steps:

- Define the application-specific attributes
- Define classes for physical documents
- Define classes for logical documents
- Define classes of relationships
- Define the permitted relationships

In the last step the rules for creating relationships of certain classes between instances of logical and physical documents are defined. KPro accesses this information at runtime within the application-specific content model, to determine whether the required activity is permitted. These rules can be used to catch unpermitted relationships between content versions of physical documents, for example.



You can create your own content model using the **Document Modeling Workbench** (transaction DMWB).

DMS Programming Interface

Use

You use this programming interface to integrate the functions of the Document Management Service into your SAP application.

Features

The following function modules are available:

- Function modules for document classes and relationship classes
 - SDOK_CLASS_ATTRIBUTES_GET
 - SDOK_CLASS_FIND_BY_QUERY
 - SDOK_CLASS_PARTNERS_GET
 - SDOK_CLASS_PROPERTIES_DELETE
 - SDOK_CLASS_PROPERTIES_GET
 - SDOK_CLASS_PROPERTIES_SET
- Function modules for physical documents
 - SDOK_PHIO_CREATE
 - SDOK_PHIO_CREATE_AS_COPY
 - SDOK_PHIO_CREATE_NEW_VERSION
 - SDOK_PHIO_CREATE_WITH_MODEL
 - SDOK_PHIO_DELETE
 - SDOK_PHIO_ATTRIBUTES_GET
 - SDOK_PHIO_PROPERTIES_SET
 - SDOK_PHIO_PROPERTIES_GET
 - SDOK_PHIO_PROPERTIES_DELETE
 - SDOK_PHIO_FROM_RELATIONS_GET
 - SDOK_PHIO_TO_RELATIONS_GET
 - SDOK_PHIO_FIND_BY_QUERY
 - SDOK_PHIO_STORE_CONTENT
 - SDOK_PHIO_STORE_AS_UPDATE
 - SDOK_PHIO_STORE_NEW_VERSION
 - SDOK_PHIO_STORE_NEW_CONTENT
 - SDOK_PHIO_LOAD_CONTENT
 - SDOK_PHIOS_CHECKIN

- SDOK_PHIOS_CHECKIN_AS_UPDATE
- SDOK_PHIOS_CHECKIN_NEW_VERSION
- SDOK_PHIOS_CHECKOUT
- SDOK_PHIOS_CHECKOUT_CANCEL
- SDOK_PHIOS_GET_COPY
- SDOK_PHIOS_FILE_PROPERTIES_GET
- SDOK_PHIO_GET_URL_FOR_GET
- SDOK_PHIO_GET_URL_FOR_UPDATE
- SDOK_PHIO_GET_URL_FOR_NEW_VERS
- SDOK_PHIO_GET_URL_NEW_CONTENT
- SDOK_PHIO_GET_URL_FOR_PUT
- Function modules for logical documents
 - SDOK_LOIO_CREATE
 - SDOK_LOIO_CREATE_WITH_PHIO
 - SDOK_LOIO_DELETE
 - SDOK_LOIO_ATTRIBUTES_GET
 - SDOK_LOIO_PROPERTIES_SET
 - SDOK_LOIO_PROPERTIES_GET
 - SDOK_LOIO_PROPERTIES_DELETE
 - SDOK_LOIO_DESCRIPTIONS_SET
 - SDOK_LOIO_DESCRIPTIONS_GET
 - SDOK_LOIO_DESCRIPTIONS_DELETE
 - SDOK_LOIO_FROM_RELATIONS_GET
 - SDOK_LOIO_TO_RELATIONS_GET
 - SDOK_LOIO_FIND_BY_QUERY
 - SDOK_LOIO_PHYSICAL_OBJECT_GET
- Function modules for relationships
 - SDOK_RELA_CREATE
 - SDOK_RELA_DELETE
 - SDOK_RELA_PROPERTIES_SET
 - SDOK_RELA_PROPERTIES_GET
 - SDOK_RELA_PROPERTIES_DELETE
- Other function modules

DMS Programming Interface

- SDOK_ATTRS_QUALITIES_GET
- SDOK_DOCUSPACE_SEARCH



The function modules listed here are just some of the more important DMS function modules.

Content Management Service (BC-SRV-KPR)

Purpose

The various storage media that can be used to store document-like objects can be distinguished by their different qualitative advantages: with regard to security, cost saving or performance, for example

The Content Management Service (CMS) is a sub-component of the information technological infrastructure provided by the SAP Knowledge Provider within the R/3 Basis System. A central feature of the CMS is that it is aimed at all different types of storage media. It, therefore, forms the interface between content servers and the R/3 System.

The CMS also provides document caching via integrated cache servers.

Implementation Considerations

Before using the CMS, you need to make some Customizing settings:

In the Implementation Guide (IMG), choose *Basis* → *Basis Services* → *SAP Knowledge Provider* → *Content Management Service*. Explanations of the individual activities can be found in the IMG modules.

Features

The CMS provides the following services:

- Integration of various external storage media:
 - R/3 database
 - External content servers

The static content may be stored either in the SAP DB or in an external content server (for which the CMS is the appropriate interface). You can integrate various types of content servers.

- Retrieving the HTTP API

The first version of the HTTP interface is available as of release 4.5. A certification program has been started. All KPro client applications can store documents via HTTP in an external content server and retrieve them where necessary.



The use of HTTP offers numerous advantages in view of its universal and uncomplicated infrastructure, which is largely independent from the content servers and clients used. The HTTP Content Server interface is designed such that the characteristics of the concrete storage medium are fully transparent to the R/3 System and the clients (see [SAP HTTP Content Server 4.5 Interface \[Page 58\]](#)).

- Administration of the SAP Content Server and administration of SAP Cache Servers

Administration can be performed directly from the R/3 System. SAP DB-specific tools are available for monitoring and administrating the SAP DB.

For more information, see the section on [SAP Content Server Administration \[Page 39\]](#).

Content Management Service (BC-SRV-KPR)

- Monitoring support as part of the CCMS
For more information, see the section on [Content and Cache Server Monitoring \[Page 50\]](#).
- Integration into the Change & Transport System
There is a connection to the SAP transport and correction systems.
- SAP Content Server
SAP delivers a standard content server that can be installed separately on an NT server. (see [SAP Content Server \[Page 35\]](#)).
- Cache Server
Any number of cache servers can be set up at different locations. Contents are transferred directly between the client and content server. If the content servers are accessed from different locations that are only connected by a WAN, cache servers should be used. Network traffic across the WAN can be minimized and performance enhanced considerably by installing at least one cache server at each location.
A client cache is also available at the user's frontend.
For further information, please refer to the section on [KPro and Caching \[Page 36\]](#).
- Protection of the further KPro services from the peculiarities of the different types of storage systems
- Protection of the KPro client applications from the peculiarities of the different types of storage systems

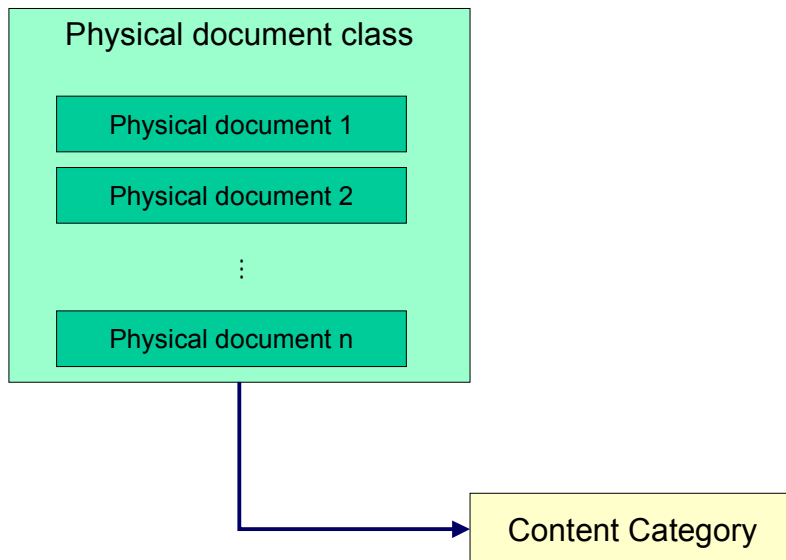
Application Areas

The CMS is implemented within KPro, to shield both the R/3 applications that use KPro document management functions and KPro itself from the peculiarities of the different existing types of storage media. This ensures the independence of the content server (that is, the storage medium) used.

Concepts

In accordance with the fundamental DMS concepts of logical documents, physical documents and components (see also [DMS concepts \[Page 15\]](#)), several physical documents can be grouped into physical document classes. Each physical document class is assigned to a default content category:

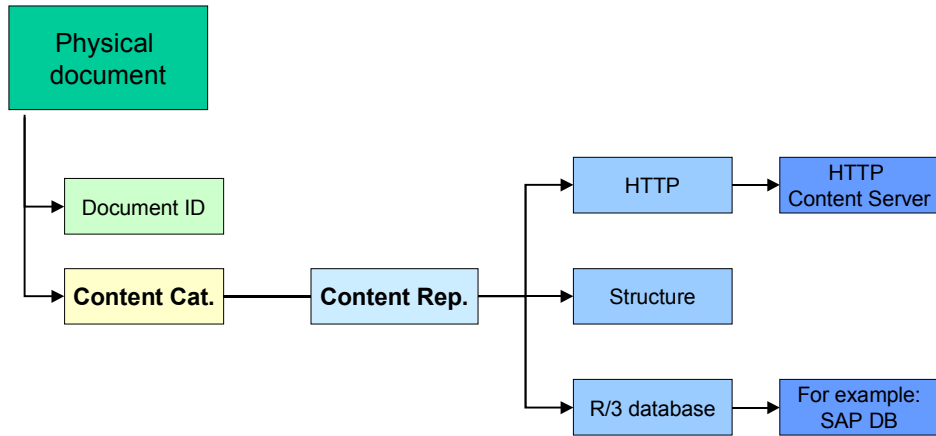
Physical Document Class and Content Category



A physical document is identified by the document ID and a content category. The content category exists only at a purely logical level. Each content category is in turn assigned to a content repository, where the content repository is the physical counterpart of the content category. The content repositories enable physical storage. Various storage types are possible, for example, HTTP or R/3 database. The storage types, in turn, are assigned to concrete content servers, for example, an HTTP content server for the storage type HTTP.

Concepts

Content Category and Content Repository



The necessary Customizing steps can be found in the Implementation Guide (IMG) under *Basis* → *Basis Services* → *SAP Knowledge Provider* → *Content Management Service*.

SAP Content Server

Purpose

The SAP Content Server is based on the SAP DB and is available as of Release 4.6, on NT.

As well as the R/3 database, there is also always an external content server available in each R/3 installation. This means that the required technical infrastructure is always available for all document-based applications and business scenarios that do not require long-term archiving. Since the SAP Content Server is also integrated via the HTTP interface (see [SAP HTTP Content Server 4.5 Interface \[Page 58\]](#)), the storage medium actually used for SAP applications remains fully transparent. You can switch to different storage media at any time.

SAP Business Workplace, SAP ArchiveLink, the document management system DMS and the Knowledge Warehouse are examples of applications that use the technical infrastructure of the content server.

Implementation Considerations

Installing the SAP Content Server is described in the installation guide *SAP Knowledge Provider Server Infrastructure Components* (German edition material number: 51005153, English edition material number: 51005154).

Constraints

The SAP Content Server is not a replacement for optical storage systems and other storage media for the long term archiving of documents.

KPro and Caching

KPro and Caching

A cache is used to store copies of documents the first time they are accessed. This means that the documents can then be accessed more quickly, since the contents are read directly from the cache. Caching, however, should not be confused with replication. With caching, the original documents are still stored in the same location, namely on the content server. The copies in the cache, however, can be overwritten at any time by new content.



Documents are checked in in Walldorf. An employee in South Africa wants to access and display these documents. The transmission time, however, is extremely long and the intercontinental network connections would be overloaded. By using cache servers, the documents are only copied over the connection once.

The Content Management Service of the SAP Knowledge Provider provides two methods of caching:

- Cache Server
Caching to special servers
- Client Cache
Caching at the user's frontend

Frontend Cache in SAPGUI

The client cache is used by the Data Provider to cache documents that are read from an HTTP Content Server or from the R/3 System.

This type of caching is extremely fast. However, users must fill their caches themselves, and additional resources are required at each frontend.



The caching settings can be made in the GUI options. Choose the *Local Data* tab and specify the directory to be used for caching and the maximum size of the cache, as well as the maximum size of the cache files.

Cache Server

The Content Server cache is used to cache special Content Server requests. Remote accesses are cached and executed locally.

This type of caching is suitable for scenarios in which many users have joint access to the cache. The documents only have to be sent once across the Wide Area Network.

Customizing for Caching

The following Customizing settings are required to use caching:

- Cache definition
 - Host name and locations (transaction *SCMSHO*)
 - Cache definition (transaction *SCMSCA*)
- Define locations for
 - Users (clients) via
 - Set/get parameter *LCA*
 - Host name (transaction *SCMSHO*)
 - Subnet (transaction *SCMSIP*)
 - Content server via
 - Host name (transaction *SCMSHO*)
 - Subnet (transaction *SCMSIP*)

Information on the individual activities can be found in the Implementation Guide (IMG) under *Basis* → *Basis Services* → *SAP Knowledge Provider* → *Content Management Service* → *Distribution*.

Cache Server

Cache Server

Purpose

Cache servers are used to speed up access to document contents if, for example, a document is to be displayed in Web browser. Cache servers can also reduce the network load and thereby enhance performance.



MS Internet Explorer uses a local cache on the user's hard disk.

Cache servers are similar to [content servers \[Page 35\]](#), but require less administration with the same level of access protection.

Implementation Considerations

Cache servers are installed from the *SAP Server Components CD-ROM* as part of the *SAP Knowledge Provider Server Infrastructure Components*. Installation instructions in PDF format are provided in English and German on this CD-ROM in the \KPro\Doku directory.

Content and Cache Server Administration

Use

SAP delivers a standard content server. This SAP Content Server can be administrated directly from the SAP System. The administrative settings in R/3 for external content servers and cache servers can also be defined here.



SAP DB-specific tools are available for monitoring and administrating the SAP DB.

From the KPro role (SAP_BC_SRV_KPR_ADMIN), you branch to the administration facility for the content and cache servers. To do so, choose the role *KPro* → *Environment* → *Content Server Administration* in the menu. Alternatively, choose transaction *CSADMIN*.

Features

You can retrieve the following information in the Administration facility:

- Informational overview of the content server/cache server
- Detailed information on the individual content repositories/cache servers
- Content repository certificates
- Content repository settings
- Content server/cache server statistics
- Creating content repositories



The *Create* tab is only displayed if you are in Change mode.

As of Release 4.6C, you can also go directly to Customizing from any screen that refers to a content repository. If you do so from change mode, all of the current values (HTTP server: port, HTTP script, and (if necessary) description) are copied automatically.



In order to be able to retrieve this information and to use the functions, you must first select a content server.


Selecting a Server

Selecting a Server

Use

You would select a content server in order to be able to retrieve information within Content Server Administration on this content server or the corresponding content repositories and use the functions in Content Server Administration.




You can use the  icon to branch to the administration of a different server at any time.

Prerequisites

- You are in Content Server Administration
- or
- You branch to Content Server Administration from central KPro Administration by choosing *Environment* → *CS Admin*.
- or
- You call up Content Server Administration with the transaction *CSADMIN*.

Procedure

1. Select a content server.
 - a. When you call Content Server Administration, you branch directly to the selection of content servers.
 - b. If you are already in Content Server Administration, you can select a server using the  icon.
2. You can continue in one of two ways:
 - a. In the *Content Repository* field, specify the name of the repository that you have defined in [Content Repository Maintenance \[Ext.\]](#).

You can use F4 help for a list of possible entries. The system then enters values in the other fields. However, this only happens in conjunction with the F4 Help.
 - b. Alternatively, you can enter an HTTP server and a port in the *HTTP Srv: Port* field.

You can use F4 help for a list of possible entries.

If you do not enter a value for the *HTTP Script*, the standard setting `ContentServer/ContentServer.d11` is used.
 - c. Alternatively, you can also enter the individual data directly.

This enables you to administrate content servers that are not known to this system (Customizing).
3. Choose *Enter* to confirm.

Result

The entries made for the HTTP server, port and script, as well as the version, appear in the header of Content Server Administration.

Functions

Functions

Informational Overview

Use


The tab page *Overview* contains general information about your content server.

Prerequisites

You are in Content Server Administration

Features

You can retrieve the following information:

- Information on the content server itself
 - Status
 - Possible statuses: running, defined, stopped, error
 - Status description
 - Explanation of status
 - Vendor
 - Vendor/Name of content server
 - Version
 - Content server version
 - Build
 - Storage system build
 - pVersion
 - SAP HTTP Content Server interface version
 - Server date
 - Current date
 - Server time
 - Current time on content server
 - Start date
 - Date on which content server was started
 - Start time
 - Time at which content server was started
- 
- All times are specified in UTC.





- Information about the content server's content repositories

Informational Overview

- Repository name
- Customizing

Information on whether the repository is known to the R/3 System in Customizing and whether the Customizing settings are consistent. Here, you can check whether the HTTP server: port and script match and whether the data entered manually or with the F4 Help matches the Customizing settings.

The following options are available here:

Icon	Descriptive text	Explanation
	Customizing ok	Customizing is consistent
	Customizing partially ok	The Administration data and the data in Customizing have minor differences, for example, upper and lower-case letters.
	Customizing missing	No Customizing settings have been maintained for the repository.
	Different Customizing settings	The Administration information differs from the Customizing data.



The Customizing information is available on all screens that refer to repositories.

- Description
 - Explanation of the content repository
- Status
 - Possible statuses: running, defined, stopped, error
- Status description
- pVersion
 - Version of the SAP Content Server HTTP interface
- Further content server-specific settings



Further values specific to the content server can be output here in addition to the information described above.

You can display detailed information by double-clicking a content repository.

Activities

You can update the information on the relevant server by choosing the *Refresh* icon.

Detailed Information

Use


The tab page *Details* contains detailed information on the content server's content repositories.

Prerequisites

You are in Content Server Administration

Features

The following functions are available:

- Branching to the detailed information on a different content repository
- You can go directly to Customizing for each repository by choosing .
- Refreshing the detailed information
- In change mode, you can:
 - Edit the description of the content repository
 - Go directly to the Customizing settings for a repository
 - The current values for the HTTP server: port, HTTP script, and (if necessary) the descriptive text are copied automatically.
 - Switch the digital signature check on or off
 - Start the content repository, that is, change its status from *defined* to *running*
 - Stop the content repository, that is, change its status from *running* to *defined*

Activities

You execute the above functions by choosing the relevant icons.

Certificates

Certificates

Use


The tab page *Certificates* contains information about the certificates sent to the content server.

Prerequisites

You are in Content Server Administration

Features

The following functions are available:

- Branching to the information about the certificates of a different content repository
- You can go directly to Customizing for each repository by choosing .
- Refreshing the information about the certificates
- You can send a certificate
- In change mode, you can:
 - Release the certificates
 - Recall the certificates
 - Delete a sent certificate
 - Go directly to the Customizing settings for a repository

The current values for the HTTP server: port, HTTP script, and (if necessary) the descriptive text are copied automatically.

Activities

You execute the above functions by choosing the relevant icons.

Settings

Use


The tab page *Settings* contains information about the settings for your content repositories.

Prerequisites

You are in Content Server Administration

Features

The following functions are available:

- Branching to the information about the settings for a different content repository
- You can go directly to Customizing for each repository by choosing .
- Refreshing the information about the settings
- In change mode, you can:
 - Change the settings for the Content Repository



These inputs are specific to each content server.

Changes made to the settings may only take effect once the content repository has been restarted.

- Go directly to the Customizing settings for a repository
 - The current values for the HTTP server: port, HTTP script, and (if necessary) the descriptive text are copied automatically.

Activities



As a general rule, no changes should be made here. The opportunities available on the tab page *Detail* to make changes, should suffice for changes necessary in the normal run of things.

You execute the above functions by choosing the relevant icons.

Statistics

Statistics

Use

The tab page *Statistics* contains statistical information on the individual HTTP commands.

Prerequisites

You are in Content Server Administration

Features

The following functions are available:

- You can read up to date information from the content server by using the *Refresh* icon
- You can delete the information in change mode.

In this case, a request is sent to the content server to reset the statistics.

Activities

You execute the above functions by choosing the relevant icons.

Creating New Content Repositories

Use

On the *Create* tab page, you can create new content repositories in addition to the existing content repositories

Prerequisites

You are in change mode in Content Server Administration

Features

You can use an existing content repository as a template when creating a new content repository.

If you have a standard installation of the SAP Content Server, you can transfer the values in the table control unchanged.

Activities

1. Enter a name for your new content repository.
2. Enter a short explanation.
3. Decide whether the digital signature check is to be performed.
4. Change the entries for the content storage host and so on, as necessary.
5. Save your entries.
6. You can continue in one of two ways:
 - a. If the content repository is created successfully, you branch directly to the detail display (*Detail* tab page). The content repository should have status *running*.
 - b. If errors occur during creation of the content repository, you should check the settings and correct them as necessary (tab page *Settings*) and repeat the creation procedure.

Content and Cache Server Monitoring

Use

The HTTP Content Server is monitored automatically as part of the Computer Center Management System (CCMS). Further information on the CCMS is provided in the sections on [Computing Center Management System \(CCMS\) \[Ext.\]](#), [Monitoring in the CCMS \[Ext.\]](#), and [The Alert Monitor \[Ext.\]](#).

All of the repositories defined in Customizing are monitored with their associated content servers. A monitoring facility for Web servers and cache servers is also provided.



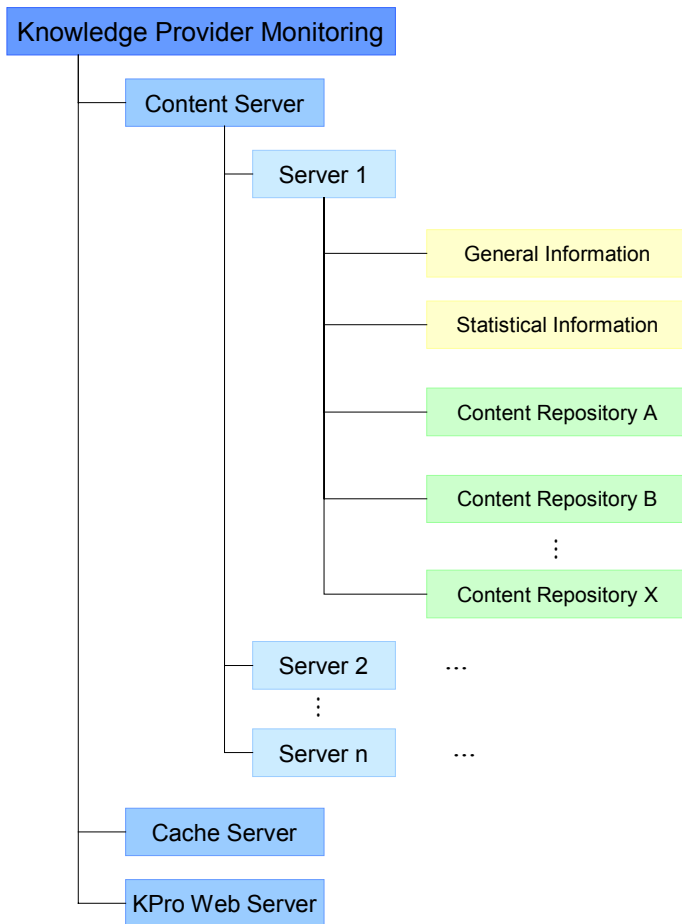
The KPro Web server is used to display content on the client. For example, a client wants to display a specific instance of a logical document. To do so, it sends the information on the logical document with attributes (for example, logon language, release) to an R/3 System via RFC. The R/3 System returns the matching physical document via RFC to the client, which then displays it via the KPro Web Server.

From the KPro role (SAP_BC_SRV_KPR_ADMIN), you branch to the monitoring facility for the content and cache servers. To do so, choose the role *KPro* → *Environment* → *Alert Monitor* in the menu. Alternatively, choose transaction *SCMSMO*.

Features

You can call up the following information in the monitoring facility:

- Information on the SAP Content Server and HTTP Content Server
- Information on the Cache Server
- Information on the KPro Web Server



If you are in *Current status* mode, you can also go directly from the monitoring facility for content and cache servers to the administration function for content and cache servers. To do so, double-click the required server or repository.



The monitoring information in the CCMS is available in English.

Monitoring for Content Servers

Monitoring for Content Servers

Use

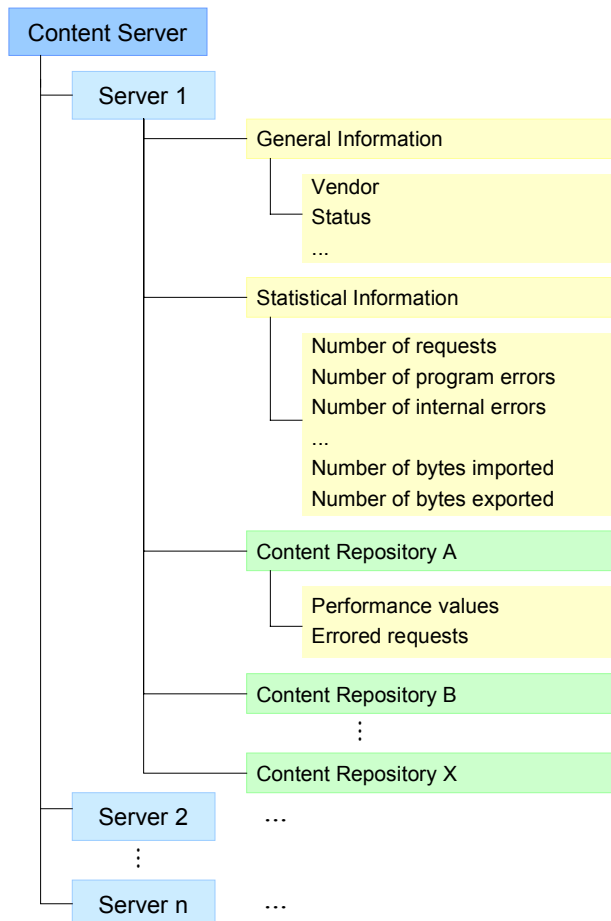
An automatic monitoring facility for the SAP Content Server and HTTP Content Server is provided. Information is output here for all of the content repositories defined.

Prerequisites

You are in Monitoring for the Knowledge Provider.

Features

The information output is shown in the figure below:



Content servers from SAP partners are also monitored. The information for these servers usually differs slightly.

Activities

Information on the activities of the CCMS Monitor is provided in the section on [The Alert Monitor \[Ext.\]](#).

By double-clicking the server or the repository, you can branch to the Administration facility directly (see also [Content and Cache Server Administration \[Page 39\]](#)).

Monitoring for Cache Servers

Monitoring for Cache Servers

Use

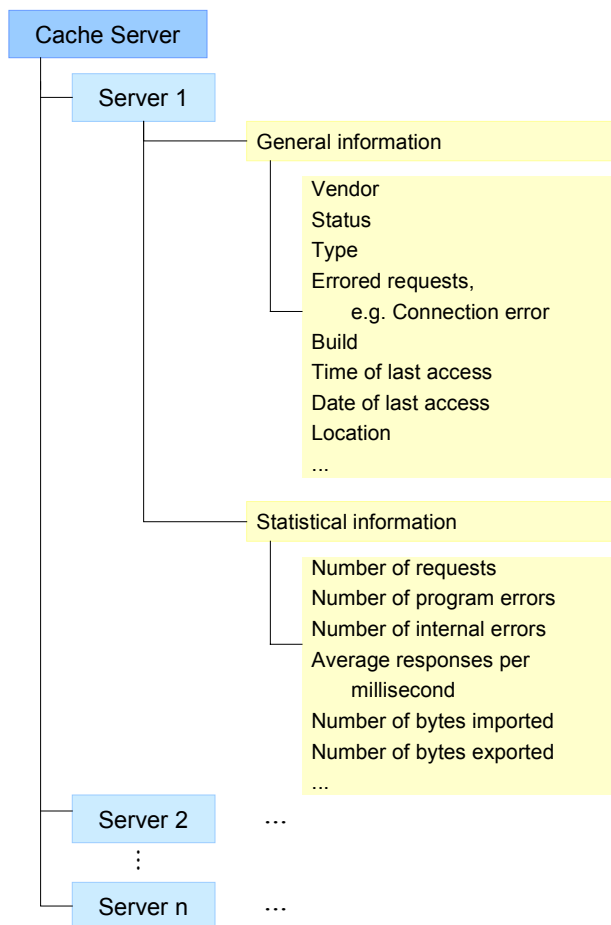
An automatic monitoring facility for cache servers is provided.

Prerequisites

You are in Monitoring for the Knowledge Provider.

Features

The information output is shown in the figure below:



Activities

Information on the activities of the CCMS Monitor is provided in the section on [The Alert Monitor \[Ext.\]](#).

By double-clicking the server, you can branch to the Administration facility directly (see also [Content and Cache Server Administration \[Page 39\]](#)).

Monitoring for KPro Web Servers

Monitoring for KPro Web Servers

Use

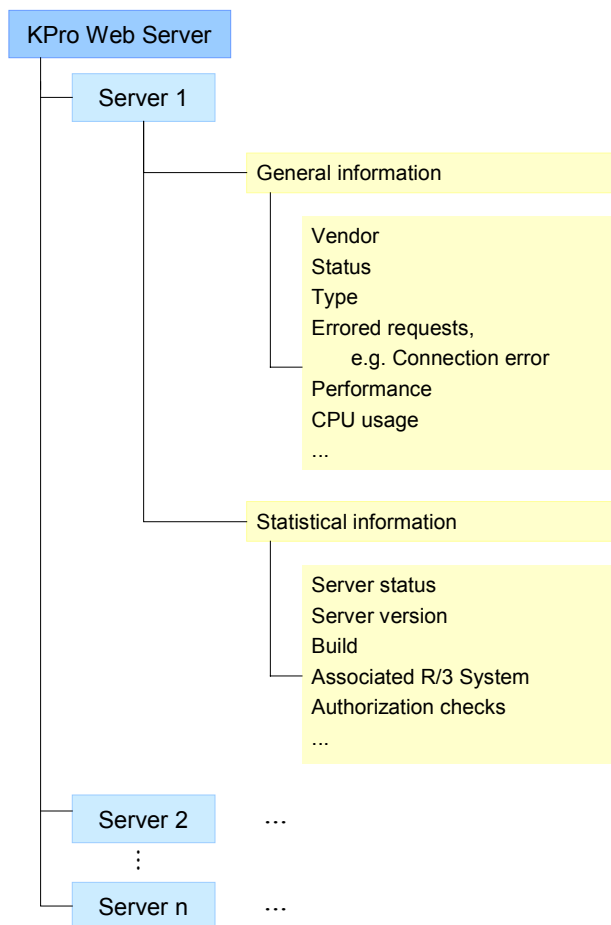
An automatic monitoring facility for Knowledge Provider Web servers is provided.

Prerequisites

You are in Monitoring for the Knowledge Provider.

Features

The information output is shown in the figure below:



Activities

Information on the activities of the CCMS Monitor is provided in the section on [The Alert Monitor \[Ext.\]](#).

SAP Content Server HTTP 4.5 Interface

This document describes the new **SAP Content Server interface HTTP 4.5 Interface**.



Points to note when transferring from SAP ArchiveLink to the new Content Server interface are explained in the relevant parts of the document.

The aim of the new interface is that only general industry standards such as HTTP and BAPIs should be used in communication with external storage systems (content servers).



The SAP Content Server HTTP 4.5 interface can be certified.

Introduction

Definition of Terms

Definition of Terms

For the purposes of the following description, a **document** comprises of **administrative data** and **content**:

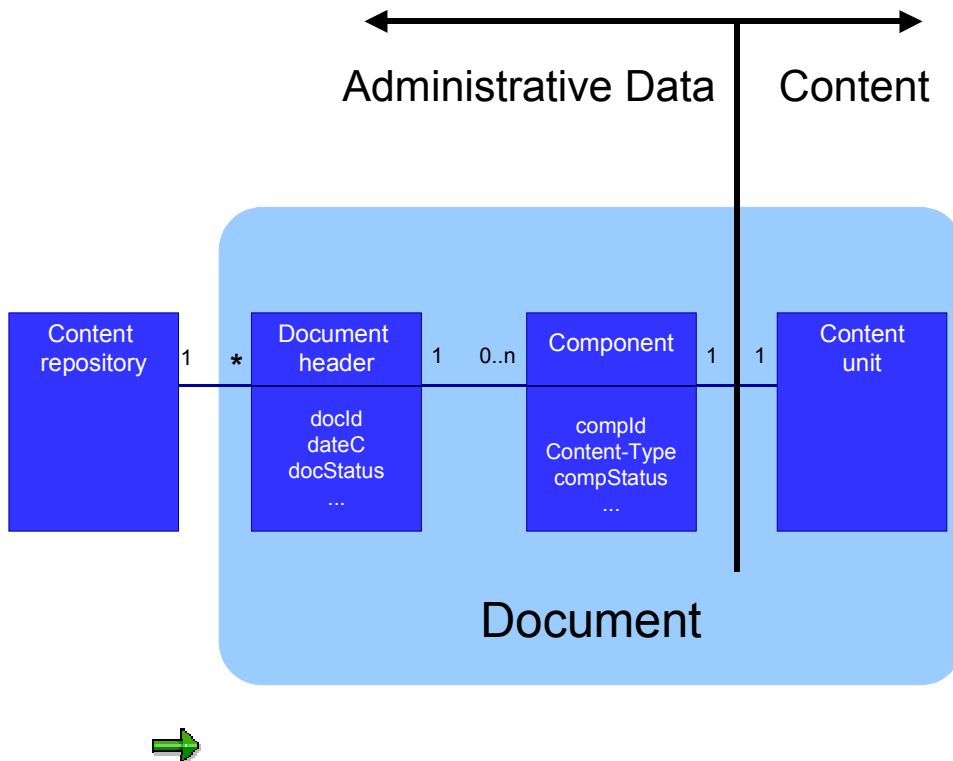
- **Administrative data** identifies and describes a document.
- The **content** of a document consists of closed datasets. The administrative data identifies and describes the content. One closed dataset is a **content unit**.

In SAP terminology, a **content server** is any server that manages content. A content server may be a database, a fileserver, an SAP R/3 System or an external archive.

The administrative data terms **content repository**, **document header** and **component** are of particular importance when identifying documents.

- A **content repository** represents the logical storage space for documents in a content server on an administrative level. Several content repositories can exist on one content server. A content repository is identified by the parameter `contRep`.
- The **document header** is an administrative quantity summarizing several components. It is identified by the parameter `docId`. A document header is assigned to one particular content repository.
- A **component** represents one particular content unit on an administrative level. It is assigned to one particular document header and is identified by the parameter `compId`.

The relationship between content repository, document header, component and content is shown in the diagram below:



Definition of Terms

It can be that a document with one component is generated and this component is then deleted. This leaves an "empty" document, that is, a document with no components. To avoid possible contradictions, we shall assume that such documents can occur. The special case of an empty component is also possible. This may occur, for example, where a file with a size of 0 bytes is stored.

The combination `contRep/docId` is the **one-to-one address** for a document header.

The combination `contRep/docId/compId` is the **one-to-one address** for a component.

Under certain circumstances documents are subject to **protection**. This means that functions executed on the document must be legitimized. For each document header, you can define whether or not legitimization is necessary for particular functions. This information is not defined in the document header for each Content Server interface function, but instead via **access modes**. Access modes are defined as disjunct groups of Content Server interface functions.

HyperText-Transfer-Protocol (HTTP) is a description of a communication process typically used to access objects on the **World Wide Web (WWW)**.

This protocol is currently being developed further by the W3C (WWW Consortium, <http://www.w3c.org>). The protocol HTTP/1.1 or HTTP/1.0 can be used for the communication process. RFC (**R**equest **F**or **C**omment) 2068 specifies protocol HTTP/1.1: Protocol HTTP/1.1 contains more precise regulations than protocol HTTP/1.0 (RFC 1945), which ensures the reliable implementation of HTTP characteristics.

The new interface is designed such that communication is always started by the client R/3 System. The content server addressed by the R/3 System is always only a server and never itself a client that instigates communication with the R/3 System.

HyperText-Markup-Language (HTML) is a standard format and description language for WWW pages.

Uniform Resource Locators (URLs), see RFC 1738) are a standardized mechanism used to address uniquely defined objects on the WWW. As well as the actual address, URLs can contain functions and parameters that can be interpreted by the object addressed.

UTC (**U**niversal **T**ime **C**oordinated) is used for all expressions of time in this specification.



The following rules apply to the spelling of functions, parameters and key words in this description:

All terms defined in the Hypertext Transfer Protocol HTTP/1.1 (RFC 2068) are used correspondingly (for example, `Content-Type`). Protocol HTTP/1.0 (RFC 1945) is also supported.

Terms specific to this interface description are not capitalized if they consist of one syntactical term (for example, `info`). A combination of lower and upper case is used if they consist of more than one syntactical term (for example, `contRep`).

Implementation

Implementation

The HTTP protocol is used for communication with content servers. Servers and documents are addressed using URLs and data is transferred in the Request-Body or in the Response-Body.

The URL specifies the function to be executed on a document: `get` (transferring from the server to client), `info` (retrieving information on the document) or `create` (creating a new document). The necessary parameters for these functions are also part of the URL.

This specification describes the URL syntax and the semantics for the various functions.

Security

Security and the related guarantee of secure data transfer are central aspects of the Content Server interface. The following principles apply:

- It is assumed that all authorization checks in the R/3 System are performed.
- To ensure that these authorization checks cannot be circumvented for content server access, a public/private key procedure is used (see also [Public Key Technology \[Ext.\]](#)).
- The public and private keys are R/3-specific, not user-specific.

The security concept of the Content Server interface is based on the fact that the R/3 System public key is stored in the Content Server. This is done using the command `putCert`. The content server uses the certificate to check URLs and signatures (see also [putCert \[Page 113\]](#)).



For more information, see the documentation [Secure Store & Forward / Digital Signatures \[Ext.\]](#).

secKey

secKey

The **secKey** ensures that a URL cannot be changed after it has been generated by the R/3 System. This ensures that access to the document is protected and that access protection is managed in the R/3 System. The **secKey** does not protect the content of the document. The following parameters are always signed in the **secKey**:

- **contRep** Content repository
- **accessMode** Access mode
- **authId** Client ID
- **expiration** Expiry time (UTC)

authId must be a unique identification of the client (for example, the R/3 System). The UTC expiry time is written in the format: `yyyymmddhhmmss`. If the expiry time has been exceeded, the content server must report HTTP status code 401 to the client.

If a **secKey** is transferred with the URL, the parameters **accessMode**, **authId** and **expiration** must also be transferred. These parameters need not be transferred if the **secKey** is not transferred.

Additional parameters must be signed. These depend on the particular function and are specified in the function description. The name of the function itself is not signed. The parameters to be signed can appear in the URL in any order. To check the signature, it must be ensured that the order in which the parameters are transferred to the signalmodule is the same as the order in the URL.

The **secKey** for the chosen procedure is about 500 bytes long.

The parameters to be signed for a particular function are specified in the function definition. They are specified in the last column of the parameter table. If these are *mandatory* parameters, they must always appear in the URL and are therefore always signed. Optional parameters can clearly only be signed if they are used. *s-mandatory* parameters must appear in the URL if a signature is used. They are always signed. If no signature is used, these parameters are not evaluated.

The URL parameters to be signed are the *Message*. The message is used to determine a hash value. The parameters must be kept in the same order for determination of the hash value. The hash or message digest is a one-way function and so cannot be reversed. Using the sender private key, the hash value is digitally signed according to DSS (Digital Signature Standard) via the SAP SSF (Secure Store & Forward) module according to PKCS#7. The digital signature is transferred in the URL in the parameter **secKey** (as described above).

Once the digital signature has been created, the URL parameters are safe from distortion. They are not encoded. All recipients can check the URL parameters using the sender public key. Any changes would therefore be detected. This ensures that an action on the content server can only be started, if the URL transferred has not been distorted.

Using the sender public key, the content server generates the message digest again from the transferred URL. It then forms a hash from the message (the order of the parameters in the URL is important here) and compares the two hashes (the message hash and the hash generated by the sender). If they match, the URL has not been distorted during transfer between the R/3 System and the content server.



The library for checking signatures is available from SAP AG if required. Since the standard format PKCS#7 was used for the signature, other products can also be used for decoding.

Brief technical information:

- Format of digital signature: PKCS#7 "Signed-Data"
- Public key procedure: DSS
- Key length: 512 – 1024 bits
- Public exponent: $2^{16} + 1$
- Public key format: X.509 v3 certificate
- MD (Message Digest) algorithm: MD5 or RIPEMD-160

Protection / Right of Access

Protection / Right of Access

The degree of protection is specified when a document is stored. When a document is accessed, the function a user may execute on this document is defined. Similar functions are grouped together. The groups are called access modes. They are listed in the following table:

Access mode	Abbreviation
Read	r
Create	c
Change	u
Delete	d



Protection applies to all components of the document. If the access mode is "change", corresponding components of a document can be deleted.

The access mode must be specified in the HTTP request as a parameter (**accessMode**). A combination of access modes can be specified, for example, **ud**. A **secKey** confirms the right of access. The corresponding access mode is specified in the descriptions of individual functions. When a document is accessed, the content server checks whether the **secKey** should be checked, that is, whether the document is protected regarding a particular function. It is often sensible that all users may read documents but only certain users may change documents. In this case, read protection would be deactivated (no **secKey** is required), for writing or deleting, however, a **secKey** must be transferred. The fact that the **secKey** can only be generated by the R/3 System ensures that an access protection check based on the R/3 authorization concept was performed.

Protection is defined when a document is created. This is done using the parameter **docProt**.



Protection	Description
docProt=	No access constraints
docProt=du	Delete and update operations only permitted with signed URLs. For delete operations, the accessMode must contain at least one d and for update operations at least one u . Read operations can be performed without a signature.



It is permitted to transfer several access modes: for example, **accessMode rd** for a read operation. This makes specific scenarios possible: If a get-URL with **accessMode=rd** and the corresponding signature is transferred to a client program, the client has the option not only of reading the document, but also of deleting the

entire document. To use the URL for deleting, it suffices to replace the command `get` with `delete` and to not transfer the `compId` (if it exists). Since the same parameters are signed for `get` and `delete`, the signature remains valid here. If the `accessMode` contains a `d`, it is possible in this example to delete the document.

On the basis of the access mode of an operation and the concrete protection of a document, the content server decides whether the `secKey` is to be checked. If the content server decides that it is not to be checked, all s-mandatory parameters are obsolete and it is not necessary to check these parameters either.



If they are unnecessarily transferred anyway, you can check them, but this does not increase security and is therefore superfluous, especially since you can increase performance in operations where protection is not required.

The parameter `docProt` is optional, but is generally transferred even if the URL is not signed. If neither the content server nor the R/3 System use the signature, this does not make a difference to the protection definition when creating documents.

If the parameter `docProt` is not transferred, the server default setting is to be used. The content server has complete freedom here.

If the R/3 System uses this opportunity, it must employ maximum protection and use the corresponding signed URLs for all subsequent accesses to the relevant documents.



The signature in the R/3 System may only be deactivated if no check is to take place in the content server.

In productive operation, you should generally use the signatures, however.

For all access modes, it must be possible to set as default on the Content Server whether a `secKey` must be specified or not. This default can be overwritten for the functions `create` and `mCreate`. If no protection is specified, the default is used.

Old data and documents that were stored in the content server not using the HTTP interface are subject to the highest level of protection, that is, all accesses must be signed.

Syntax

Syntax

General

The URL syntax is:

`http://servername:port/script?command¶meters`

The **servername** is the name of the server machine which is accessed and **port** (optional) is a TCP/IP port that can be used to address the server. **script** is the name of the program used to access the content server. This may be a DLL, a CGI script or an **Active Server Page (ASP)**. The object is created by the content server provider. A command must exist, followed by one or more **parameters**.



There must not be any blank spaces in the URL.



`http://pswdf009:1080/ContentServer/ContentServer.dll?get&pVersion=0045&contRep=K1&docId=361A524A3ECB5459E0000800099245EC`

Coding in the URL

Coding in the URL

The structure of URLs is described in RFC 1738. It also specifies which character set may be used for a URL and how characters not in this set should be encoded.

Only characters from a ASCII character set may be used in a URL (0x00 - 0x7F). Some of these characters must be encoded. (The characters 0x00 - 0x1F and 0x7F.) If they are to appear in the URL, a '%' (percentage sign) followed by the hexadecimal representation of the character should be used.



A line feed (0x0A) is represented as %0A (for example) in a URL.

Unsafe Characters

A set of *unsafe* characters must be coded in the same way: space, <, >, ", #, %, {, }, |, \, ^, ~, [,], ` , . These characters are unsafe either because they execute special functions in the URL or because they could be interpreted as special characters during transfer.

Reserved characters

There are also *reserved* characters: ;, /, ?, :, @, =, &.

Reserved characters must also be encoded.

Transferring Binary Data

A further problem occurs if binary data is to be transferred in a URL. This is the case when using this interface since the `secKey` consists of binary data. Coding must first be carried out in the ASCII character set. Base64 coding must be used (RFC 1521).

Example

1. Compiling the URL

```
http://pswdf009:1080/ContentServer/ContentServer.dll?get&pVersion=0045&contRep=K1&docId=361A524A3ECB5459E0000800099245EC&accessMode=r&authId=pawdf054_BCE_26&expiration=19981104091537
```

2. Generating the secKey

The `secKey` is made up of the encoded parameters. The parameters to be signed are specified in the function definition.

In the present example (`get` function) they are:

```
ContRep = K1  
DocId = 361A524A3ECB5459E0000800099245EC  
AccessMode = r  
AuthId = pawdf054_BCE_26  
Expiration = 19981104091537
```

In the next step the parameter values are summarized according to the sequence in the URL without separators to form a message:

Coding in the URL

```
K1361A524A3ECB5459E0000800099245ECrpdf054_BCE_2619981104091537
```

The message is used to form the hash from which the `secKey` is calculated. For reasons of clarity, arbitrary values will be taken for the `secKey` in this example.

`secKey` value: 0x83, 0x70, 0x21, 0x42.

3. Encoding the `secKey` in the ASCII character set

Base64 must always be used to encode the `secKey`.

0x83, 0x70, 0x21, 0x42 -> g3AhQg==

4. Encoding the URL in accordance with the URL character set limitations

Characters may need to be encoded. That is the case in this example:

g3AhQg== -> g3AhQg%3D%3D

The following URL is generated:

```
http://pswdf009:1080/ContentServer/ContentServer.dll?get&pVersion=0045&contRep=K1&docId=361A524A3ECB5459E0000800099245EC&accessMode=r&authId=pdf054_BCE_26&expiration=19981104091537&secKey=g3AhQg%3D%3D
```

Coding in the Response Body

Coding in the Response Body

Many of the functions described return information in the response body. If the information is returned in ASCII format, the lines always consist of key/value pairs separated by a semicolon:

```
key1="value1";key2="value2";...keyn="value2";CRLF
```

Only ASCII characters that can be printed may be used. If a value contains an inverted comma, this is in addition to the inverted commas already inserted around the value.

Functions

The functions available and their parameters are described below. For each function, the effect, the possible parameters and an example are given.

- **Effect**
Under Effect, the executed function is described. The meaning of the individual parameters is given.
- **Default**
Under Default, the effect of transferring only the mandatory parameters of a function with many parameters is described.
- **Access mode**
Under Access mode, the access mode for the function is specified.
- **Client → Server**
Under Client → Server, the parameters that are transferred from the client to the server are listed and specified as optional or mandatory. It is specified whether the parameters are optional or mandatory. s-mandatory means that the relevant parameter must only be specified if a `secKey` is transferred. The HTTP-Request type is defined and the way the parameters are to be coded in the URL and/or in the body is described.
- **Example**
Under Example, a function is performed using example parameters. Line breaks in the examples are purely to aid legibility. The actual URLs do not contain any line breaks.
- **Server → Client**
Under Server → Client, the structure of the HTML-Response is defined. This response is generated by the server and sent to the client.

The HTTP status codes specific to the content server are also listed in this section. If no security key is entered, this may cause (for example) error 401 (unauthorized); a wrongly addressed document can cause error 404 (not found). If an error occurs, the content server must also deliver an ASCII string describing the error. The error must be entered in the header field `x-ErrorDescription`.

Function Overview

Command	Effect	Access mode
<code>info</code>	Retrieve information about the document	r
<code>get</code>	Fetch (a range of) a content unit of a component	r
<code>docGet</code>	Fetch the whole content of a document	r
<code>create</code>	Create a new document	c
<code>mCreate</code>	Create several new documents	c

Functions

append	Append data to a content unit	u
update	Modify an existing document	u
delete	Delete a document or a component	d
search	Search for a text pattern within a content unit	r
attrSearch	Search for one or more attributes within a document (search within a print list)	r
putCert	Transfer client (for example, the R/3 System) certificate	-
serverInfo	Retrieve information about the content server and the corresponding content repositories	-

Access Functions

info

info

Effect

Document information is retrieved. As well as the document header information, the server sends information on all components. If information on only one component is required, a `compId` should be specified. The command `info` has the same effect as the command `docGet`, except that with `info` no component data is transferred.

Using `resultAs`, you can specify the format in which the information is to be provided. Return values can be provided in an ASCII format which can be parsed easily or in an HTML file. Use of `resultAs` is optional, `ascii` is standard. The format is defined further below.

If `resultAs=ascii` and the function is executed successfully, the data is transferred as an entity body in *multipart/form-data* format (see RFC 1867) as a response to an `HTTP GET-Request`.

Default

Standard information about the document header and the components of the addressed document is returned in ASCII format. The results are given in ASCII format.

Access Mode

read (r)

Client → Server

The client sends an `HTTP-GET-Request`. The URL contains the following parameters:

Parameter	Optional/Mandatory	Default	Sign
<code>contRep</code>	mandatory		X
<code>docId</code>	mandatory		X
<code>compId</code>	optional		
<code>pVersion</code>	mandatory		
<code>resultAs</code>	optional	ascii	
<code>accessMode</code>	s-mandatory		X
<code>authId</code>	s-mandatory		X
<code>expiration</code>	s-mandatory		X
<code>secKey</code>	optional		



s-mandatory means that this parameter must only be specified if the URL is signed.

Example

<http://pswdf009:1080/ContentServer/ContentServer.dll?info&pVersion=0045&contRep=K1&docId=361A524A3ECB5459E0000800099245EC>

The example is a request for information about the document header and all the document components. Information about the document header and all components of the document is requested.

Server → Client

The server answers the request with a response. The response status code indicates the outcome of the call.

HTTP Status Code	Meaning
200 (OK)	OK, information delivered
400 (bad request)	Unknown function or unknown parameter
401 (unauthorized)	Breach of security
404 (not found)	Document/component not found
409 (conflict)	Administrative data not accessible
500 (Internal Server Error)	Internal error in content server

The response header contains the following information about the document:

Keyword	Format	Meaning
Content-Type	String	Content-Type (if known)
boundary	String	Separator between individual components
Content-Length	Integer string	Entire length of the body actually transferred
X-dateC	YYYY-MM-DD	Creation date (UTC)
X-timeC	HH:MM:SS	Creation time
X-dateM	YYYY-MM-DD	Last changed on (UTC)
X-timeM	HH:MM:SS	Last changed at (UTC)
X-numberComps	Integer string	Number of components
X-contentRep	String	Content repository
X-docId	String	Document ID
X-docStatus	String	Status
X-pVersion	String	Version

Each time the function is called, all document header information is provided and, if no particular component is addressed, information on all components. If information is required on only one

info

component, output can be limited to this component by specifying the `compId`. The following combinations are possible:

- `docId=ID, compId=ID`:
Information about the document header and one component is provided.
- `docId=ID`:
Information about the document header and all components of the document is provided.

The component header contains the following information about the component:

Keyword	Format	Meaning
<code>Content-Type</code>	String	<code>Content-Type</code> (if known)
<code>charset</code>	String	Character set (if known)
<code>version</code>	String	Application version used to create the content of the component
<code>Content-Length</code>	Integer string	Actual body size in the response, always 0
<code>X-Content-Length</code>	Integer string	Size of the component in bytes
<code>X-compId</code>	String	Component ID
<code>X-compDateC</code>	YYYY-MM-DD	Creation date (UTC)
<code>X-compTimeC</code>	HH:MM:SS	Creation time
<code>X-compDateM</code>	YYYY-MM-DD	Last changed on (UTC)
<code>X-compTimeM</code>	HH:MM:SS	Last changed at (UTC)
<code>X-compStatus</code>	String	Component status
<code>X-pVersion</code>	String	Interface version

There are two ways of coding the results in the response body. The parameter `resultAs` controls coding.

1. `resultAs=ascii` (default)

The server sends a response in *multipart/form-data* format (see RFC 1867). The total length of the body is specified by the parameter `Content-Length` in the response header. The individual parts of the response body are separated by a boundary defined in the response header. Each part represents one component. Each component has a component header and a component body with Length 0, because no component data is transferred (in contrast to the `docGet` command). The component parameter `Content-Length` is therefore always immediately set to 0. Alternatively, the component length can be determined by the parameter `X-Content-Length`.

If the `charset` of a component is known, it must be transferred as a `Content-Type` parameter. Likewise, the parameter `version` (that is, the version number of the application used to create the component content (see [Parameters and Key Words](#)

[\[Page 120\]](#)) for a component must be transferred as a Content-Type parameter, if known.



```
HTTP/1.1 200 (OK)
Server: Microsoft-IIS/4.0
Date: Wed, 04 Nov 1998 07:41:03 GMT
Content-Type: multipart/form-data;
boundary=A495ukjfasdfddrg4hztzu...
Content-Length: 32413
X-dateC: 1998-10-07
X-timeC: 07:55:57
X-dateM: 1998-10-07
X-timeM: 07:55:57
X-contentRep: K1
X-numberComps: 2
X-docId: ID
X-docStatus: online
X-pVersion: 0045

--A495ukjfasdfddrg4hztzu898aA0jklmAxcvla12319981147528895
  Content-Type: application/x-alf; charset=
  Content-Length: 0
  X-compId: descr
  X-Content-Length: 2591
  X-compDateC: 1998-10-07
  X-compTimeC: 07:55:57
  X-compDateM: 1998-10-07
  X-compTimeM: 07:55:57
  X-compStatus: online
  X-pVersion: 0045

--A495ukjfasdfddrg4hztzu898aA0jklmAxcvla12319981147528895
  Content-Type: application/x-alf; charset=
  Content-Length: 0
  X-compId: data
  X-Content-Length: 29213
  X-compDateC: 1998-10-07
  X-compTimeC: 07:55:57
  X-compDateM: 1998-10-07
  X-compTimeM: 07:55:57
  X-compStatus: online
  X-compStatus: online
  X-pVersion: 0045
```

```
--A495ukjfasdfddrg4hztzu898aA0jklmAxcvla12319981147528895--
```

With the `info` command for an empty document, for example, the response body contains the following:

```
--A495ukjfasdfddrg4hztzu898aA0jklmAxcvla1231999102562159269
--A495ukjfasdfddrg4hztzu898aA0jklmAxcvla1231999102562159269--
```

info**2. resultAs=html**

If `resultAs=html` is set, the server sends an HTML page. The structure of the HTML page is not specified and graphical elements can be used freely.

get

Effect

A content unit of a component or a range within a content unit is retrieved from the content repository. The parameters `contRep`, `docId` and `compId` describe the component. The range of the content unit is described by `fromOffset` and `toOffset`.

If the function is executed successfully, the content unit is transferred from the server to the client as an entity body in the response to an HTTP GET-Request.

Default

If no `compId` is specified, the following conditions must be tested in the corresponding order:

1. If there is a component "data", this component is returned.
2. If there is a component "data1", this component is returned.

The function returns error 404 (not found), if a wrong `compId` or no `compId` was specified and none of the above conditions is fulfilled.

Access mode

Read (r)

Client → Server

The client sends an `HTTP-GET-Request`. The URL contains the following parameters:

Parameter	Optional/Mandatory	Default	Sign
<code>contRep</code>	mandatory		X
<code>docId</code>	mandatory		X
<code>compId</code>	optional	see above	
<code>pVersion</code>	mandatory		
<code>fromOffset</code>	optional	0	
<code>toOffset</code>	optional	-1	
<code>accessMode</code>	s-mandatory		X
<code>authId</code>	s-mandatory		X
<code>expiration</code>	s-mandatory		X
<code>secKey</code>	optional		

get



s-mandatory means that this parameter must only be specified if the URL is signed.

Example

```
http://pswdf009:1080/ContentServer/ContentServer.dll?get&pVersion=0045&
contRep=K1&docId=361A524A3ECB5459E0000800099245EC&compId=data
```

The document component “data” is requested.

Server → Client

The server answers the request with a response. The response status code indicates the outcome of the call.

HTTP Status Code	Meaning
200 (OK)	OK, content unit of component is transferred
400 (bad request)	Unknown function or unknown parameter
401 (unauthorized)	Breach of security
404 (not found)	Document/component not found
409 (conflict)	Document/component not accessible
500 (internal server error)	Internal error in content server

The response header contains the following standard information about the document:

Keyword	Meaning
Content-Type	Content-Type
charset	The character set of the component (as a Content-Type parameter).
version	The version of the component (as a Content-Type parameter).
Content-Length	Length of document

The response Content-Type depends on the Content-Type of the component requested. If the **charset** of a component is known, it must be transferred as a **Content-Type** parameter.

Likewise, the parameter **version** (that is, the version number of the application used to create the component content (see [Parameters and Key Words \[Page 120\]](#))) for a component must be transferred as a **Content-Type** parameter, if known.

The content unit (or range within the content unit) of the component is transferred in the response body.

docGet

Effect

The entire content of a document is retrieved from the content repository.

If an incorrect `docId` was specified, error 404 (not found) occurs. The error is always error 404 (not found).

If the function is executed successfully, the data is transferred as an entity body in *multipart/form-data* format (see RFC 1867) as a response to an `HTTP GET-Request`.

Default

-

Access mode

Read (r)

Client → Server

The client sends an `HTTP-GET-Request`. The URL contains the following parameters:

Parameter	Optional/Mandatory	Default	Sign
contRep	mandatory		X
docId	mandatory		X
pVersion	mandatory		
accessMode	s-mandatory		X
authId	s-mandatory		X
expiration	s-mandatory		X
secKey	optional		



s-mandatory means that this parameter must only be specified if the URL is signed.

Example

```
http://pswdf009:1080/ContentServer/ContentServer.dll?docGet&pVersion=0045&contRep=K1&docId=361A524A3ECB5459E0000800099245EC
```

The entire content of a document is transferred to the client.

docGet**Server → Client**

The server answers the request with a response. The response status code indicates the outcome of the call.

HTTP Status Code	Meaning
200 (OK)	OK, document is transferred
400 (bad request)	Unknown function or unknown parameter
401 (unauthorized)	Breach of security
404 (not found)	Document/component not found
409 (conflict)	Document/component not accessible
500 (internal server error)	Internal error in content server

The server sends a response in *multipart/form-data* format (see RFC 1867). The individual parts of the response body are separated by a boundary defined in the response header. In contrast to the `info` command, when `docGet` is used, components are actually transferred and the Length of the transferred components is specified in the field `Content-Length` of the relevant component, that is, `Content-Length` and `X-Content-Length` have identical values. The response header contains the following information about the document:

Keyword	Format	Meaning
<code>Content-Type</code>	String	Content-Type, always multipart/form-data
<code>boundary</code>	String	Separator between individual components
<code>Content-Length</code>	Integerstring	Entire length of the body actually transferred
<code>X-dateC</code>	YYYY-MM-DD	Creation date (UTC)
<code>X-timeC</code>	HH:MM:SS	Creation time
<code>X-dateM</code>	YYYY-MM-DD	Last changed on (UTC)
<code>X-timeM</code>	HH:MM:SS	Last changed at (UTC)
<code>X-numComps</code>	Integerstring	Number of components
<code>X-contRep</code>	String	Content repository
<code>X-docId</code>	String	Document ID
<code>X-docStatus</code>	String	Status
<code>X-pVersion</code>	String	Version

The component header contains the following information about the component:

Keyword	Format	Meaning
<code>Content-Type</code>	String	Content-Type (if known)

charset	String	Character set (if known)
version	String	
Content-Length	Integerstring	Actual Body size in the response, always 0
X-Content-Length	Integerstring	Size of the component in bytes
X-compId	String	Component ID
X-compdateC	YYYY-MM-DD	Creation date (UTC)
X-compTimeC	HH:MM:SS	Creation time
X-compDateM	YYYY-MM-DD	Last changed on (UTC)
X-compTimeM	HH:MM:SS	Last changed at (UTC)
X-compStatus	String	Component status
X-pVersion	String	Interface version

If the **charset** of a component is known, it must be transferred as a **Content-Type** parameter. Likewise, the parameter **version** (that is, the version number of the application used to create the component content (see [Parameters and Key Words \[Page 120\]](#))) for a component must be transferred as a **Content-Type** parameter, if known.

Example:

```
HTTP/1.1 200 (OK)
Server: Microsoft-IIS/4.0
Date: Wed, 04 Nov 1998 07:41:03 GMT
Content-Type: multipart/form-data; boundary=A495ukjfasdfddrg4hztzu...
...some more header information...
Content-Length: 32413
X-dateC: 1998-10-07
X-timeC: 07:55:57
X-dateM: 1998-10-07
X-timeM: 07:55:57
X-contRep: K1
X-numComps: 2
X-docId: ID
X-docStatus: online
X-pVersion: 0045

--A495ukjfasdfddrg4hztzu898aA0jklmAxcvla12319981147528895
Content-Type: application/x-alf; charset=
Content-Length: 2591
X-compId: descr
X-Content-Length: 2591
X-compDateC: 1998-10-07
X-compTimeC: 07:55:57
X-compDateM: 1998-10-07
X-compTimeM: 07:55:57
X-compStatus: online
X-pVersion: 0045
```

docGet

```
...component data ...
--A495ukjfasdfddrg4hztzu898aA0jklmAxcvla12319981147528895
Content-Type: application/x-alf; charset=
Content-Length: 29313
X-compId: data
X-Content-Length: 29213
X-compDateC: 1998-10-07
X-compTimeC: 07:55:57
X-compDateM: 1998-10-07
X-compTimeM: 07:55:57
X-compStatus: online
X-compStatus: online
X-pVersion: 0045
```

```
...component data ...
--A495ukjfasdfddrg4hztzu898aA0jklmAxcvla12319981147528895--
```

When the `docGet` command is used on an empty document, the following is an example of what could be in the Response-Body:

```
--A495ukjfasdfddrg4hztzu898aA0jklmAxcvla1231999102562159269
--A495ukjfasdfddrg4hztzu898aA0jklmAxcvla1231999102562159269--
```

create

Effect

A document with one or more components is stored in the content repository. The parameters `contRep`, `docId` and `compId` describe the component (see [Definition of Terms \[Page 60\]](#)). The function is used to create new documents. If a document already exists in the content repository, an error occurs in the function. The functions `update` and `append` can be used to modify existing documents. The `create` function always creates an entire document.

The function can be called once with an HTTP-PUT or POST (for further information, see [HTTP-PUT \[Page 89\]](#) and [HTTP-POST multipart/form-data \[Page 90\]](#)).

Default

A new document with the specified `docId` is created. One or more components are stored in the content repository. Protection is according to the standard set on the content server.

Access Mode

create (c)

Client → Server

The following parameters exist:

Parameter	Optional/Mandatory	Default	Position (POST/PUT)	Sign (POST/PUT)
<code>contRep</code>	mandatory		URL/URL	X/X
<code>compId</code>	mandatory		body/URL	-/X
<code>docId</code>	mandatory		URL/URL	X/X
<code>pVersion</code>	mandatory		URL/URL	
<code>Content-Type</code>	optional		body/body	
<code>charset</code>	optional		body/body	
<code>version</code>	optional		body/body	
<code>Content-Length</code>	mandatory		Header body/ Header body	
<code>docProt</code>	optional	server setting	URL/URL	X/X
<code>accessMode</code>	s-mandatory		URL/URL	X/X

create

authId	s-mandatory		URL/URL	X/X
expiration	s-mandatory		URL/URL	X/X
secKey	optional		URL/URL	



For HTTP-POST, the **Content-Length** in the request header is the total length of the body and the **Content-Length** in each part header is the length of the individual content units. For HTTP-PUT, the **Content-Length** is always the total length of the body.

It makes a difference whether the parameter **docProt** is not transferred or whether nothing is transferred as **docProt** (**docProt=**). In the first case, the content server default is used. In the second case it is specified explicitly that no protection exists.

s-mandatory means that this parameter must only be specified if the URL is signed.

The function can be executed in two ways. A single component can be transferred using an [HTTP-PUT \[Page 89\]](#). Alternatively, an [HTTP-POST \[Page 90\]](#) is used in the format **multipart/form-data**. In the first version, only one single component can be loaded on the server in each case. This restriction does not apply in the second version; 0 to n components can be transferred.

HTTP-PUT

In this case, all of the parameters are entered in accordance with the table in the [create \[Page 87\]](#) section.



```
http://pswdf009:1080/ContentServer/ContentServer.dll?create&pVersion=0045  
&contRep=K1&docId=4B7689654E73D21197E70060B0672A3C&compId=data&  
Content-Length=300
```

The document component “data” is stored. The component data is transferred as an entity body.

HTTP-POST multipart/form-data

HTTP-POST multipart/form-data

The data is transferred as **HTTP-POST** and as **multipart/form-data**. The document header information is transferred in the URL. One or more components are transferred in the body. This version of the function is particularly suitable for transferring documents consisting of several components into the content repository, as a whole. The component information is specified in the header of each part; the data in the body.

In practice, this means that the URL contains the parameters **contRep**, **docId**, **pVersion**, **docProt**, **accessMode**, **authId**, **expiration** and **secKey**. All the other parameters are in the body.

The request body is in **multipart/form-data** format. With this format, it is possible to transfer several independent parts to an HTTP content server. The individual parts have a header and a body and are in MIME format (RFC 2045, 2046). This MIME format enables several components to be transferred to the content server simultaneously. If an error occurs when storing a component, the entire action is cancelled.

The parameters **compId** and **Content-Type** are contained in the header of each part. The **CompId** is transferred in field **X-compId**. The component length is in the field **Content-Length**. The parameters **charset** and **version** can be appended to the **Content-Type**.

Example 1

```
http://pswdf009:1080/ContentServer/ContentServer.dll?create&pVersion=0045&contRep=K1&docId=4B7689654E73D21197E70060B0672A3C
```

A document consisting of one or more components is transferred in **multipart/form-data** format.

Document header

```
Content-Type: multipart/form-data;
boundary=A495ukjfasdfddrg4hztzu898aA0jklm
...some more header information...
Content-Length: 32413
```

Content part

```
--A495ukjfasdfddrg4hztzu898aA0jklm
X-compId: data
Content-Type: application/msword; charset=ISO-8859-1; version=6
Content-Length: 4242

... 4242 Bytes Data ...
--A495ukjfasdfddrg4hztzu898aA0jklm--
```

Example 2 (Create with 0 Components)

```
http://pswdf009:1080/ContentServer/ContentServer.dll?create&pVersion=0046&contRep=M1&docId=3810FF00804C257DE10000009B38FA09&docProt=ud&accessMode=c&authId=CN%3DKPR&expiration=19991025080635&secKey=MIIBlQYJKoZIhvcNA ...
```

HTTP-POST multipart/form-data

A document consisting of one or more components is transferred in `multipart/form-data` format.

Document header

```
Content-Type: multipart/form-data; boundary=KoZIhvcNAQcB
...some more header information...
Content-Length: 38
```

Content part

```
--KoZIhvcNAQcB
--KoZIhvcNAQcB--
```

Server → Client

The server answers the request with a response. The response status code indicates the outcome of the call.

HTTP Status Code	Meaning
201(created)	OK, document(s) created
400 (bad request)	Unknown function or unknown parameter
401 (unauthorized)	Breach of security
403 (forbidden)	Document already exists
500 (internal server error)	Internal error in content server

The content server must set the dates (`dateC` and `compDateC`) and the times (`timeC` and `compTimeC`) for creating components and the document.

mCreate

mCreate

Effect

One or more documents each with one or more components are stored in the content repository. The function has a similar effect to that of several sequential `create` functions, but is significantly more efficient for signed URLs in storing many new documents at once.

Within an `mCreate` call, objects can only be stored in one and the same content repository. The content repository is described by the URL parameter `contRep`.

The individual components of the documents are transferred in a *multipart/form-data* entity body. Components of the same document must be transferred one after the other so that transfer of a document can be assumed to be complete, as soon as a component from a different document begins.

The parameter `docId` is absolutely necessary for all components (each multipart-part) and is entered as the headerfield `"X-docId"`.

Storage of one document is performed within one transaction, but not the storage of all documents transferred in one `mCreate` call.

Access mode

Create (c)

Client → Server

The client sends an `HTTP-POST-Request`. The following parameters exist:

Parameter	Header Field in Body	Optional/Mandatory	Default	Position	Sign
<code>contRep</code>		mandatory		URL	X
<code>compId</code>	<code>"X-compId"</code>	mandatory		body	
<code>docId</code>	<code>"X-docId"</code>	mandatory		body (1. <code>docId</code> also in URL)	X (1. <code>docId</code>)
<code>pVersion</code>		mandatory		URL	
<code>Content-Type</code>		optional		body	
<code>charset</code>		optional		body	
<code>version</code>		optional		body	

Content-Length		mandatory		body	
docProt		optional	server setting	URL	X
accessMode		s-mandatory		body	X
authId		s-mandatory		URL	X
expiration		s-mandatory		URL	X
secKey		optional		URL	




s-mandatory means that this parameter must only be specified if the URL is signed.

The parameters for which the request body is specified as position are transferred in the header field of the multipart part of the corresponding components. For special parameters in this interface, the name of the header field is in column 2 of the table.

Server → Client

The server answers the request with a response. The response status code indicates the outcome of the call. A distinction is made between general status codes, which describe the success of the call as a whole, and specific status codes, which document the creation of individual documents.

General HTTP Status Codes	Meaning
201(created)	OK, all documents were created
250 (missing documents created)	OK, all missing documents were created  In practice, this status can only occur when <code>mcreate</code> is called more than once.
400 (bad request)	Unknown function or unknown parameter
401 (unauthorized)	Breach of security
500 (internal server error)	Internal error in content server

Specific HTTP Status Codes	Meaning
----------------------------	---------

mCreate

201(created)	OK, document created
403 (forbidden)	Document already exists
500 (internal server error)	Internal error in content server

An ASCII text must be returned whether the function is executed successfully or whether an error occurs. The documents stored (HTTP status code 201) and/or not stored (HTTP status code 403) are specified in this text. The following format is used:

```
docId="string";retCode="integerstring";errorDescription="string";CRLF
```

The value of the parameter `retCode` is the corresponding specific HTTP status code.

As a summary, the response body contains the following standard information about each document:

Keyword	Format	Meaning
docId	string	Document ID
retCode	Integerstring	HTTP status code
errorDescription	string	Text explaining the error (optional)

append

Effect

Data is appended to a content unit of a component in the content repository. The parameters `contRep`, `docId` and `compId` describe the component (see [Definition of Terms \[Page 60\]](#)). The document addressed and the corresponding component must exist.

Default

Data is appended to the content unit of the addressed component.

Access mode

change (u)

Client → Server

The client sends an `HTTP-PUT-Request`. The URL contains the following parameters:

Parameter	Optional/Mandatory	Default	Sign
<code>contRep</code>	mandatory		X
<code>docId</code>	mandatory		X
<code>compId</code>	mandatory		X
<code>pVersion</code>	mandatory		
<code>accessMode</code>	s-mandatory		X
<code>authId</code>	s-mandatory		X
<code>expiration</code>	s-mandatory		X
<code>secKey</code>	optional		



s-mandatory means that this parameter must only be specified if the URL is signed.

The data to be appended is transferred as an entity body.

Example

`http://pswdf009:1080/ContentServer/ContentServer.dll?append&pVersion=0045&contRep=K1&d
ocId=4B7689654E73D21197E70060B0672A3C&compId=data&Content-Length=980`

append

Data transferred in the request body is appended to the content unit of the component “data” in the specified document.

Server → Client

The server answers the request with a response. The response status code indicates the outcome of the call.

HTTP Status Code	Meaning
200 (OK)	OK, data appended
400 (bad request)	Unknown function or unknown parameter
401 (unauthorized)	Breach of security
404 (not found)	Document/component not found
409 (conflict)	Document/component not accessible
500 (Internal Server Error)	Internal error in content server

The content server must set the dates (`dateM` and `compDateM`) and the times (`timeM` and `compTimeM`) for changing components and the document.

update

Effect

One or more components of a document in the content repository are overwritten. The parameters `contRep`, `docId` and `compId` describe the component (see [Definition of Terms \[Page 60\]](#)). The function is used to modify existing documents and components and can be called once with an [HTTP-PUT \[Page 99\]](#) or [HTTP-POST \[Page 100\]](#). The function can be called once with a HTTP-PUT or POST (see below for details).

The variant `HTTP-PUT` is used to create or overwrite a component of an existing document.

The variant `HTTP-POST` (multipart/form_data) is used to bring an entire document up to date. When this variant is used, the entire document is overwritten, not only individual components.

Default

One or more components are stored in the content repository. Protection is according to the standard set on the content server.

Access mode

change (u);

Client → Server

The following parameters exist:

Parameter	Optional/mandatory	Default	Position (POST/PUT)	Sign (POST/PUT)
<code>contRep</code>	mandatory		URL/URL	X/X
<code>compId</code>	mandatory		body/URL	-/X
<code>docId</code>	mandatory		URL/URL	X/X
<code>pVersion</code>	mandatory		URL/URL	
<code>Content-Type</code>	optional		body/body	
<code>charset</code>	optional		body/body	
<code>version</code>	optional		body/body	
<code>Content-Length</code>	mandatory		body/body	
<code>accessMode</code>	s-mandatory		URL/URL	X/X

update

authId	s-mandatory		URL/URL	X/X
expiration	s-mandatory		URL/URL	X/X
secKey	optional		URL/URL	



s-mandatory means that this parameter must only be specified if the URL is signed.

The function can be executed in two ways. Either a single component can be transferred using an [HTTP-PUT \[Page 99\]](#) or an [HTTP-POST \[Page 100\]](#) in `multipart/form-data` format is used. If the former version (**HTTP-PUT**) is used, only one component at a time can be loaded onto the content server. There is no such restriction if the latter version (**HTTP-POST**) is used.

HTTP-PUT

This variant is used to create or overwrite an individual component of a document.

For further details, see the command [create \[Page 89\]](#).

HTTP-POST multipart/form-data

HTTP-POST multipart/form-data

Similarly to the `create` function, this variant of the function is used to replace a complete document with all its components in the content repository at once. Document components not already in the content repository are created if necessary. Components in the content repository that are not transferred when the `update` function is executed are considered obsolete and deleted. The structure details of the request is the same as for the function [create \[Page 90\]](#).

Server → Client

The server answers the request with a response. The response status code indicates the outcome of the call.

HTTP Status Code	Meaning
200 (OK)	OK, document(s)/component(s) changed
400 (bad request)	Unknown function or unknown parameter
401 (unauthorized)	Breach of security
404 (not found)	Document/component not found
409 (conflict)	Document/component not accessible
500 (Internal Server Error)	Internal error in content server

The content server must set the dates (`dateM`, `compDateM` and `compDateC`) and the times (`timeM`, `compTimeM` and `compTimeC`) for changing components and the document.

delete

Effect

A component or an entire document is deleted. A document to be deleted is addressed via `contRep` and `docId`. The parameters `contRep`, `docId` and `compId` identify the component to be deleted.

Default

The document, including all administrative data (document header and components) and the content, is deleted completely.

Access mode

delete (d)

Client → Server

The client sends an `HTTP-GET-Request`. The URL contains the following parameters:

Parameter	Optional/Mandatory	Default	Sign
<code>contRep</code>	mandatory		X
<code>docId</code>	mandatory		X
<code>compId</code>	optional	all components	X
<code>pVersion</code>	mandatory		
<code>accessMode</code>	s-mandatory		X
<code>authId</code>	s-mandatory		X
<code>expiration</code>	s-mandatory		X
<code>secKey</code>	optional		



s-mandatory means that this parameter must only be specified if the URL is signed.

Example

`http://pswdf009:1080/ContentServer/ContentServer.dll?delete&pVersion=0045&contRep=K1&docId=4B7689654E73D21197E70060B0672A3C&compId=data`

Component "data" is deleted in the named document.

delete

Server → Client

The server answers the request with a response. The response status code indicates the outcome of the call.

HTTP Status Code	Meaning
200 (OK)	OK, document/component(s) deleted
400 (bad request)	Unknown function or unknown parameter
401 (unauthorized)	Breach of security
404 (not found)	Document/component not found
409 (conflict)	Document/component not accessible
500 (Internal Server Error)	Internal error in content server

search

Effect

This function searches for a text pattern in the content unit of a component. The range of the search can be restricted. The search begins at the point specified by `fromOffset` and continues until the `toOffset` point. If `fromOffset > toOffset`, the function searches the component backwards.

A text pattern is found if the following conditions are met:

- if `fromOffset <= toOffset`
- the location of first character of the text found is greater than or equal to `fromOffset`
- the location of the last character of the text found is smaller than or equal to `toOffset`
- if `fromOffset >= toOffset`
- the location of the last character of the text found is less than or equal to `fromOffset`
- the location of the first character of the text found is greater than or equal to `toOffset`

The pattern contains the string searched for. The string can contain blank characters.

The number of result entries and up to `numResults` hits are returned as the result. A hit is the entry of the character position. The character position is the position of the found location in relation to the start of the document. The position of the first character of the text searched for is defined as the position of the found location, irrespective of the search direction.

Default

The pattern is searched for in the whole addressed component.

Access mode

Read (r)

Client → Server

The client sends an `HTTP-GET-Request`. The URL contains the following parameters:

Parameter	Optional/Mandatory	Default	Sign
contRep	mandatory		X
docId	mandatory		X
pattern	mandatory		
compld	mandatory		
pVersion	mandatory		

search

caseSensitive	optional	n	
fromOffset	optional	0	
toOffset	optional	-1	
numResults	optional	1	
accessMode	s-mandatory		X
authId	s-mandatory		X
expiration	s-mandatory		X
secKey	optional		



s-mandatory means that this parameter must only be specified if the URL is signed.

Example

```
http://pswdf009:1080/ContentServer/ContentServer.dll?search&pVersion=0045&contRep=K1&docId=4B7689654E73D21197E70060B0672A3C&compId=data&pattern=Manfred%20M%FC1ler&fromOffset=80
```

A search for “Manfred Mueller” is carried out in the component data of the named object from Offset=80 to the end.

Server → Client

The server answers the request with a response. The response status code indicates the outcome of the call.

HTTP Status Code	Meaning
200 (OK)	OK, component was searched
400 (bad request)	Unknown function or unknown parameter
401 (unauthorized)	Breach of security
404 (not found)	Document/component not found
409 (conflict)	Document/component not accessible
500 (internal server error)	Internal error in content server

The result of the search is the number of hits and the Offset for each hit. An ASCII string with the following structure is returned:

```
number;offset;offset;...
```

There are no blank characters between the individual characters. There is a semicolon between the values and at the end.



```
2;122;222;
```

attrSearch

attrSearch

This function is used for attribute-based searches in print lists (attribute search). It is a prerequisite of this search that a print list has a description file (`compId=descr`) as well as a data file (`compId=data`). Unlike [search \[Page 103\]](#) this is a specific search, which is carried out in the description file of a print list (`compId=descr`). Only the description file is relevant for the implementation of an attribute search.

Basic Principles

The structure of the description file is explained below with the aid of an example.



Content of a description file (extract; the periods stand for blank characters):

```
0 72 DPRL
73 0 DKEYclient.....0 3
73 0 DKEYcompany_code.....3 5
73 0 DKEYaccount_number.....8 7
73 0 DKEYcustomer_name.....15 25
73 138 DAIN00100010147119Broeselplc
211 120 DAIN001000020147129Obelixplc
...
1147 1 DEPL
```

The description file consists of a sequence of lines (**index lines**). These index lines describe attributes of a range of the relevant data file.

An index line consists of:

- Offset and Length in the data file:
Specification of the Offset and the Length of the range described (in bytes) relative to the start of the data file.
- Record type
Type of line. The record type consists of four bytes. The following record types are used:
 - DPRL
Prolog
 - DKEY
Description of attributes
 - DAIN
Value of attributes
 - DEPL
Epilog

The various record types occur in the description file in the order specified here. The fact that the DAIN lines come after the DKEY lines is of particular note.

Only DKEY and DAIN lines are decisive for the attribute search. The DKEY lines specify the attributes and how they are stored. The DAIN lines specify the attribute values.

- Parameter

Remaining content of the index line dependent on the record type.

The individual index lines are closed with linefeed (0x0A). The value for the "Offset in the data file" increases steadily within the description file.



Interpreted content of the description file (extract):

Offset and Length in the Data File	Record Type	Parameter				
0	72	DPRL				
73	0	DKEY	Client	0	3	
73	0	DKEY	Company code	3	5	
73	0	DKEY	Account number	8	7	
73	0	DKEY	Customer name	15	25	
73	138	DAIN	001	0001	01471 19	Bröse lplc
211	120	DAIN	001	0002	01471 29	Obeli xplc
...						
1147	1	DEPL				

The structure of the DKEY and DAIN lines is described in detail below.

DKEY lines (description of attributes)

Content	Length (in Bytes)
Offset in data file	Variable
Separator (space)	1
Length in data file	Variable
Separator (space)	1
Record type ("DKEY")	4
Attribute names	40

attrSearch

Offset in the DAIN line parameter	3
Length in the DAIN line parameter	3

The DKEY lines specify the names (attribute names) and the structure (Offset and Length in the DAIN line parameter) of the attributes that occur in the DAIN lines. The Offset position is counted starting with 0. Each DKEY line describes one particular attribute.

The values "Offset and Length in data file" are not relevant here.

DAIN lines (values of attributes)

Content	Length (in Bytes)
Offset in data file	Variable
Separator (space)	1
Length in data file	Variable
Separator (space)	1
Record type ("DAIN")	4
Parameter	Variable

Each DAIN line specifies the attribute value for a specific range of the data file. The DAIN line parameter consists of the attribute values corresponding to the standards in the DKEY lines. Blank characters at the end of the DAIN lines are irrelevant. If the DAIN line contains less data than is specified in the DKEY lines, the attributes must be filled with blank characters.

Here, the specifications "Offset and Length in data file" are relevant. They relate to the data file and specify the range for which the given attribute values are valid.



The content of the above example is as follows:

Description of attributes

Attribute Name	Offset in the DAIN Line Parameter	Length in the DAIN Line Parameter
Client	0	3
Company code	3	5
Account number	8	7
Customer name	15	25

Value of attributes

Offset in Data File	Length in Data File	Attribute Name	Attribute Value
73	138	Client	"001"

		Company code	"00001"
		Account number	"0147119"
		Customer name	"Bröselplc"
211	120	Client	"001"
		Company code	"00002"
		Account number	"0147129"
		Customer name	"Obelixplc"
...

Effect

This function is used for attribute-based searches in print lists. The parameters **ContRep** and **docId** describe the component. The parameter **pattern** specifies (as well as the pattern) the attribute to be searched for. The attribute is described by its Offset and Length. The pattern is made up of the Offset, followed by the character "+", followed by the Length, followed by the character "+", followed by the attribute value. If several attributes are to be searched for, the individual **patterns** should be separated by a #.

The patterns can contain any characters. Unsafe and reserved characters are coded as normal here. This is also true for the separator "#" if this occurs in the pattern (see [Coding in the URL \[Page 70\]](#)).

The result of the attribute search is the values for "Offset and Length in the data file" in the DAIN lines, that correspond to the pattern and are in the search range.

For a pattern to be found, the following conditions must be fulfilled:

- The value for the "Offset in the data file" in the DAIN line is within the search range.
- The attribute values given in the **pattern** match those in the DAIN line. This means that each attribute value specified by the **pattern** must be contained fully in the corresponding attribute value in the DAIN line.

The number of result entries (number of appropriate DAIN lines), as well as the result entries themselves (values for "Offset and Length in the data file" in the DAIN line) are returned as the result. The values are separated by a semicolon. The result entries are arranged according to the search direction. Control of the search direction is as for the [search \[Page 103\]](#) function. The number of results can be restricted by the parameter **numResults**.

Default

The pattern is searched for in the document addressed by the parameters **contRep** and **docId**. The **CompId** is not specified in the call. The function always searches in the description file (**compId=descr**).

Access mode

Read (r)

attrSearch**Client → Server**

The client sends an **HTTP-GET-Request**. The URL contains the following parameters:

Parameter	Optional/Mandatory	Default	Sign
contRep	mandatory		X
docId	mandatory		X
pattern	mandatory		
pVersion	mandatory		
caseSensitive	optional	n	
fromOffset	optional	0	
toOffset	optional	- 1	
numResults	optional	1	
accessMode	s-mandatory		X
authId	s-mandatory		X
expiration	s-mandatory		X
secKey	optional		



s-mandatory means that this parameter must only be specified if the URL is signed.

Example

```
http://pswdf009:1080/ContentServer/ContentServer.dll?attrSearch&pVersion=0045&contRep=K1&docId=361A524A3ECB5459E0000800099245EC&pattern=3+5+12345#15+25+GmbH&numResults=5
```

A search is run in a component for *12345* in the attribute with Offset 3 and Length 5 (in the example, this is the *company code*) and also for the content *GmbH* in the attribute with Offset 15 and Length 25 (in the example, this is the *customer name*). Up to 5 hits are found.



As can be seen from the example, # is not coded.

Server → Client

The server answers the request with a response. The response status code indicates the outcome of the call.

HTTP Status Code	Meaning
200 (OK)	OK, component was searched
400 (bad request)	Unknown function or unknown parameter
401 (unauthorized)	Breach of security
404 (not found)	Document/component not found
409 (conflict)	Document/component not accessible
500 (internal server error)	Internal error in content server

The result of the search is the number of hits and the Offset and Length for each hit:

number;offset;length;...



2 ; 73 ; 138 ; 211 ; 120 ;

If an attribute search cannot be carried out properly because the values in the attribute do not match the standards in the DKEY lines (for example, attribute names wrong or attribute value too long in the **pattern**), status code 400 (bad request) is returned.

If, however, nothing is found, the status code is set to 200 (OK) and 0 is returned as the result.

Administration Functions

Administration Functions

In the current HTTP Content Server interface 4.5, only two administrative functions are defined: `putCert` and `serverInfo`. Further administration functions will be defined in later versions of the interface.

putCert

Effect

The client certificate is transferred. The system identifies itself via its authenticity (`authId`).

The client certificate (see [secKey \[Page 64\]](#)) is decoded in the message body and transferred in binary format.

For reasons of security, it is recommended that after the certificate has been transferred, manual action by an administrator is necessary before access is actually allowed. This could be a public key fingerprint check or any other plausibility check.

The logon procedure therefore consists of two steps:

- Certificate is transferred and entered in a central location
- Administrator allows access via a tool

The client can only access after the second step of this procedure. After the first step, the certificate is only created.

Access mode

-

Client → Server

The client sends an `HTTP-PUT-Request`.

Parameter	Optional/Mandatory	Sign
authId	mandatory	
pVersion	mandatory	
contRep	mandatory	

The certificate is transferred in the request body, all the other parameters are transferred in the URL. The URL does not contain a `secKey`.

Server → Client

The server answers the request with a response. The response status code indicates the outcome of the call.

HTTP Status Code	Meaning
200 (OK)	OK
400 (bad request)	Unknown function or unknown parameter

putCert

406 (not acceptable)	Certificate cannot be recognized
500 (Internal Server Error)	Internal error in content server

serverInfo

Effect

The function supplies information about the status of the content server and the content repositories that it manages.

Default

The standard information is returned to the content server addressed and the content repositories that it manages. The results are given in ASCII format.

Access mode

-

Client → Server

The client sends an **HTTP-GET-Request**. The following parameters exist:

Parameter	Optional/Mandatory	Default	Sign
contRep	optional	all components	
pVersion	mandatory		
resultAs	optional	ascii	

Example

`http://pswdf009:1080/ContentServer/ContentServer.dll?serverInfo&pVersion=0045`

The example requests information about the content server and all the content repositories it manages.

Server → Client

The server answers the request with a response. The response status code indicates the outcome of the call.

HTTP Status Code	Meaning
200 (OK)	OK
400 (bad request)	Unknown function or unknown parameter
500 (internal server error)	Internal error in content server

The following information about the content server status is provided:

Keyword	Format	Meaning
---------	--------	---------

serverInfo

serverStatus		Status of content server: [running stopped error]
serverVendorId		Vendor and software version
serverVersion		Version of server
serverBuild		Build of server
serverTime	HH:MM:SS	Content server time (UTC)
serverDate	YYYY-MM-DD	Content server date (UTC)
serverStatusDescription		Text describing server status
pVersion		Content server interface version

The following information about the status of each content repository is provided:

Keyword	Format	Meaning
contRep		Content repository
contRepDescription		Text describing content repository content
contRepStatus		Status of content repository: [running stopped error]
contRepStatusDescription		Text describing content repository status



The specified parameters are mandatory. The list of parameters is designed with a view to extension.

For each function call, all information about the content server is provided. If no content repository is addressed, information on all content repositories is provided. **ContRep** can be used to limit the content repository information to a single content repository.

There are two ways of coding the results in the response body. The parameter **resultAs** controls coding.

1. **resultAs=ascii (default)**

A pure ASCII-Text is returned. The information about the content server is at the start of the string, the information about the content repositories follows. The following format is used:

- For the content server:

```
serverStatus="string";serverVendorId="string";serverTime="string";serverDate="string";serverErrorDescription="string";pVersion="0045";CRLF
```

- For each content repository:

```
contRep="string";contRepDescription="string";contRepStatus="string";pVersion="0045";CRLF
```

If no value is entered, the value remains free.

```
contRepDescription="";contRepStatus="string";...
```

The order of the key words does not matter but there must not be any blank characters. The key words (together with their values) are separated from each other by a semicolon. The corresponding values are in quotation marks.

2. **resultAs=html**

If **resultAs=html** is set, the server sends an HTML page. The structure of the HTML page is not specified and graphical elements can be used freely.

Error Codes

Error Codes

An error occurring when the function is executed is recognizable from the HTTP status code.

HTTP Status Code	Meaning	Used for
200 (OK)	OK, information/component is delivered/transferred/changed /appended/deleted	info, get, docGet, update, append, delete, putCert, search, attrSearch
201(created)	OK, component(s) created (if create) OK, (all) document(s) created (if mCreate)	create, mCreate
250 (missing documents created)	OK, all missing documents were created	mCreate
400 (bad request)	Unknown function or unknown parameter	All functions
401 (unauthorized)	Breach of security	info, get, docGet, create, update, append, delete, mCreate, search, attrSearch
403 (forbidden)	Document/component already exists	create, mCreate
404 (not found)	Document/component/content repository not found	info, get, docGet, update, append, delete, search, attrSearch
406 (not acceptable)	Certificate cannot be recognized	putCert
409 (conflict)	Document/component/administrative data is inaccessible	info, get, docGet, append, update, delete, search, attrSearch
500 (Internal Server Error)	Internal error in content server	All functions

If an error occurs, the content server must also deliver an ASCII string describing the error. The error must be entered in the header field **X-ErrorDescription**.

Appendix

Parameters and Key Words

Parameters and Key Words

A parameter appears no more than once per URL. The parameters and key words defined are listed below in alphabetical order. The data type is given in square brackets (a “string” consists of characters from the ASCII character set and an “integerstring” consists of characters from the set {0,1,2,3,4,5,6,7,8,9}): The following parameters and key words are defined:

- **accessMode** [**string**]
Access mode
- **authId** [**string**]
Client ID
- **caseSensitive** [**y|n**]
Determines whether the search is case-sensitive. Standard is **n**.
 - caseSensitive=**n**
Search is not case-sensitive
 - caseSensitive=**y**
Search is case-sensitive
- **charset** [**string**]
Describes the character set in which the component content is encoded (for example, ISO-8859-1; see also RFC 2046). Further values can be defined, but must have an **x**-placed before them. The character set is transferred as a **Content-Type** parameter.
- **compDateC** [**string**]
Date component created (UTC); format: YYYY-MM-DD
- **compDateM** [**string**]
Date component last changed (UTC); format: YYYY-MM-DD
- **compId** [**string**]
Identifies a component within a document.




Additional information for partners who already support the SAP ArchiveLink interface:

Data files for stored print lists and outgoing documents are interpreted as **compId** “data”; the corresponding description files as interpreted as **compId** “descr”. Notes have **compId** “note”.

Predefined values for components:

- “**data**” data file
- “**descr**” attribute file
- “**note**” note

- **compTimeC** [**string**]
Time component created (UTC); format: HH:MM:SS
- **compTimeM** [**string**]
Time component last changed (UTC); format: HH:MM:SS
- **compStatus** [**online|offline**]
Status of component in the content repository. Meaning:
 - **online**:
Component known and accessible
 - **offline**:
Component known and currently inaccessible
- **Content-Disposition**
The **Content-Disposition** can be transferred as an additional parameter to the **compId**, if documents are transferred as **multipart/form-data** (see also [HTTP-POST multipart/form-data \[Page 90\]](#)). In this case, the **Content-Disposition** must be transferred at the same time as the **X-compId**.

This parameter can be ignored.
- **Content-Length** [**integerstring**]
Size of body or component in bytes. The parameter **Content-Length** can occur both in the response header and in part headers.
- **Content-Type** [**string**]
Identifies the **Content-Type** of a component or a transferred document. Can occur in the response header and in part headers. With **charset**, the character set used to write the component content can be specified as the **Content-Type** parameter.
- **contRep** [**string**]
Specifies the content repository.
- **contRepDate** [**string**]
Content repository date (UTC); Format: YYYY-MM-DD.
- **contRepDescription** [**string**]
Text describing content repository content.
- **contRepErrorDescription** [**string**]
Text describing a content repository error.
- **contRepStatus** [**running|stopped|error**]
Content repository status. Meaning:
 - **running**

Parameters and Key Words

- Content repository is running
 - **stopped**
 - Content repository has been stopped
 - **error**
 - Error in content repository
- **contRepTime [string]**
 - Content repository time (UTC); format HH:MM:SS
- **contRepVendorId [string]**
 - Vendor and version of content repository software
- **dateC [string]**
 - Date document created (UTC); format: YYYY-MM-DD.
- **dateM [string]**
 - Date document last changed (UTC); format: YYYY-MM-DD
- **docID [string]**
 - Unique identifier for document header
- **docProt [string]**
 - Document protection **docProt** controls protection of a document and its information. **docProt** is a combination of the access rights **r**, **c**, **u**, or **d**. A standard that can be overwritten by the parameter is set on the content server.
 - For example, the combination **docProt=rcud** provides full protection for a document.
- **docStatus [online|offline]**
 - Status of document in the content repository. Meaning:
 - **online**
 - Document known and accessible
 - **offline**
 - Document known and inaccessible
- **expiration [string]**
 - Expiry time of a signed URL, (UTC) format: YYYYMMDDHHMMSS.
- **fromOffset [integerstring]**
 - Specifies the starting point for a search or the beginning of a byte range within the component. The default is 0. This parameter is needed to be able to read parts of the component (with print lists, for example).
- **numComps [integerstring]**
 - Number of components in a document

- **numResults** [*integerstring*]
Determines the maximum number of results (hits) a search can give.
- **pattern** [*string*]
Character pattern searched for in the free search. See the function [search \[Page 103\]](#).
The character patterns must be replaced by corresponding **escape** coding in the form **escape = % HEX HEX**.
The disallowed characters must be replaced by corresponding **escape** coding in the form **escape = \"%\" HEX HEX**.
The counterpart of **pattern** [*string*] in the attribute search is **pattern** [*integerstring+integerstring+string*].
- **pattern** [*integerstring+integerstring+string*]
Attribute pattern searched for in the attribute search. See function [attrSearch \[Page 106\]](#).
The counterpart of **pattern** [*integerstring+integerstring+string*] in the free search is **pattern** [*string*].
- **pVersion** [*string*]
Specifies the interface version. Versions 0021, 0030 and 0031 were defined for the SAP ArchiveLink interface. The HTTP Content Server interface begins with version 0045.
- **resultAs** [*string*]
Chooses the presentation form for the result of the **info** function. Standard is **ASCII**.
 - **resultAs=ascii**
Results given in ASCII format
 - **resultAs=html**
Results given in HTML format
- **retCode** [*integerstring*]
Part of the ASCII string sent by the content server to the client after an **mCreate** call. Contains the HTTP status code for the corresponding document.
- **secKey** [*string*]
Specifies an access key that can be used to check access authorization. The access key is generated by the R/3 System.
- **serverDate** [*string*]
Content server date (UTC); Format: YYYY-MM-DD.
- **serverErrorDescription** [*string*]
Text describing content server error.
- **serverStatus** [*running|stopped|error*]
Content server status. Meaning:
 - **running**

Parameters and Key Words

Content server is running

- **stopped**

Content server has been stopped

- **error**

Error in content server

- **serverTime [string]**

Content server time (UTC); format: HH:MM:SS

- **serverVendorId [string]**

Vendor and version of content server software

- **timeC [string]**

Document creation date (UTC); format: HH:MM:SS

- **timeM [string]**

Time document last changed (UTC); format: HH:MM:SS

- **toOffset [integerstring]**

Specifies the end of a byte range within the component. The default -1 means that the search should continue to the end of the component. **toOffset** has priority over a potential Content-Range.

- **version [string]**

Describes the application version used to create a document/component. The version is transferred as the **Content-Type** parameter. For some MIME-Types, version numbers are registered at IANA, so that they are used in the same way worldwide. For example, versions 2w, 4, 5 and 6 are used for application/msword.

- **serverStatusDescription [string]**

Header field in which the content server enters an explanatory text if an error occurs.

Information on Migrating Existing Archives



This section is only relevant for those partners who have supported the SAP ArchiveLink interface and now want to support the new HTTP Content Server interface.

When a component is stored, the MIME type is transferred in the field **Content-Type** in the request header, see RFC 2045 / RFC 2046. The content server holds the MIME type, but does not evaluate it. The MIME type is used as the response **Content-Type** (for the **get** function, for example).

Until now, only document classes such as ALF, FAX, DOC have been recognized in the archives. An MIME type must be determinable from the old document classes in the new interface so that documents already stored can be accessed. The MIME type is derived from the document class.

Document Structure

There are already many documents in the SAP ArchiveLink interface that consist of several components. This is particularly true of documents of class FAX, OTF or ALF. As well as one or more (FAX) data files, these documents sometimes also contain a description file (ALF) and a note file.

During migration a component ID (**compId**) must be assigned to each component. The procedure is as follows.

- The **compId** data is assigned to the data file. If several data files exist (for document class FAX, for example, if there are several pages and each page is saved separately as a TIFF file), they are assigned to the components data1, data2, In this case there is no component data.
- A description file (ALF only) is assigned to the component descr.
- Finally, the note file is assigned to the component note.

Converting from Document Classes to MIME Types

When an archive is migrated, the Mime type of the component data (or data1, data2, ...) is determined on the basis of the document class.

Conversion is shown in the following table. The order of the entries is significant. The first appropriate entry from the top is used for conversion.

Document Class	MIME Type
FAX	image/tiff
BIN	application/octet-stream
DOC	application/msword
PDF	application/pdf

Information on Migrating Existing Archives

PS	application/postscript
RTF	application/rtf
XLS	application/vnd.ms-excel
MPP	application/vnd.ms-project
PPT	application/vnd.ms-powerpoint
ALF	application/x-alf
OTF	application/x-otf
RAW	application/x-raw
REO	application/octet-stream
SCR	application/x-scr
BMP	image/bmp
GIF	image/gif
JPG	image/jpeg
PCX	image/pcx
TIFF	image/tiff
TIF	image/tiff
HTM	text/html
TXT	text/plain



For document classes not listed above, no MIME type is set.

The components descr (ALF only) and note contain the MIME type as follows:

compId	MIME Type
note	application/x-note
descr	application/x-alf-descr

Index Management Service (BC-SRV-KPR)

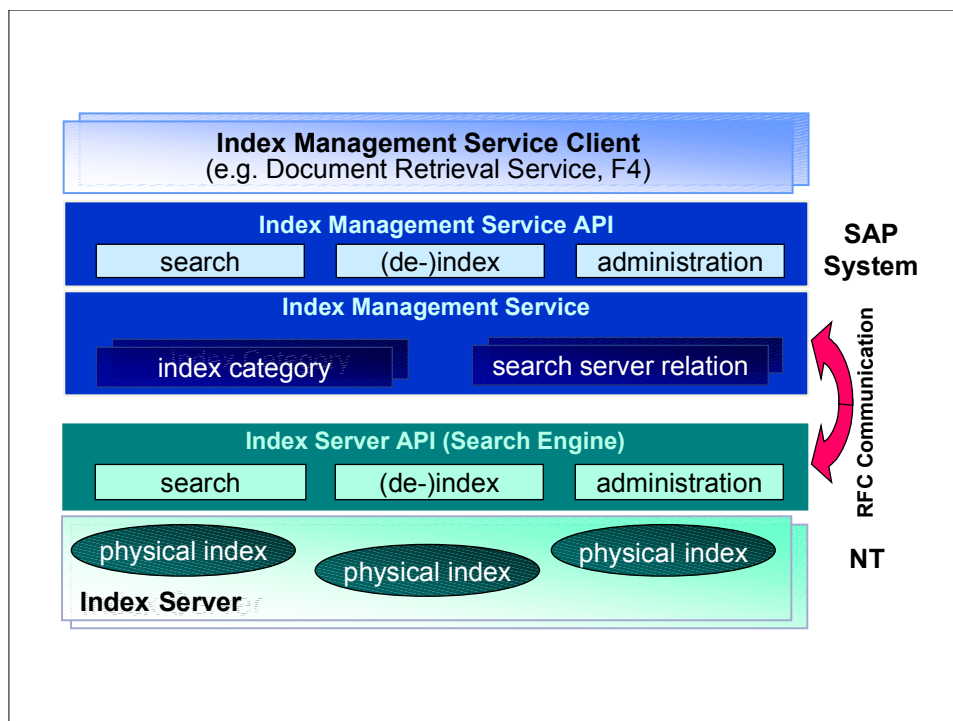
Index Management Service (BC-SRV-KPR)

Purpose

The IMS is an independent service in the [SAP Knowledge Provider \[Ext.\]](#) infrastructure. Its use is not restricted to SAP Knowledge Provider Client applications.

Integration

All SAP applications can use the Index Management Service via the ABAP programming interface to carry out full-text searches and build full-text indexes. It also enables various search engines to be integrated via a C interface, thereby enhancing text-based searches in particular. The IMS maintains the transparency of the specific attributes of the search engines for the client applications.



Architecture of the Index Management Service

Features

The IMS enables index servers (search engines) that support the interface to the SAP System to be integrated for specific applications and tasks. Using the IMS, you can integrate full-text searches in your SAP System. The Index Management Service is not dependent on the representation of the document content (for example, full-text index). The IMS provides all of the functionality required by a full-text search engine:

- Exact search
- Linguistic search

- Fuzzy search
- Phrase-based search
- Wildcard search
- Attribute search
- Natural-language search

A document is specified for the indexing process as an ASCII or binary text (for example, from the data base), a file name, or a URL. All of the standard document formats are supported.

The client applications can deploy different retrieval strategies. The service can also be used for objects that are not managed as documents.



One example of these solutions is the F4 Help.

The Index Management Service is divided into three areas:
Services for

- Administration
- Indexing/deindexing
- Search

Constraints

The first version of the IMS, in Release 4.6, will support all LATIN I languages.