

Object Reference Manual



Version 5.3 SP1
September 2005

Copyright © 1994-2005 EMC Corporation

Table of Contents

Preface	11
Chapter 1 Object Basics	13
The persistent object type	13
Persistent attributes	14
Single-valued and repeating attributes	14
Read and write attributes	15
Read-only attributes	15
Local and global attributes	15
Domains	16
Computed attributes	16
Attribute datatypes	27
Referencing attributes	28
DMCL methods	28
General syntax	29
Method scope	29
Indirect references	30
Identifiers	30
Names	31
Object type and attribute names	31
User and group names	32
Table and column names	32
Object identifiers	32
Collection identifiers	32
Type identifiers	33
Session identifiers	33
Subconnection identifiers	33
Relationships	34
User-defined relationships	34
Creating a relationship type	34
Security and relationships	35
SYSTEM security	35
PARENT and CHILD security	36
None	36
The permanent_link and copy_child attributes	36
Compatibility with prior releases	37
Defining delete behavior	38
Destroying a relationship type	39
Chapter 2 Object Reference	41
Appendix A The Documentum Views	553
The dm_queue view	553
The dm_tasks_queued view	556

	The dm_tasks_dequeued view.....	559
Appendix B	Language and Country Codes	563
Appendix C	RDBMS Tables for Documentum Types	567
	Overview	567
	The tables.....	567
	The views	568
	Returning database table information	568
	The user type tables in the Oracle RDBMS.....	568

List of Tables

Table 1–1.	The computed attributes	17
Table 1–2.	Computed attributes for lifecycles	25
Table 1–3.	The attribute datatypes	27
Table 1–4.	Effects of setting permanent_link and copy_child attributes	37
Table 1–5.	Deletion behavior when deleting a parent object.....	39
Table 1–6.	Deletion behavior when deleting a child object.....	39
Table 2–1.	Attributes defined for the ACL type.....	42
Table 2–2.	Attributes defined for the ACS config type	47
Table 2–3.	Attributes defined for the activity type.....	52
Table 2–4.	Attributes defined for the aggr domain type.....	64
Table 2–5.	Attributes defined for the alias set type	66
Table 2–6.	Attributes defined for the api config type	68
Table 2–7.	Attribute defined for the app ref type.....	80
Table 2–8.	Attribute defined for the application type.....	81
Table 2–9.	Attributes defined for the aspect relation type	84
Table 2–10.	Attributes defined for the aspect type object type.....	85
Table 2–11.	Attributes defined for the assembly type	86
Table 2–12.	Attributes defined for the audit trail type	88
Table 2–13.	Attributes defined for the audit trail acl type	94
Table 2–14.	Attributes defined for the audit trail attrs type.....	97
Table 2–15.	Attributes defined for the audit trail group type	98
Table 2–16.	Attributes defined for the auth config type	101
Table 2–17.	Attributes defined for the blob store type	102
Table 2–18.	Attribute defined for the builtin expr type.....	103
Table 2–19.	Attributes defined for the CA store type.....	104
Table 2–20.	Attributes defined for the cabinet type	107
Table 2–21.	Attributes defined for the cache config type.....	108
Table 2–22.	Attributes defined for the category type	110
Table 2–23.	Attributes defined for the category assign type	113
Table 2–24.	Attributes defined for the category class type	116
Table 2–25.	Attributes defined for the change record type.....	120
Table 2–26.	Attributes defined for the CI config type	122
Table 2–27.	Attributes defined for the comment type.....	125
Table 2–28.	Attributes defined for the completed workflow type	127
Table 2–29.	Attributes defined for the completed workitem type	131
Table 2–30.	Attributes defined for the component type	136
Table 2–31.	Attributes defined for the composite predicate type.....	137

Table 2–32.	Attributes defined for the cond expr type	138
Table 2–33.	Attributes defined for the cond ID expr type	139
Table 2–34.	Attributes defined for the connection config tType	140
Table 2–35.	Attributes defined for the containment type	143
Table 2–36.	Attributes defined for the content type	146
Table 2–37.	Attributes defined for the cryptographic key type	154
Table 2–38.	Attributes defined for the DD attr info type	156
Table 2–39.	Attributes defined for the DD common info type	162
Table 2–40.	Attributes applicable to object types and attributes	168
Table 2–41.	Attributes applicable only to object types	170
Table 2–42.	DD info attributes applicable to attributes	171
Table 2–43.	Attributes defined for the DD type info type	175
Table 2–44.	Attributes defined for the display config type	178
Table 2–45.	Attributes defined for the distributed store type	180
Table 2–46.	Attributes defined for the docbase config type	181
Table 2–47.	Attributes defined for the docbase locator type	191
Table 2–48.	Attributes defined for the docbroker locator type	194
Table 2–49.	Attributes defined for the docset type	195
Table 2–50.	Attributes defined for the docset run type	197
Table 2–51.	Attributes defined for the domain type	200
Table 2–52.	Attributes defined for the dmc dsm application type	202
Table 2–53.	Attributes defined for the dmc dsm backbone type	203
Table 2–54.	Attributes defined for the dmc dsm doc properties type	204
Table 2–55.	Attributes defined for the dmc dsm drug product type	206
Table 2–56.	Attributes defined for the dmc dsm drug substance type	207
Table 2–57.	Attributes defined for the dmc dsm excipient type	208
Table 2–58.	Attributes defined for the dmc dsm facilities equip type	209
Table 2–59.	Attributes defined for the dmc dsm indication type	210
Table 2–60.	Attributes defined for the dmc dsm safety eval type	212
Table 2–61.	Attributes defined for the dmc dsm sect doc attributes type	213
Table 2–62.	Attributes defined for the dmc dsm section type	215
Table 2–63.	Attributes defined for the dmc dsm section extension type	217
Table 2–64.	Attributes defined for the dmc dsm stf backbone type	218
Table 2–65.	Attributes defined for the dmc dsm stf section type	220
Table 2–66.	Attributes defined for the dmc dsm study attributes type	221
Table 2–67.	Attributes defined for the dmc dsm study report type	223
Table 2–68.	Attributes defined for the dmc dsm submissionrt type	224
Table 2–69.	Attributes defined for the dump object record type	225
Table 2–70.	Attributes defined for the dump record type	226
Table 2–71.	Attributes defined for the email address table type	229
Table 2–72.	Attributes defined for the email message type	231
Table 2–73.	Attribute defined for the esign template type	232
Table 2–74.	Attribute defined for the expr code type	234

Table 2-75.	Attribute defined for the expression type	235
Table 2-76.	Attributes defined for the external file store type	237
Table 2-77.	Attributes defined for the external store type.....	239
Table 2-78.	Attribute defined for the federation type	242
Table 2-79.	Attributes defined for the file store type	245
Table 2-80.	Attributes defined for the folder type.....	246
Table 2-81.	Attributes defined for the foreign key type	247
Table 2-82.	Attributes defined for the format type.....	248
Table 2-83.	Attributes defined for the FT index agent config type.....	252
Table 2-84.	Attributes defined for the FT engine config type	256
Table 2-85.	Attributes defined for the fulltext index type	257
Table 2-86.	Attributes defined for the func expr type.....	259
Table 2-87.	Attributes defined for the group type.....	262
Table 2-88.	Attributes defined for the index type	266
Table 2-89.	Attributes defined for the jar type	268
Table 2-90.	Attributes defined for the java library type	269
Table 2-91.	Attributes defined for the job type	270
Table 2-92.	Attributes defined for the job request type.....	275
Table 2-93.	Attributes defined for the job sequence type	277
Table 2-94.	Attributes defined for the key type	279
Table 2-95.	Attributes defined for the LDAP config type.....	280
Table 2-96.	Attributes defined for the linked store type	286
Table 2-97.	Attributes defined for the link record type.....	287
Table 2-98.	Attributes defined for the load object record type	290
Table 2-99.	Attributes defined for the load record type.....	292
Table 2-100.	Attributes defined for the location type	294
Table 2-101.	Attributes defined for the locator type.....	296
Table 2-102.	Attributes defined for the mail message type	297
Table 2-103.	Attributes defined for the media profile type	304
Table 2-104.	Attributes defined for the method type	305
Table 2-105.	Attributes defined for the module type	311
Table 2-106.	Attributes defined for the module config type	314
Table 2-107.	Attributes defined for the mount point type	316
Table 2-108.	Attributes defined for the network location map type	318
Table 2-109.	Attributes applicable to object type and attributes	320
Table 2-110.	Attributes applicable only to attributes.....	321
Table 2-111.	Attributes defined for the notepage type	324
Table 2-112.	Attributes defined for the other file type.....	325
Table 2-113.	Attributes defined for the output device type	326
Table 2-114.	Attributes defined for the package type.....	327
Table 2-115.	Attributes defined for the plugin type	330
Table 2-116.	Attributes defined for the policy type.....	332
Table 2-117.	Attributes defined for the procedure type.....	341

Table 2–118.	Attributes defined for the process type.....	342
Table 2–119.	Attributes defined for the public key certificate type	346
Table 2–120.	Attributes defined for the qual comp type	348
Table 2–121.	Attributes defined for the queue item type	352
Table 2–122.	Attributes defined for the readcomment type	360
Table 2–123.	Attributes defined for the reference type	361
Table 2–124.	Attributes defined for the registered type	366
Table 2–125.	Attributes defined for the registry type	368
Table 2–126.	Attributes defined for the relation type	371
Table 2–127.	Attributes defined for the relation SSA policy type.....	373
Table 2–128.	Attributes defined for the relation type object type.....	374
Table 2–129.	Attributes defined for the replica record type	378
Table 2–130.	Attributes defined for the retainer type	380
Table 2–131.	Attributes defined for the richtext type.....	385
Table 2–132.	Attributes defined for the room type	386
Table 2–133.	Attributes defined for the routecase condition type.....	389
Table 2–134.	Attributes defined for the route table type	393
Table 2–135.	Attributes defined for the rps action type	395
Table 2–136.	Attributes defined for the rps action rel type.....	396
Table 2–137.	Attributes defined for the RPS authority type	397
Table 2–138.	Attributes defined for the RPS base date type	399
Table 2–139.	Attributes defined for the RPS child strategy type.....	401
Table 2–140.	Attributes defined for the RPS condition type	403
Table 2–141.	Attributes defined for the RPS contact type	404
Table 2–142.	Attributes defined for the RPS disposition method type	405
Table 2–143.	Attributes defined for the RPS event type.....	407
Table 2–144.	Attributes defined for the rps execution rule type	409
Table 2–145.	Attributes defined for the RPS hold type	410
Table 2–146.	Attributes defined for the rps notification type	412
Table 2–147.	Attributes defined for the RPS phase rel type.....	414
Table 2–148.	Attributes defined for the RPS retainer Type.....	416
Table 2–149.	Attributes defined for the RPS retainer event rel type.....	419
Table 2–150.	Attributes defined for the RPS retention policy type.....	420
Table 2–151.	Attributes defined for the scope config type.....	423
Table 2–152.	Attributes defined for the sequence type	425
Table 2–153.	Attributes defined for the server config type	426
Table 2–154.	Attributes defined for the server locator type.....	437
Table 2–155.	Attributes defined for the session config type	441
Table 2–156.	Attributes defined for the state extension type	450
Table 2–157.	Attributes defined for the state type object type	451
Table 2–158.	Attributes defined for the SSA policy type.....	452
Table 2–159.	Attributes defined for the store type.....	453
Table 2–160.	Attributes defined for the subcontent type.....	459

Table 2–161.	General attributes of SysObjects.....	462
Table 2–162.	SysObject folder-related attributes	464
Table 2–163.	Virtual document-related attributes	465
Table 2–164.	SysObject content-related attributes	467
Table 2–165.	Web-content attributes	468
Table 2–166.	SysObject security-related attributes	470
Table 2–167.	SysObject version-related attributes	472
Table 2–168.	SysObject event-related attribute	474
Table 2–169.	SysObject lifecycle-related attributes	474
Table 2–170.	Attributes defined for the taxonomy type	475
Table 2–171.	Attributes defined for the TCF activity type.....	476
Table 2–172.	Attributes defined for the TCF activity template type.....	477
Table 2–173.	Attributes defined for the topic type	478
Table 2–174.	Attributes defined for the transition condition type.....	479
Table 2–175.	Attributes defined for the type object type.....	483
Table 2–176.	Attributes defined for the type info type.....	485
Table 2–177.	Attributes defined for the user type	488
Table 2–178.	Attributes defined for the userdata table type.....	497
Table 2–179.	Attributes defined for the validation module type.....	498
Table 2–180.	Attributes defined for the validation relation type.....	500
Table 2–181.	Attributes defined for the value assist type	502
Table 2–182.	Attributes defined for the value func type	504
Table 2–183.	Attributes defined for the value list type	505
Table 2–184.	Attributes defined for the value query type	506
Table 2–185.	Attributes defined for the vstamp type.....	508
Table 2–186.	Attributes defined for the webc config type	509
Table 2–187.	Attributes defined for the webc target type.....	515
Table 2–188.	Attributes defined for the WF attachment type	520
Table 2–189.	Attribute defined for the WF package schema type	521
Table 2–190.	Attribute defined for the wf package skill type	522
Table 2–191.	Attributes defined for the WF timer type.....	524
Table 2–192.	Attributes defined for the workflow type	526
Table 2–193.	Attributes defined for the work item type.....	531
Table 2–194.	Attributes defined for the work queue type	536
Table 2–195.	Attributes defined for the work queue doc profile type	538
Table 2–196.	Attributes defined for the work queue policy type	539
Table 2–197.	Attributes defined for the work queue user profile type	541
Table 2–198.	Attributes defined for the xfm form type.....	543
Table 2–199.	Attributes defined for the xfm instance type	545
Table 2–200.	Attributes defined for the xfm schema type	546
Table 2–201.	Attributes defined for the xml application type.....	547
Table 2–202.	Attributes defined for the xml config type	549
Table 2–203.	Attributes defined for the xml custom code type.....	550

Table 2-204.	Attributes defined for the xml style sheet type	551
Table 2-205.	Attributes defined for the xml zone type	552
Table A-1.	The dm_queue view	553
Table A-2.	The dm_tasks_queued view	556
Table A-3.	The dm_tasks_dequeued view.....	559
Table B-1.	Recommended language codes.....	563
Table B-2.	Recommended country codes.....	564
Table C-1.	Columns of the dm_user_s table	568
Table C-2.	Columns of the dm_user_r table	569

This manual is the reference manual for the EMC Documentum object hierarchy. It is a companion to *Content Server API Reference Manual*, *Content Server DQL Reference Manual*, *Content Server Fundamentals*, *Content Server Administrator's Guide*, and *Documentum Distributed Configuration Guide*.

Intended audience

This manual is written for application developers and system administrators and any others who want to build a content or work-group management application that accesses and uses repository objects. It assumes that you are familiar with the concepts of document processing, object-oriented programming, and client-server applications. It also assumes working knowledge of SQL.

Conventions

This manual uses the following conventions in the syntax descriptions and examples.

Syntax conventions

Convention	Identifies
<i>italics</i>	A variable for which you must provide a value.
[] square brackets	An optional argument that may be included only once
{ } curly braces	An optional argument that may be included multiple times

Terminology changes

Two common terms are changed in 5.3 and later documentation:

- Docbases are now called repositories, except where the term “docbase” is used in the name of an object or attribute (for example, docbase config object).

- DocBrokers are now called connection brokers.

Revision history

The following changes have been made to this document.

Revision history

Revision date	Description
September 2005	Initial publication

Object Basics

This chapter contains the general reference information about object types and attributes. The topics in this chapter are:

- [The persistent object type, page 13](#), which describes the attributes inherited by all persistent object types
- [Persistent attributes, page 14](#), which describes the characteristics of persistent attributes
- [Computed attributes, page 16](#), which lists and describes the computed attributes
- [Attribute datatypes, page 27](#), which describes the valid datatypes for attributes
- [Referencing attributes, page 28](#), which describes how to reference attributes in API methods and DQL statements
- [DMCL methods, page 28](#), which discusses how DMCL methods are referenced and their scope
- [Indirect references, page 30](#), which describes how to use an indirect reference in methods
- [Identifiers, page 30](#), which describes the various identifiers used in a repository
- [Relationships, page 34](#), which describes relationship definitions and how to create them.

The persistent object type

Documentum is an object-oriented system. All the items that a user manipulates are objects and much of the information that Content Server stores and maintains is handled as an object. The objects that are actually stored in the repository are all subtypes, directly or indirectly, of the persistent object type.

The persistent object type is an internal type has only three attributes, which it passes to all of its subtypes. These attributes are:

- `r_object_id`

The `r_object_id` attribute contains the object's identifier, its object ID. This number is generated by the server when the object is created and is unique.

- `i_vstamp`

The `i_vstamp` attribute contains a count of the number of committed transactions that have changed this object. This value is used internally to support locking and versioning.

- `i_is_replica`

The `i_is_replica` attribute indicates whether the object is a local replica of an object in a remote repository.

Persistent attributes

All persistent objects have persistent attributes whose values are stored in the repository. The persistent attributes associated with a particular object are defined by the object's type. This means that all objects of the same type have the same set of attributes. However, the values in those attributes differ from object to object.

A type's persistent attributes are a combination of attributes inherited from its supertype and attributes defined for the type. A few types, such as `dm_document`, inherit all of their attributes and have no defined attributes.

Persistent attributes are:

- Single-valued or repeating
- Read and write or read-only
- Local or global

Each persistent attribute also has a domain that describes the characteristics of the individual attribute.

Single-valued and repeating attributes

A single-valued attribute can have only one value. For example, the document object's `title` attribute is a single-valued attribute. A document can have only one title.

A repeating attribute can have more than one value. For example, the document object's `authors` attribute is a repeating attribute. A document can have more than one author.

The server stores repeating attributes in a list. Consequently, when you refer to a repeating attribute in a method call, you must specify an index value with the attribute's name. This index value identifies which specific value you want in the attribute's list of values. Index values are integer values that begin at zero and go up to $n-1$ where n is the number of values in the attribute.

Read and write attributes

Attributes that are primarily useful to users and applications are the read and write attributes. Documentum divides them into two groups:

- Attributes intended primarily for use by both users and applications.

The names of these attributes have no prefix (for example, `object_name`, `title`, `subject`, and `owner_name`).

- Attributes intended primarily for use by applications.

Although users can read and write these attributes, it is likely that application developers will keep them invisible. The names of these attributes have the prefix `a_` on their names (for example, `a_application_type` and `a_content_type`).

Read-only attributes

Read-only attributes are be read but not set by end-users or applications. These attributes are set and maintained by Content Server.

Note: Although some `r_` attributes can be set by a superuser, it is not recommended. Setting the value of a read-only attribute may result in undefined behavior.

Like read and write attributes, there are two groups of read-only attributes:

- Attributes that an application may need to read.

The names of these attributes have the prefix `r_` (for example, `r_object_type`, `r_object_id`, and `r_creation_date`).

- Attributes that applications or users will likely never need to see.

The names of these attributes have the prefix `i_` (for example, `i_reference_cnt`, `i_has_folder`, and `i_contents_id`).

Local and global attributes

This characteristic of an attribute is important only in a multi-repository distributed configuration. In a multi-repository distributed configuration, global attributes have the same values in all participating repositories. Local attributes can have different values in each repository.

A global attribute is an attribute whose value is the same all its replicas or copies of the object created by reference links. For example, the `title` attribute (in `SysObjects`) is a global attribute. If a user changes the title in a local replica or reference link for a remote

document, the action affects the source object. Content Server automatically refreshes all reference links and replica objects associated with the source object to reflect the change.

A local attribute is an attribute whose value can be different in each repository to which the object is replicated. For example, when a user links a replica of a folder to a local folder, it sets the `i_ancestor_id` attribute of the replica. This is a local attribute. The change affects only the local replica, not the source object. Neither does it affect any replicas of the folder in other repositories.

There is one exception to the behavior of local attributes. The exception occurs when a repository is part of a federation. In that case, there are four local user attributes that you can choose to manage as global attributes. Changes made to those four attributes through the governing repository, using Documentum Administrator, can be propagated to all member repositories. (Refer to [User operations, page 121](#) in the *Distributed Configuration Guide* for details.)

Most attributes are global. There are few local attributes. All user-defined attributes are global attributes.

Domains

Persistent attributes have domains. An attribute's domain defines the attribute's datatype and several other characteristics of the attribute, such as its default value or label text for it.

The datatype defines what kind of data you can legally put into the attribute. For example, if an attribute has an integer datatype, then you can only put integer values (whole numbers) into the attribute. If the datatype is a string datatype, then the attribute can only contain character strings. The maximum length of the character string is defined when the attribute is created and is part of the datatype definition. For example, an attribute might have a datatype of `char(32)` or `character(64)`. (For a complete list of the datatypes recognized by Content Server and their correspondences to RDBMS datatypes, refer to [Table 1–3, page 27](#).)

You can query the data dictionary to retrieve the characteristics defined by a domain.

Computed attributes

In addition to persistent attributes, each object has a set of computed attributes. These attributes support the API. Computed attribute values are not stored in the repository with an object's description but are computed at run time when needed.

Table 1–1, page 17 lists all computed attributes except those that are specific to lifecycles. (The lifecycle-specific computed attributes are found in Table 1–2, page 25.)

Table 1-1. The computed attributes

Attribute	Single/ repeating	Description
<code>_accessor_app_permit</code>	R	Application_permit values of the entries for an ACL. The value at each index position represents the application_permit attribute value for the entry represented by the corresponding index position in <code>_accessor_name</code> .
<code>_accessor_name</code>	R	<p>The list of users and groups for whom some level of access to the selected object is defined. This attribute has valid values only if the repository is using ACLs for security.</p> <p>This attribute reflects the current status of an object's ACL. Therefore, if you are changing an object's permissions, this value reflects those changes, regardless of whether you have saved the changes. Check the <code>_acl_ref_valid</code> attribute to determine whether the values in <code>_accessor_name</code> have been saved.</p>
<code>_accessor_permit</code>	R	<p>The permission levels assigned to each user or group defined in <code>_accessor_name</code>. The values, expressed as integers, are associated with the user or group at the corresponding index position. For example, the permission level at <code>_accessor_permit[4]</code> is assigned to the user or group specified by <code>_accessor_name[4]</code>.</p> <p>This attribute reflects the current status of an object's ACL. Therefore, if you are changing an object's permissions, this value always reflects those changes, regardless of whether you have saved them. Check the <code>_acl_ref_valid</code> attribute to determine whether the values in <code>_accessor_permit</code> have been saved.</p>

Attribute	Single/ repeating	Description
<code>_accessor_permit_type</code>	R	Permit types of the entries. The value at each index position is the permit type of the entry represented by the corresponding index position in <code>_accessor_name</code> .
<code>_accessor_xpermit</code>	R	Contains the integer value of the extended permissions assigned to each user or group returned by <code>_accessor_name</code> . The values, expressed as integers, are associated with the user or group at the corresponding index position. For example, the permission level at <code>_accessor_xpermit[4]</code> is assigned to the user or group specified by <code>_accessor_name[4]</code> .
<code>_accessor_xpermit_names</code>	R	A list of the extended permissions, in string form, assigned to each user or group defined in <code>_accessor_name</code> .
<code>_acl_ref_valid</code>	S	A Boolean attribute that indicates whether the values of <code>_accessor_name</code> and <code>_accessor_permit</code> have been saved. If they have been saved, this attribute is FALSE. If the values have not been saved, <code>_acl_ref_valid</code> is TRUE.
<code>_alias_set</code>	S	The name of the alias set object bound to an object. This attribute is applicable to SysObjects, user objects, and server config objects.
<code>_all_users_names</code>	R	A list of all users directly or indirectly contained in a group. This attribute is cached on the server and client the first time it is queried in a session. Subsequent queries against it in the session return the cached values. To ensure that the returned values are correct, you can force the server to refresh the cache by issuing a Revert method on the group object. Changing the group membership also invalidates the cache and forces a refresh.

Attribute	Single/ repeating	Description
<code>_allow_change_location</code>	S	A Boolean value indicating whether the user has the Change Location permission.
<code>_allow_change_permit</code>	S	A Boolean value indicating whether the user has the Change Permission permission.
<code>_allow_change_state</code>	S	A Boolean value indicating whether the user has the Change State permission.
<code>_allow_execute_proc</code>	S	A Boolean value indicating whether the user has the Execute Procedure permission.
<code>_allow_change_owner</code>	S	A Boolean value indicating whether the user has the Change Ownership permission.
<code>_attribute_list_values</code>	R	<p>A string attribute containing the list of audited attributes and their values recorded in an audit trail entry.</p> <p>The first index position stores the value of <code>dm_audittrail.attribute_list</code>. The remaining index positions stores the values from the <code>dmi_audittrail_attrs</code> object associated with the <code>audittrail</code> object, if any.</p> <p>The length of the attribute is the maximum length allowed by the underlying RDBMS for character data types.</p>
<code>_cached</code>	S	A Boolean attribute indicating whether the object is in the DMCL cache. It returns TRUE if the object is in the cache and FALSE if not.
<code>_changed</code>	S	<p>A Boolean attribute that indicates whether the object has been changed since it was last saved.</p> <p>The server checks the DMCL cache and if the object is not in the cache or is in the cache but hasn't been changed, server returns FALSE. TRUE is returned if the object is present in the cache and has been changed.</p>

Attribute	Single/ repeating	Description
<code>_componentID</code>	R	<p>The chronicle IDs of the component documents that make up a virtual document. The ID value at each index position is the chronicle ID for the component at that position in the virtual document.</p> <p>The zero position, <code>_containID[0]</code>, represents the chronicle ID for the virtual document. A virtual document is always considered a component of itself.</p>
<code>_containID</code>	R	<p>The object IDs of the containment objects that link the component document to its containing virtual document. The ID value at each index position represents the containment object for the component at that position in the virtual document.</p> <p>The zero position, <code>_containID[0]</code>, represents the containment object for the virtual document. A virtual document is always considered a component of itself.</p>
<code>_content_buffer</code>	S	<p>A pointer to the actual location in memory of the retrieved content file. This special attribute is only visible when you execute a <code>Getcontent</code> method. You cannot select this attribute using DQL.</p>
<code>_content_state</code>	R	<p>An integer value representing the state of a content for a given <code>SysObject</code>. State values and their meanings are:</p> <ul style="list-style-type: none"> 0, meaning available and online 1, meaning archived 2, meaning offline 3, meaning error fetching content from the repository <p>For example, if <code>_content_state[1] = 1</code>, then the first content for the object is archived.</p>
<code>_count</code>	S	<p>The number of attributes in the object</p>

Attribute	Single/ repeating	Description
<code>_current_state</code>	S	The current lifecycle state of the object.
<code>_docbase_id</code>	S	That portion of a specified object ID that corresponds to the repository ID. (Use the Get method to return the value for this computed attribute.)
<code>_dump</code>	S	A description of the object, including all attributes and their values
<code>_has_config_audit</code>	S	A Boolean attribute indicating whether the user has the privilege. T (TRUE) means the user has the privilege.
<code>_has_create_type</code>	S	A Boolean attribute indicating whether the user has the privilege. T (TRUE) means the user has the privilege.
<code>_has_create_group</code>	S	A Boolean attribute indicating whether the user has the privilege. T (TRUE) means the user has the privilege.
<code>_has_create_cabinet</code>	S	A Boolean attribute indicating whether the user has the privilege. T (TRUE) means the user has the privilege.
<code>_has_purge_audit</code>	S	A Boolean attribute indicating whether the user has the privilege. T (TRUE) means the user has the privilege.
<code>_has_superuser</code>	S	A Boolean attribute indicating whether the user has the privilege. T (TRUE) means the user has the privilege.
<code>_has_sysadmin</code>	S	A Boolean attribute indicating whether the user has the privilege. T (TRUE) means the user has the privilege.
<code>_has_view_audit</code>	S	A Boolean attribute indicating whether the user has the privilege. T (TRUE) means the user has the privilege.
<code>_id</code>	S	A unique object identifier that contains the object's <code>r_object_id</code> value

Attribute	Single/ repeating	Description
<code>_is_restricted_session</code>	S	A Boolean attribute that indicates whether the current session is a restricted session in which the only valid operation is <code>Changepassword</code> . T means that the current is such a session. F means that the session is not restricted to the change password operation.
<code>_isdeadlocked</code>	S	A Boolean attribute that indicates whether the current session was chosen as the victim of a deadlock. This attribute is reset when the next method that issues an RPC call is executed.
<code>_isnew</code>	S	A Boolean attribute that indicates whether the object is a new object, that is, whether the object has been saved in the repository.
<code>_isreplica</code>	S	For use in a distributed storage environment, this indicates whether an object is a replica (read-only copy) or the original object. This attribute is valid only for cabinets, folders, documents, notes, and smart lists.
<code>_istransactionopen</code>	S	A Boolean attribute that indicates whether an explicit transaction is open. TRUE means a transaction is open. FALSE means there is no transaction open.
<code>_lengths</code>	R	Lengths of an object's attributes. Attribute length is measured in the number of bytes required by its datatype, such as a 4-byte integer. However, this information is only useful for character string columns.
<code>_masterdocbase</code>	S	The repository ID of the repository that owns an object. Unless the object is a replica, this value is the ID of the local repository. If the object is a replica, the value is the ID of the foreign repository that owns the object. This attribute is an integer attribute.
<code>_names</code>	R	Names of an object's attributes.

Attribute	Single/ repeating	Description
_permit	S	<p>An integer number that corresponds to the permissions for an object of a specified user. If the user is unspecified in the query (for example: <code>get,c,object_id,_permit</code>), the current user is assumed.</p> <p>The permission returned represents the most permissive permission the user has for the specified object.</p> <p>A value of 0 indicates that the object is not a SysObject. Values of 2 to 7 correspond to the object-level permissions Browse through Delete, respectively. (Refer to Object-level permissions, page 383 of the <i>Content Server Administrator's Guide</i> for details about the permissions defined for objects.)</p>
_policy_name	S	Name of the lifecycle to which the object is attached.
_repeating	R	Whether attributes are single or repeating. Values are 0 for single-valued attributes and 1 for repeating attributes.
_resume_state	S	Name of the lifecycle's normal state which immediately preceded the current state of the object.
_sign_data	S	<p>An XML file that contains the attributes used to generate the hashed signature for an audit trail entry.</p> <p>The output is well-formed XML. It includes the values of the entry's associated <code>dmi_audittrail_attrs</code> object, if one exists.</p>

Attribute	Single/ repeating	Description
_status	S	<p>Level of the error (if any) resulting from a method call against an object. The level is represented as a numeric value. The values are:</p> <p>0, for informational messages (OK, for example) 1trace messages</p> <p>2, for warning messages</p> <p>3, for error messages</p> <p>4, for fatal error messages</p> <p>If two or more errors occur, only the first error's level is stored unless the subsequent error is more severe. In such cases, the number associated with the more severe error is stored. If no errors occur, this attribute contains 0 (for OK). To clear this attribute, use the Reset method.</p>
_type_id	S	The object ID of the dm_type object for the object type of the specified object. This is returned as a string value. (Use the Get method to return the value for this computed attribute.)
_type_name	S	The internal name of the object's type. (Use the Get method to return a value for this computed attribute.)
_types	R	<p>Datatypes of an object's attributes. The datatypes are specified as numbers:</p> <p>0, for Boolean</p> <p>1, for Integer</p> <p>2, for String</p> <p>3, for ID</p> <p>4, for Time</p> <p>5, for Double</p>

Attribute	Single/ repeating	Description
_typestring	S	Returns the number of supertypes in a specified object's hierarchy and type information about each supertype. (Use the Get method to return a value for this computed attribute.)
_values	R	Total number of values of an attribute (useful for repeating attributes).
_xpermit	R	Contains the integer value of the extended permissions the current user or the specified user has on the object.
_xpermit_list	S	Contains a full list of the extended permissions, in string form, currently supported by server. Note that this computed attribute returns the same list regardless the object ID.
_xpermit_names	s	Contains the list of the extended permissions, in string form, the current user or the specified user has on the object.

[Table 1–2, page 25](#) lists additional computed attributes that apply only to lifecycles (dm_policy objects).

Table 1-2. Computed attributes for lifecycles

Attribute	Single/ repeating	Description
_alias_sets	R	List of the object names of the valid alias sets for the lifecycle. The value at each index position corresponds to the alias set specified at the same index position in the alias_set_ids attribute.

Attribute	Single/ repeating	Description
<code>_entry_criteria</code>	R	<p>List of expressions that must evaluate to TRUE before an object can be promoted to a state. The value at each index position corresponds to the conditions set for the state defined at that index position.</p> <p>This computed attribute is read and write. Setting <code>_entry_criteria</code> sets the <code>entry_criteria_id</code> attribute in the policy object. Refer to Entry criteria definitions, page 282 in <i>Content Server Fundamentals</i> for details.</p>
<code>_included_types</code>	S	The list of acceptable object types for this policy, in lower case, separated by commas.
<code>_next_state</code>	R	The name of next state. Exception states and the terminal state do not have a next state.
<code>_previous_state</code>	R	The name of previous state. Exception states and the base state do not have a previous state.
<code>_state_extension_obj[[<i>index</i>]]</code>	R	<p>The object ID of the state extension object associated with a particular lifecycle state.</p> <p>The index value identifies the lifecycle state by identifying the state's state number. For example, <code>_state_extension_obj[0]</code> returns the object ID of the state extension object associated with the lifecycle state number 0.</p> <p>If an index value is not included, the default is 0.</p>
<code>_state_type</code>	R	<p>Identifies whether the state is an exception state or, if is not an exception state, it's place in the sequence of lifecycle states. The information at each index position is applicable to the state whose state number corresponds to the index position number. Valid values are:</p> <ul style="list-style-type: none"> 1, for Exception 0, for Base (first normal state) 1, for Terminal (last normal state) 2, for Step (normal between base and terminal states)

Attribute datatypes

Table 1-3, page 27 lists the Documentum datatypes, their RDBMS equivalents, their range of values, and their default values.

Table 1-3. The attribute datatypes

Documentum Attribute Datatypes	DQL Type	Oracle	DB2, Sybase, and MS SQL Server	Value Range	Default Value
DM_ INTEGER	integer	number(10)	integer	-2147483647 to +2147483647	0
DM_ BOOLEAN	boolean, bool	number(6)	integer	1 (TRUE) or 0 (FALSE)	0 (FALSE)
DM_ STRING	string, character, char	char(64) max length is 4000	varchar, char DB2: maximum length is 2000 Sybase: *refer to the footnote SQL Server: max length is 7000	0 (blank string) to maximum for RDBMS	a null string, specified ""
DM_ DOUBLE	double, float	number 1 x 10-129 to 1 x 10+129	float 1.7e-308 to 1.7e+308	specific to RDBMS	0.0
DM_ TIME	time, date	date	timestamp (DB2) datetime (MS SQL Server and Sybase)	01/01/1753 to 12/31/ 9999	current time

Document Attribute Datatypes	DQL Type	Oracle	DB2, Sybase, and MS SQL Server	Value Range	Default Value
DM_ID	ID	char(16)	char(16)	not applicable	an ID of all zeros

*The maximum length of a string attribute on a Sybase RDBMS is dependent on the configuration of the underlying database tables; specifically, on the page size of the tables and whether they are configured as APL or DOL tables. For allowed sizes for the various sizes and configurations, refer to the Sybase installation and configuration documentation.

Referencing attributes

You can reference persistent attributes in API method calls or DQL statements. Computed attributes can be referenced only in API Get method calls.

When you want to refer to an attribute in a method call, use the attribute's name. For example, the following method call sets the title attribute for a document:

```
dmAPISet("set,s0,doc_id,title","Proposed Changes to Floor Plan")
```

Similarly, you use the attribute's name when you want to reference the attribute in a DQL statement:

```
SELECT "title","subject" FROM "dm_document"  
WHERE ANY "keywords" IN ('floorplan','cubicle')
```

If the name matches a DQL reserved word, you must double-quote the name when you reference it in a method call or DQL statement.

DMCL methods

DMCL methods are API operations that you can execute against an object or objects in a repository.

General syntax

A method's command string consists of the method's name followed by a comma, followed by a comma-separated list of the method's arguments. There are no spaces between the parts of the command string:

```
method_name, argument{, argument}
```

The arguments to all methods are positional. That is, if an argument is optional and not the final argument in the argument list, you must include the comma that is its place-holder in the list whether you include the argument or not. If the argument is the final argument, it is not necessary to include the comma if you do not include the argument.

In an application, DMCL methods are executed by one of the following API functions:

- dmAPIExec
- dmAPIGet
- dmAPISet

The dmAPIExec and dmAPIGet functions take one parameter, which is the method's command string. For example, the following method returns the number of values in the authors attribute for DocA:

```
dmAPIGet("values,c,DocA_id,authors")
```

The dmAPISet function takes two parameters: the method's command string and the value you are assigning in the method. (The second parameter, the value, is typically the last argument in the syntax shown in the method's description in the *Content Server API Reference Manual*.) For example, the following method sets the title attribute of DocA:

```
dmAPISet("set,c,DocA_id,title","Office Supply Vendors")
```

Method scope

A method executes against one or more objects in a particular repository. That repository is called the method's repository scope. If you have only one repository in your enterprise, then the repository scope is always that one repository. If your enterprise has multiple repositories in a distributed configuration, then the repository scope may change from method to method.

A method's scope is determined by either a command line argument or the default repository scope. Command line arguments that determine scope are called scoping arguments and can be a repository ID, repository name, or an object ID. For example, the scope of a Checkin method is set in its command line when you specify the object that you want to check in.

Those methods that don't take a scoping argument are called default-scoped methods. The default scope is determined by the value of the `docbase_scope` attribute in the session config object or, alternatively, by a subconnection identifier specified in the method's command line. Most of the methods such as `Query` or `Execquery`, that query a repository are default scoped.

[Defining the repository scope, page 43](#) of *Content Server Fundamentals* contains instructions about setting the default scope and obtaining and using a subconnection identifier.

Indirect references

An indirect reference is a method argument in the format:

```
@object_id
```

The *object_id* is generally the object ID of a mirror object in the current repository. The @ sign directs the system to use the associated reference object to determine the object ID of the source object and the source repository. Using that information, the system opens a subconnection to the source repository (if one is not currently open) and executes the method against the source object.

For example, suppose that `DocA`, in repository Alpha, is actually a mirror object that represents `Document_A` in repository Beta. Now suppose that you log in to repository Alpha and issue the following method:

```
set,c,@DocA_id,subject,myproject
```

The indirect reference (*@DocA_id*) directs the system to go to the repository that contains the source, `Document_A` and set the subject attribute value in that document. `DocA`'s associated reference object contains the information that tells the system the name of `DocA`'s source document and where to find it.

It is not an error to specify a local object ID using the indirect reference format. The system simply resolves the reference to the local object.

Most of the methods for which an object ID is the second argument allow you to use an indirect reference for that argument. For information about an individual method, refer to the method's description in [Chapter 2, API Server Methods](#), in the *Content Server API Reference Manual*.

Identifiers

Content Server recognizes the following kinds of identifiers:

- Names (refer to [Names, page 31](#))

- Object identifiers (refer to [Object identifiers, page 32](#))
- Collection identifiers (refer to [Collection identifiers, page 32](#))
- Type identifiers (refer to [Type identifiers, page 33](#))
- Session identifiers (refer to [Session identifiers, page 33](#))
- Subconnection Identifiers (refer to [Subconnection identifiers, page 33](#))

Names

Names identify types, attributes, tables, columns, users, and groups.

Names fall into three groups, depending on which naming rules apply to them. The groups are:

- Object type and attribute names (refer to [Object type and attribute names, page 31](#))
- User and group names (refer to [User and group names, page 32](#))
- Table and column names (refer to [Table and column names, page 32](#))

Three naming rules apply to all names:

- Do not begin a name with the prefix dm. Documentum reserves this prefix for its own use.
- The name must consist of ASCII characters.
- It is good practice not to choose names that conflict with DQL reserved words. However, if a name matches a DQL reserved word, enclose the name in double quotes whenever you use it in a DQL query.

In addition to these rules, other rules apply specifically to each group of names.

Object type and attribute names

Object type and attribute names must observe the following rules:

- The maximum allowed length is 27 characters.
- The first character must be a letter. The remaining characters can be letters, digits, or underscores.
- The name cannot contain spaces or punctuation.
- The name cannot be any of the words reserved by the underlying RDBMS.

In addition, object type names cannot end in an underscore (_).

Type and attribute names are not case sensitive.

User and group names

User and group names must observe the following rules:

- The maximum allowed length is 32 characters.
- If the name is enclosed in single quotes when referenced, it can contain any character allowed in a character string literal (spaces, apostrophes, and so forth).
- If the name is enclosed in double quotes when referenced, it can be a DQL reserved word.

User and group names are case sensitive.

Table and column names

Table and column names must observe the following rules:

- The maximum length allowed by Content Server is 64 characters. The RDBMS or operating system may impose stricter limits on the name's length.
- The first character must be a letter. The remaining characters can be letters, digits, or underbars.
- The name cannot contain spaces or punctuation.
- The name cannot be any of the words reserved by the underlying RDBMS.

Table and column names may or may not be case sensitive, depending on the rules imposed by the RDBMS or operating system.

Object identifiers

Object identifiers, or object IDs, are assigned by Content Server when an object is created. You cannot assign them yourself nor can you change them. Object IDs are expressed as a 16-character string.

Collection identifiers

A collection identifier represents the non-persistent object (collection) that contains the results of a query or a method call. When the server executes the query or method that returns a collection, it places the results in a collection and returns an identifier for the collection to the user or application. The user or application must use that identifier to retrieve the results.

Collection identifiers are two characters. The first is the letter q and the second is a numeric character.

Type identifiers

A type identifier is a two-character string that appears as the first two characters of an object's object ID. Each system-defined object type has a unique type identifier. For example, the type identifier for the dm_document type is 09, and all objects of type dm_document have an object ID that begins with 09.

When you create an object type, it inherits its type identifier from the system-defined supertype. For example, any user-defined subtypes of dm_document also have a type identifier of 09.

User-defined types that have no supertype have a type identifier of 00.

Session identifiers

Session identifiers are returned by a Connect method and identify a primary session. A session identifier has the format Sn , where n is typically a number from 0 to 9. Session identifiers are used in method calls to establish the repository scope for the method.

Note: The number of sessions a single user can establish is set by the max_session_count key in the dmcl.ini file. The default number is 10.

Subconnection identifiers

Subconnection identifiers are returned by a Getconnection method (described in [Getconnection](#), page 229 in the *Content Server API Reference Manual*). They have the format $SnCx$, where Sn is a session identifier and Cx represents a subconnection established under the primary session. Typically, x can be any number from 0 to 29. Subconnection identifiers are used in method calls to establish the repository scope for the method.

Note: The number of subconnections a user can establish under a primary session is set by the max_connection_per_session key in the dmcl.ini file. The default number is 30.

Relationships

A relationship is an association between two objects in the repository. Some system-defined relationships are installed with Content Server. (For example, annotations are implemented as a system-defined relationship between a SysObject, generally a document, and a note object.) Documentum also allows you to define your own kinds of relationships.

Each kind of relationship is described by a `dm_relation_type` object. The attributes of a relation type object name the relationship, define the security that applies to instances of that relationship, and define the behavior when an object participating in an instance of that relationship is copied, versioned, or deleted.

Each instance of a particular relationship is described by a `dm_relation` object. A relation object identifies the two objects involved in the relationship and describes the type of relationship. Relation objects also have several attributes that you can use to manage and manipulate the relationship.

Both types, `dm_relation_type` and `dm_relation`, are persistent and, consequently, objects of those types have object IDs and are stored in the repository. However, you cannot version relation type or relation objects, nor are object-level permissions applied to these object types. (Security for relationships is set by the value defined for the `security_type` attribute of the relation type object. Refer to [Security and relationships, page 35](#) for more information.)

User-defined relationships

You must provide the procedures that manage and manipulate the objects in a user-defined relationship. Content Server enforces only security for user-defined relationships. If the nature of the relationship requires actions other than security management, you must write the procedures for those actions. For example, suppose you define a relationship between two documents that requires one document to be updated automatically when the other document is updated. The server does not perform this kind of action. You must write a procedure that determines when the first document is updated and then updates the second document.

Creating a relationship type

A relation type object describes a relationship. Before you can create relation objects representing instances of a relationship, the relationship must be described by a relation type object. Creating a relation type object requires Sysadmin or Superuser privileges.

To define a type of relationship:

1. Create a relation type object.
2. Set the object's `child_type`, `parent_type`, `relation_name`, and `security_type` attributes. [Security and relationships, page 35](#), describes the possible settings for `security_type`.
3. Set the `permanent_link` and `child_copy` attributes.
These attributes control how instances of the relationship are handled when the parent object is copied or versioned using a 5.3 (or higher) DFC-based application. For complete information about how these attributes control behavior and compatibility with client library versions, refer to [The permanent_link and copy_child attributes, page 36](#).
4. To define non-default delete behavior, set the `direction_kind` and `integrity_kind` attributes.
[Defining delete behavior, page 38](#) describes how the settings of these attributes affect behavior when users delete objects participating in relationships.
5. Use the Save method to save the object.

Security and relationships

Documentum manages security for all relationships, system- or user-defined.

The security level defined for a relation type object determines who can create, modify, or drop instances of that relationship. You define security by setting the `security_type` attribute for the relation type object. There are four valid settings:

- SYSTEM
- PARENT
- CHILD
- NONE

SYSTEM security

SYSTEM security means that only superusers or system administrators can directly create, modify, or drop the relation objects that represent instances of the relationship.

SYSTEM-level security does not prevent an owner of an object from destroying an object that participates in the relationship even though destroying the object automatically destroys all relation objects associated with the object. The object owner is not required to be a superuser or system administrator.

PARENT and CHILD security

PARENT and CHILD security levels mean that the permissions to create, modify, or drop instances of a relationship depend on the object type of either the parent or child, whichever is specified. If you specify PARENT, the server uses the object type specified as the parent in the relationship to determine the permissions. If you specify CHILD, the server uses the object type specified as the child to determine the permissions. In either case, the rules that the server applies are the same:

- If the object type is `dm_user` or `dm_group`, only a superuser or system administrator can create, modify, or drop instances of the relationship.
- If the object type is `dm_sysobject` or a `SysObject` subtype, a user must have at least `Relate` permission on the object to create, modify, or drop instances of the relationship.
- If the object type is a persistent type that is not covered by either of these two rules, security is not enforced. For example, objects of type `dm_store` fall into this category.

None

NONE security means that any user can create, modify, or drop instances of the relationship. If you do not want the server to enforce security for a particular type of relationship, set the `security_type` attribute for the relation type object to `NONE`.

The `permanent_link` and `copy_child` attributes

The `permanent_link` and `copy_child` attributes control how instances of the relationship are handled by a 5.3 DFC-based application when the parent object is copied or versioned. They do not effect behavior if the application is based on pre-5.3 DFC or any version of the DMCL. (For more details, refer to [Compatibility with prior releases, page 37](#).)

The `permanent_link` attribute determines whether the relationship is maintained when the parent is copied or versioned. The attribute is `T` (`TRUE`) by default, meaning that the relationship is maintained. The `copy_child` attribute is used only when `permanent_link` is true and when the parent is copied. `copy_child` is not used if the parent is versioned.

If `permanent_link` is true and the parent is copied, then `copy_child` is used to determine whether the relationship with the new parent is established with the old child or a new copy of the child. [Table 1-4, page 37](#), describes the behavior for all combinations of `permanent_link` and `copy_child`.

Table 1-4. Effects of setting permanent_link and copy_child attributes

	permanent_link=T	permanent_link=F
copy_child=0	The child object is not copied. A new relation object is created that associates the new copy or version of the parent object with the old child.	The relationship is not maintained when the parent is copied or versioned.
copy_child=1	<p>If the parent is copied (using saveasnew), the child object is copied. A new relation object is created that associates the new copy or version of the parent object with the new copy of the child object.</p> <p>If the parent object is versioned, the child object is not copied and a new relation object is created that associates the new version of the parent object with the old child.</p>	The relationship is not maintained when the parent is copied or versioned.

Compatibility with prior releases

The `dm_relation_type.permanent_link` and `dm_relation_type.copy_child` attributes are recognized and supported only by DFC 5.3. The DMCL and pre-5.3 versions of DFC do not recognize or support these attributes.

The DMCL and pre-5.3 versions of DFC will continue to use `dm_relation.permanent_link` to determine whether to maintain an instance of a relation when the parent object is copied or versioned.

In DFC 5.3, the value of `dm_relation_type.permanent_link` is the default value for the `dm_relation.permanent_link` attribute. Consequently, if a user or application creates an instance of a relationship using a 5.3 DFC-based application and does not set the `dm_relation.permanent_link` attribute, the DFC sets the attribute's value to the value of the `dm_relation_type.permanent_link` attribute by default. This ensures that the relationship is correctly handled whether the parent object is manipulated by a DFC 5.3-based application, the DMCL, or a pre-5.3 DFC application.

Defining delete behavior

How Content Server behaves when a user deletes an object that participates in a relationship depends on how the `direction_kind` and `integrity_kind` attributes are set in the relationship's definition (the `dm_relation_type` object). These attributes are integer attributes.

The `direction_kind` attribute defines the direction of the relationship. Valid values are:

Value	Meaning
0	From the parent to the child (default)
1	From the child to the parent
2	Bidirectional (the objects are treated as siblings)

The `integrity_kind` attribute defines what deletion rule is applied to objects participating in instances of the relationship. The possible values are:

Value	Meaning
0	Allow delete (default)
1	Restrict delete
2	Cascade delete

The allow delete option means that a user with appropriate permissions can delete an object referenced as parent or child in the relationship without restriction. All relation objects that reference the object are also deleted.

The restrict delete option means that an object participating in the relationship cannot be deleted until the relation object is deleted. You must destroy the relation object before you can delete the participating object.

The cascade delete option means that when a user deletes an object participating in a relationship instance, Content Server may also destroy the participating partner in the relationship, depending on the direction of the relationship. For example, if the relationship is parent to child and a user deletes the parent, Content Server automatically removes the child also. The deletions may cascade to other relationships in which the partner is participating.

[Table 1–5, page 39](#) describes the behavior for all combinations of `direction_kind` and `integrity_kind` when a parent object is deleted from the repository.

Table 1-5. Deletion behavior when deleting a parent object

direction_kind	integrity_kind		
	0 (allow delete)	1 (restrict delete)	2 (cascade delete)
0 (parent to child)	Deletion succeeds; child object is not deleted.	Deletion fails	Deletion succeeds; child object is also deleted.
1 (child to parent)	Deletion succeeds; child object is not deleted.	Deletion succeeds; child object is not deleted.	Deletion succeeds; child object is not deleted.
2 (bidirectional -siblings)	Deletion succeeds; child object is not deleted.	Deletion fails	Deletion succeeds; child object is also deleted.

Table 1-6, page 39 describes the behavior for all combinations of direction_kind and integrity_kind when a child object is deleted from the repository.

Table 1-6. Deletion behavior when deleting a child object

direction_kind	integrity_kind		
	0 (allow delete)	1 (restrict delete)	2 (cascade delete)
0 (parent to child)	Deletion succeeds; parent object is not deleted.	Deletion succeeds; parent object is not deleted.	Deletion succeeds; parent object is not deleted.
1 (child to parent)	Deletion succeeds; parent object is not deleted.	Deletion fails	Deletion succeeds; parent object is also deleted.
2 (bidirectional -siblings)	Deletion succeeds; parent object is not deleted.	Deletion fails	Deletion succeeds; parent object is also deleted.

Destroying a relationship type

To remove a particular kind of relationship from your repository, you destroy the relationship type object that describes that relationship. Destroying a relationship type object requires either Superuser or Sysadmin privilege. Use the Destroy method.

You cannot destroy a relation type object if there are relation objects that reference that relationship. You must first destroy the relation objects.

Object Reference

This chapter contains the reference information for object types recognized by Content Server. Most of the types listed in this chapter are direct or indirect subtypes of the persistent object type. The persistent object type has three attributes that it passes to all of its subtypes:

- `r_object_id`

The `r_object_id` attribute contains a 16-character hexadecimal string that is assigned by the system when an object is created. This value uniquely identifies the object in the repository.

- `i_vstamp`

The `i_vstamp` attribute contains an integer value that represents the number of committed transactions that have changed an object. This value is used for versioning, as part of the locking mechanism, to ensure that one user does not overwrite the changes made by another.

- `i_is_replica`

The `i_is_replica` attribute indicates whether the object is a local replica of an object in a remote repository.

The non-persistent types described in this chapter are not stored in the repository. Consequently, they do not have object IDs or `i_vstamp` values, and they are never replicated. They are objects that are created and used during a session and vanish when the session is terminated.

This chapter lists the attributes defined for the persistent and non-persistent types in Content Server. For the persistent types, the information also includes the type's supertype, subtypes, and internal name.

ACL

Purpose An ACL object represents an Access Control List.

Implementation

Supertype: Persistent Object

Subtypes: None

Internal name: dm_acl

Object type tag: 45

For SysObjects, the entries in the repeating attributes control who can access the object to which the ACL is attached. If the repository security mode is set to acl, Content Server enforces the permissions granted by the ACL entries.

[Table 2-1, page 42](#) lists the attributes defined for the type.

Table 2-1. Attributes defined for the ACL type

Attribute	Datatype	Single/ repeating	Description
acl_class	integer	S	Indicates whether the ACL is a regular ACL, a template, an instance of a template, or a public ACL. Valid values are: 0, Regular ACL 1, Template ACL 2, Template instance 3, Public ACL
description	string(128)	S	User-defined description of the ACL.
globally_managed	Boolean	S	Indicates whether the acl object is managed globally or locally. The default is FALSE, meaning that it is locally managed.

Attribute	Datatype	Single/ repeating	Description
i_has_access_restrictions	Boolean	S	Whether the ACL has one or more entries with a permit_type value of 2.
i_has_required_groups	Boolean	S	Whether the ACL has one or more entries with a permit_type value of 3.
i_has_required_group_set	Boolean	S	Whether the ACL has one or more entries with a permit_type value of 4.
object_name	string(32)	S	Names the ACL object. The name, if provided, must be unique among the ACLs in the repository. This attribute can be NULL.
owner_name	string(32)	S	Identifies the owner of the ACL. This will be the user who created the ACL or, for system-level ACLs, the name of the repository owner or the alias dm_dbo.
r_accessor_name	string(32)	R	List of those for whom access to the ACL's attached object is defined. Valid entries in the list are individual user names, group names, and aliases. This must be or resolve to a group name if the r_permit_type is RequiredGroup or RequiredGroupSet.

Attribute	Datatype	Single/ repeating	Description
r_accessor_permit	integer	R	<p>Identifies the access level granted to the user or group at the corresponding index level. Valid values are:</p> <ul style="list-style-type: none"> 0, for NULL, 1, for None 2, for Browse 3, for Read 4, for Relate 5, for Version 6, for Write 7, for Delete <p>Refer to Object-level permissions, page 383 of the <i>Content Server Administrator's Guide</i> for more information about these levels.</p>
r_accessor_xpermit	integer	R	<p>Identifies the extended permission level granted to the user or group at the corresponding index level.</p> <p>Extended permissions are:</p> <ul style="list-style-type: none"> execute_proc change_location change_state change_permit change_owner delete_object

Attribute	Datatype	Single/ repeating	Description
r_application_permit	string(128)	R	<p>Specifies a permission recognized by an application. An application permit is not recognized or enforced by Content Server. The application is responsible for enforcing the permission.</p> <p>The permission defined at a particular index position is applicable to the user or group identified in the corresponding position in r_accessor_name.</p>
r_is_group	Boolean	R	Indicates whether the r_accessor_names at the corresponding index levels are individual users or groups.
r_has_events	Boolean	S	Whether someone has registered the ACL for auditing.
r_is_internal	Boolean	S	Indicates whether the ACL was created explicitly by a user or implicitly by the server.
r_permit_type	integer	R	<p>Identifies the permit type for the entry. Valid values are:</p> <p>0, meaning AccessPermit</p> <p>1, meaning ExtendedPermit</p> <p>2, meaning ApplicationPermit</p> <p>3, meaning AccessRestriction</p> <p>4, meaning ExtendedRestriction</p> <p>5, meaning ApplicationRestriction</p> <p>6, meaning RequiredGroup</p> <p>7, meaning RequiredGroupSet</p>

Attribute	Datatype	Single/ repeating	Description
			To set this to either 6 or 7, the value in <code>r_accessor_name</code> at the corresponding index position must be a group name.
			The default value is 0.

ACS Config

Purpose Stores the configuration information for an ACS Server.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_acs_config
 Object type tag: 08

An ACS config object describes the configuration of an Accelerated Content Services server (ACS). One ACS server is installed with each Content Server installed. The installation automatically creates the associated ACS config object when the ACS server is installed. To change or modify the settings, use Documentum Administrator.

Table 2–2, page 47, lists the attributes defined for the type.

Table 2-2. Attributes defined for the ACS config type

Attribute Name	Datatype	Single/repeating	Description
acs_base_url	string(240)	R	Base URL to use when accessing the ACS using the protocol identified at the corresponding index position in acs_supported_protocol.
acs_network_locations	string(80)	R	The network identifiers of the network locations served by the ACS.
acs_rw_capability	integer	S	Defines the read and write capabilities of this ACS. Valid values are: 0, meaning unconfigured 1, meaning read 2, meaning read and surrogate get

Attribute Name	Datatype	Single/repeating	Description
acs_supported_protocol	string(6)	R	Set of network protocols supported by the ACS. Values in this attribute must be an all lowercase. For example: http or https.
config_type	integer	S	Whether the accessible storage areas and defined projection targets are configured in the server config object or this acs config object. Valid values are: 1, meaning use the configuration values in the server config object (identified in srv_config_id) 2, meaning use the configuration values in this acs config object (defined in the near_stores and projection-related attributes)
is_cache_acs	Boolean	S	T (TRUE) if this acs config object represents a BOCS server, capable of caching content. The value is F (FALSE) for ACS servers.
near_stores	string(32)	R	Storage areas that are accessible for this ACS server The default is a single blank.

Attribute Name	Datatype	Single/repeating	Description
projection_enable	Boolean	R	Indicates whether projection to the connection broker specified at the corresponding index position in projection_targets is enabled.
projection_netloc_enable	Boolean	R	Indicates whether the server is allowed to service users in the network location specified at the corresponding index position in projection_netloc_ident.
projection_netloc_ident	string(80)	R	Names of the network locations that this server can service. The corresponding index position in projection_netloc_enable must be set to T (TRUE) to allow the server to actually handle requests from users in a particular network location specified in this attribute.
projection_ports	integer	R	Identifies the port on which the connection broker is listening. The value at each index position is matched to the connection broker specified at the corresponding level in projection_targets. The default is 1489.
projection_proxval	integer	R	Proximity values representing the ACS server's proximity to the network locations identified at the corresponding index position in projection_netloc_ident.

Attribute Name	Datatype	Single/repeating	Description
projection_targets	string(80)	R	Names of the host machines on which the connection brokers reside.
supported_compression_mode	integer	R	Specifies whether the ACS supports compression for the storage type identified in the corresponding index position of supported_store_types. Valid values are: 0, meaning compression is not supported 1, meaning compression is supported
supported_crypto_mode	integer	R	Specifies whether the ACS supports encryption for the storage type identified in the corresponding index position of supported_store_types. Valid values are: 0, meaning encryption is not supported 1, meaning encryption is supported
supported_store_types	integer	R	Set of storage area types supported by the ACS. Valid values are: <ul style="list-style-type: none"> • 1, for file store • 3, for linked store • 4, for distributed store • 5, for blob store • 7, for external store • 8, for external file store • 9, for external URL store • 10, for external free store • 11, for ca store
srv_config_id	ID	S	Object ID of the server config object of the ACS

Attribute Name	Datatype	Single/repeating	Description
			server's associated Content Server.

Activity

Purpose An activity object defines a workflow activity.

Implementation

Supertype: SysObject
Subtypes: None
Internal name: dm_activity
Object type tag: 4c

The attributes in an activity object define who performs the activity and the packages and work items generated from the activity.

[Table 2-3, page 52](#) lists the attributes defined for the activity type.

Table 2-3. Attributes defined for the activity type

Attribute	Datatype	Single/ repeating	Description
control_flag	integer	S	Controls who receives a task if automatic delegation fails. Valid values are: 0, meaning that the task is reassigned to the workflow supervisor 1, meaning the task is reassigned to the original task performer.

Attribute	Datatype	Single/ repeating	Description
exec_err_handling	integer	S	<p>Indicates how to handle execution failure for a runtime instance of this activity. Valid values are:</p> <p>0, meaning stop if the activity fails 1, meaning proceed with the transition even if the activity fails</p> <p>The default is 0.</p>
exec_method_id	ID	S	<p>Contains the ID of the dm_method object that represents the application. This is required if exec_type is automatic.</p>
exec_save_results	Boolean	S	<p>Indicates whether to save results generated by the application. The default is F (FALSE).</p> <p>If set to T (TRUE), the server does not verify the execution results and assumes that the method executed properly. Execution results are saved for later verification in the object whose ID is recorded in the r_exec_result_id attribute of the work item.</p>

Attribute	Datatype	Single/ repeating	Description
exec_time_out	integer	S	<p>Defines how long an automatic activity's method is allowed to execute before timing out.</p> <p>You must set this to a positive number if the activity is an automatic activity (exec_type = 1). The value is interpreted in seconds.</p> <p>This is not used for manual activities.</p>
exec_type	integer	S	<p>Indicates the execution type. Valid values are:</p> <p>0, for manual 1, for automatic</p>
is_private	Boolean	S	<p>Indicates whether the definition is private or public. The default is F (private).</p>
performer_flag	integer	S	<p>Indicates whether delegation or extension is allowed. Valid values are:</p> <p>0, No delegation or extension 1, Allow delegation 2, Allow extension 3, Allow both delegation and extension</p> <p>The default is 0.</p>

Attribute	Datatype	Single/ repeating	Description
performer_name	string(66)	R	<p>Identifies the activity's performer or performers. Valid values are a user name, a group name, the string <code>dm_world</code>, representing all users in the repository, a work queue name, or an alias. The alias format is</p> <pre> %[alias_set_name.]alias_name </pre> <p>Note: Do not specify a user or group name that begins with <code>%</code>. The server interprets that as an alias.</p>
performer_type	integer	S	<p>Identifies the category of the activity performer. Valid values are:</p> <ul style="list-style-type: none"> 0, Workflow supervisor 1, repository owner 2, Last performer 3, A user 4, All members in a group 5, Any user in a group 6, The group member who is least loaded 8, Some members of a group or some users in the repository 9, Some members of a group or some users in the repository sequentially 10, A user from a work queue

Attribute	Datatype	Single/ repeating	Description
post_timer	integer	S	<p>Defines the length of time, in hours, in which the activity must be completed after it is started.</p> <p>The default value is 0, which means the timer is not in use.</p> <p>This attribute is only set when Workflow Manager is used to create a post-timer. Using Business Process Manager to define a post-timer sets the <code>post_timer_increment</code> attribute instead.</p>
post_timer_action	ID	R	<p>Object IDs of the module config objects that identify the actions associated with the timer's firing. The ID at a particular index position represent the action associated with the triggering associated with the corresponding index position in <code>post_timer_increment</code>.</p>
post_timer_increment	integer	R	<p>Value in the first index position defines a time interval after which the timer is triggered if the activity has not been completed. The interval is expressed in minutes and is counted from the start of the activity.</p> <p>Values in the following index positions define intervals for second and subsequent triggerings of the timer if the activity has not been completed. The intervals in these index positions are counted from the end of the previous interval.</p>

Attribute	Datatype	Single/ repeating	Description
post_timer_repeat_ last	Boolean	S	Whether to repeat the last action specified in the post_timer_action attribute.
pre_timer	integer	S	<p>Defines the length of time, in hours, in which the activity must be started after the workflow starts.</p> <p>The default value is 0, which means the timer is not in use.</p> <p>This attribute is only set when Workflow Manager is used to create a pre-timer. Using Business Process Manager to define a pre-timer sets the pre_timer_increment attribute instead.</p>
pre_timer_action	ID	R	Object IDs of the module config objects that identify the actions associated with the timer's firing. The ID at a particular index position represent the action associated with the triggering associated with the corresponding index position in pre_timer_increment.
pre_timer_ increment	integer	R	<p>Value in the first index position defines a time interval after which the timer is triggered if the activity has not been started. The interval is expressed in minutes and is counted from the start of the workflow.</p> <p>Values in the following index positions define intervals for second and subsequent triggerings of the timer if the activity has not been started. The intervals in these</p>

Attribute	Datatype	Single/ repeating	Description
			index positions are counted from the end of the previous interval.
pre_timer_repeat_last	Boolean	S	Whether to repeat the last action specified in the pre_timer_action attribute.
r_condition_id	ID	S	Contains the object ID of a compound condition (a dm_cond_expr object).
r_condition_name	string(16)	R	Contains the names of route cases.
r_condition_port	string(16)	R	Contains the output port names.
			The ports are a particular index position are the ports associated with the route case condition at the corresponding index position in r_condition_name.
r_definition_state	integer	S	Identifies the state of this activity definition. Valid values are: 0, Draft 1, Validated 2, Installed
r_package_flag	integer	R	Indicates whether the package is visible and whether it is allowed to be empty. Valid values are: 0, meaning the package is invisible but cannot be empty 1, meaning the package is visible and cannot be empty 2, meaning the package is invisible but may be empty 3, meaning the package is visible and may be empty The default is 1.
r_package_id	ID	R	Contains the ID of a single package component. Not

Attribute	Datatype	Single/ repeating	Description
			used if the package has more than one component.
r_package_label	string(32)	R	Contains a version label for the package.
r_package_name	string(16)	R	Contains the package name. The package name must be unique within the scope of the containing port.
r_package_oprtn	string(64)	R	Contains the operation to be performed by performers.
r_package_type	string(40)	R	Contains the type name of the package object.
r_port_name	string(16)	R	Contains the unique identifier of a port.
r_port_type	string(8)	R	Specifies the port type, one of: INPUT, OUTPUT, or REVERT.
r_predicate_id	ID	R	Object ID of a persistent predicate, to be evaluated for the corresponding route case.
repeatable_invoke	Boolean	S	Indicates whether this activity can be included multiple times or only once within a process definition. The default value is T (TRUE), meaning the activity may be included multiple times.
resolve_pkg_name	string(16)	S	Contains the name of the package whose associated alias set will be used to resolve the activity's performer alias. This attribute is used only if resolve_type is set to 1. If resolve_type is 1 and resolve_pkg_name is not set, then all incoming packages are checked in the order in which they are defined.

Attribute	Datatype	Single/ repeating	Description
resolve_type	integer	S	<p>Defines the resolution algorithm for the performer aliases. Valid values are:</p> <p>0, Normal resolution path (default) 1, Use alias set associated with incoming package 2, Use alias set associated with user or group</p>
sign_off_required	Boolean	S	<p>Whether the activity requires a signoff. If this is set to T, the user or application must execute a Signoff method to complete the activity.</p> <p>The default is F (FALSE).</p>
task_subject	string(255)	S	<p>A user-defined string describing the task generated for this activity. The string may contain key phrases that are interpreted at runtime. The key phrases have the format:</p> <p style="text-align: center;"><i>\$type.attribute</i></p> <p>Type may be one of: dm_workflow dmi_workitem dmi_queue_item dmi_package dmi_package[x]</p> <p>If type is dm_workflow, dmi_workitem or either package option, you can specify any attribute of the type.</p> <p>If type is dmi_queue_item, you can specify any queue item attribute except task_subject.</p>

Attribute	Datatype	Single/ repeating	Description
transition_eval_cnt	integer	S	<p>Defines how many tasks must be completed to trigger activity completion.</p> <p>This attribute is not valid for activities with a performer type of 9, Some Users Sequentially.</p> <p>Valid values are 0 and any positive integer. Setting this to 0 means that all tasks must be completed before the activity is completed. The default value is 0.</p>
transition_flag	integer	S	<p>Determines which output links are actually triggered when an activity completes. This value is used only if transition_type is set to 1.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> • 0, meaning trigger all selected output links. • 1, Trigger only revert links if any are selected and ignore forward links. If there are no revert links selected, trigger selected forward links. • 2, Trigger only forward links if any are selected and ignore any selected revert links. If there are no forward links selected, trigger revert links. • 3, meaning trigger a revert link immediately if one is selected • 4, meaning trigger a forward link immediately if one is selected.

Attribute	Datatype	Single/ repeating	Description
transition_max_ output_cnt	integer	S	<p>Defines the maximum number of output ports that a performer can select.</p> <p>This attribute is valid only when transition_type is set to 1.</p> <p>A valid value is 0 or any positive integer. 0 means that there are no limits on the number of output ports the user can select. The default is 0.</p>
transition_type	integer	S	<p>Specifies the type of the transition condition. Valid values are:</p> <p>0, Prescribed (Packages are forwarded to output ports when the post-condition is met.)</p> <p>1, Manual (Activity performers specify the output ports with the Setoutput method.)</p> <p>2, Automatic (Conditional routing logic computes the output ports.)</p> <p>The default value is 0.</p>

Attribute	Datatype	Single/ repeating	Description
trigger_event	string(64)	S	Contains the expected event type that can trigger this activity instance.
trigger_threshold	integer	S	Indicates the threshold of the trigger condition; any number between 1 and the total number of input ports. The default value is 0. 0 is only allowed for begin activities, which have no input ports.

Aggr Domain

Purpose An aggr domain object describes the set of attributes in an object type for which data dictionary information is defined.

Implementation

Supertype: Domain
 Subtypes: None
 Internal name: dm_aggr_domain
 Object type tag: 51

Aggr domain objects are created and managed by the server. Users cannot create an aggr domain object.

[Table 2-4, page 64](#), lists the attributes defined for the type.

Table 2-4. Attributes defined for the aggr domain type

Attribute	Datatype	Single/ repeating	Description
attr_domain_id	ID	R	Object IDs of the domains that describe the attributes.
attr_domain_name	string(32)	R	Currently unused.
attr_name	string(32)	R	The names of attributes that have data dictionary information in their definitions.
code_objects	ID	R	Object IDs of the expr code objects holding source code that implements the expressions associated with the type, if any. Currently, only one value is allowed in this attribute.

Attribute	Datatype	Single/ repeating	Description
i_dd_flags	integer	R	Used internally.
type_name	string(32)	S	Name of the object type to which the attributes belong. This value is NULL for lifecycle state or other non-type overrides.

Alias Set

Purpose Records aliases and their actual values.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal Name: dm_alias_set
 Object type tag: 66

An alias set object contains one or more alias names that can be used as placeholders in template ACLs, in certain SysObject attributes, in workflow definitions, and in Link and Unlink methods. Any user can create alias sets.

[Table 2-5, page 66](#) lists the attributes defined for the type.

Table 2-5. Attributes defined for the alias set type

Attribute	Datatype	Single/ repeating	Description
alias_category	integer	R	Identifies the category of the value defined at the corresponding index position in alias_value. Valid values and their associated categories are: 0, Unknown 1, User 2Group 3, User or Group 4, Cabinet path 5, Folder path 6, ACL name
alias_description	string(255)	R	User-defined description of the alias name at the corresponding index position in alias_name.
alias_name	string(32)	R	List of alias for the alias set. An alias_name value cannot contain a period (.).

Attribute	Datatype	Single/ repeating	Description
alias_usr_category	integer	R	User-defined category values for the values in the alias_value attribute.
alias_value	string(255)	R	The actual user or group name associated with the alias that appears at the corresponding index level in alias_name.
object_description	string(128)	S	User-defined description of the alias set.
object_name	string(32)	S	Name of the alias set. The name must be unique among all alias set objects in the repository and cannot contain a period (.) or be composed entirely of numerical digits.
owner_name	string(32)	S	Name of the user who owns the alias set.

Api Config

Purpose An api config object describes the configuration of an API session.

Implementation

The api config object type is a non-persistent type.

Most values in an api config object are taken from the values in the dmcl.ini file used by the client establishing the session. Api config objects are accessed through the Get and Set methods, using the object's alias, apiconfig. This object is intended for use by system administrators.

Table 2–6, page 68, lists the attributes defined for the type.

Table 2-6. Attributes defined for the api config type

Attribute	Datatype	Single/ repeating	Description
application_code	string(32)	R	Identifies, by application, the application-controlled objects that this session can modify. If this is NULL, the session cannot modify any application-controlled objects. Application codes can contain only alphanumeric characters and the underscore character. They cannot contain spaces nor can they start with the characters dm_. Codes beginning with dm_ are reserved for use by Documentum. This attribute cannot be set through the dmcl.ini file.
auto_request_forward	Boolean	S	Whether to forward a connection request to secondary connection

Attribute	Datatype	Single/ repeating	Description
backup_host	string(128)	R	brokers named in the dmcl.ini file. The default is T. Identifies the host machine on which the backup connection broker resides. This has a value only if a backup connection broker is defined.
backup_port	integer	R	Defines the port number which the backup connection broker is using. This has a value only if a backup connection broker is defined.
backup_protocol	string(32)	R	Names the protocol in use by the backup connection broker. This has a value only if a backup connection broker is defined.
backup_service	string(32)	S	Contains the service name of the backup connection broker. This has a value only if a backup connection broker is defined.
backup_timeout	integer	R	Defines how long a client waits for a response from the backup connection broker before forwarding the request to another backup connection broker. This has a value only if a backup connection broker is defined.
batch_hint_size	integer	S	Defines the number of rows returned to Content Server by the RDBMS in a single call to the RDBMS. The default is 20.

Attribute	Datatype	Single/ repeating	Description
block_during_rpc	Boolean	S	Determines whether user can work in another application while a client is communicating with Content Server. The default is TRUE.
cache_queries	Boolean	S	Indicates whether the session is enabled for query caching. The default value is FALSE. This attribute is superceded by client_persistent_caching. Setting this attribute to T is only effective if client_persistent_caching is set to F.
client_cache_size	integer	S	Defines the size of the client cache.
client_cache_write_interval	integer	S	Controls how often the client persistent caches are written to disk. The default value is 60 minutes.
client_codepage	string(64)	S	Identifies the preferred code page for all repository sessions started in the API session. Valid values are: US-ASCII UTF-8 ISO_8859-1 Shift_JIS EUC-JP EUC-KR

Attribute	Datatype	Single/ repeating	Description
client_locale	string(32)	S	Identifies the preferred locale for all repository session started in the API session. Valid values are: en, for English es, for Spanish de, for German fr, for French it, for Italian ja, for Japanese ko, for Korean
client_os_codepage	string(64)	S	Currently unused
client_persistent_caching	Boolean	S	Enables or disables persistent client caching. The default is T (TRUE), meaning caching is enabled for the session.
connect_callback_enabled	Boolean	S	T or F, indicating whether the server calls the content callback functions. The default is T.
connect_failure_callback	integer	R	Contains the memory addresses of user-defined applications to capture and handle login failures.
connect_failure_data	integer	R	Contains the memory addresses of data to be passed to user-defined applications that handle login failures.
connect_pooling_enabled	Boolean	S	TRUE indicates that connection pooling (of repository sessions) is enabled for the API session. The default is FALSE.

Attribute	Datatype	Single/ repeating	Description
connect_recycle _interval	integer	S	Defines how long a single repository connection may reside in the connection pool before being recycled. The default is 300 seconds.
connect_retry _interval	integer	S	The time, in seconds, that a client waits before retrying a failed connection. The attempted connection may be either with a Connect method call or a server-initiated connection. Useful when many connections are being attempted in a short period of time.
connect_retry _limit	Integer	S	The total number of times a client tries to connect through either a Connect method call or a server-initiated connection, minus one.
connect_success _callback	integer	R	Contains the memory addresses of user-defined applications that clear status messages resulting from execution of applications identified in new_connection_callback.
connect_success _data	integer	R	Contains the memory addresses of data to be passed to applications identified in connect_success_callback.
connect_timeout	integer	S	The time, in seconds, that a client waits for a network request to complete. Used for 16-bit Windows only.

Attribute	Datatype	Single/ repeating	Description
content_callback_data	integer	R	Contains the memory address of the argument values for the content callback function.
content_callback_function	integer	R	Contains the memory address of a content callback function.
debug_dbid	ID	S	Contains the repository's repository ID. This only has a value if the dmcl.ini defines a DOCBROKER_DEBUG_BYPASS section.
debug_host	string(32)	S	Identifies the host machine on which the repository resides. This only has a value if the dmcl.ini defines a DOCBROKER_DEBUG_BYPASS section.
debug_port	integer	S	Defines the port number which repository's server is using. This only has a value if the dmcl.ini defines a DOCBROKER_DEBUG_BYPASS section.
debug_service	string(32)	S	Contains the service name for the server. This only has a value if the dmcl.ini defines a DOCBROKER_DEBUG_BYPASS section.
docbroker_search_order	string	S	Determines whether connection requests are sent to connection brokers in the order in which they are listed in the dmcl.ini or randomly. Valid values are: <ul style="list-style-type: none"> • sequential • random <p>The default is sequential.</p>

Attribute	Datatype	Single/ repeating	Description
exception_count	integer	S	<p>This attribute is supported on Windows platforms only.</p> <p>Defines how many exceptions the DMCL keeps a record of during a session. The default is 0.</p>
exception_count_interval	integer	S	<p>This attribute is supported on Windows platforms only.</p> <p>Defines a time interval in which the number of recorded exceptions may not exceed the value in exception_count plus one. If the number of recorded exceptions exceed the value of exception_count plus one with the given interval, the DMCL shuts down.</p> <p>The value is interpreted in minutes.</p> <p>The default is 0.</p>
force_coherency_checks	Boolean	S	<p>T disables the use of consistency check rules (for persistent client caches) defined in queries or a cache config object. The default is F (FALSE).</p>
ini_file_path	string(255)	S	<p>Specifies the location of the .ini file.</p>
i_override_list	Boolean	S	<p>Used internally</p>

Attribute	Datatype	Single/ repeating	Description
local_clean_on_init	Boolean	S	Indicates whether the server should automatically purge the client local area whenever the client is started. The default is T.
local_diskfull_limit	integer	S	Specifies how full the disk that contains the local area can become. Expressed as a percentage between 1 and 100. When the disk reaches the defined limit, getfiles to the local area fail with an error.
local_diskfull_warn	integer	S	Specifies the threshold at which a getfile to the client local area generates a diskfull warning. Expressed as a percentage between 1 and 100. The percentage is a percentage of the total possible space.
local_path	string(255)	S	Specifies the location of the local common area.
local_purge_on_diskfull	Boolean	S	Indicates if you want the system to automatically execute a Purgelocal method when the diskfull limit is reached. The default is true—purging will occur automatically when the diskfull limit is reached.
max_collection_count	integer	S	Determines how many collections are allowed per repository session.
max_connection_per_session	integer	S	Maximum number of subconnections allowed in each primary repository session.

Attribute	Datatype	Single/ repeating	Description
max_session _count	integer	S	Determines how many repository sessions are allowed in an API session.
network_callback_ data	integer	R	Contains the memory address of the argument values for the network callback function.
network_callback _function	integer	R	Contains the memory address of the network callback function.
network__enabled	Boolean	S	T or F, indicating whether the server calls the network callback functions. The default is T.
network_requests	integer	S	Number of RPC calls for all sessions in current API session. This is a cumulative total, incrementing by one each time an RPC call is made during a session.
new_connection _callback	integer	R	Contains the memory address of user-defined applications that display status messages for new repository connections prior to establishment of the connection.
new_connection _data	integer	R	Contains the memory address of data used by applications identified in new_connection_callback.
nfs_enabled	Boolean	S	Indicates whether the client is using NFS as its file-sharing protocol.
primary_host	string(128)	S	Identifies the host machine on which the primary connection broker resides.

Attribute	Datatype	Single/ repeating	Description
primary_port	integer	S	Defines the port number which the primary connection broker is using.
primary_protocol	string(32)	S	Names the protocol in use by the client's primary connection broker.
primary_service	string(32)	S	Contains the service name of the primary connection broker.
primary_timeout	integer	S	Defines how long a client waits for a response from the primary connection broker before forwarding the request to the backup connection broker.
r_dmcl_version	string(64)	S	Contains the version number of the dmcl in use.
r_process_domain_name	string(64)	S	Identifies the domain of the user who is executing the current process. This attribute supports servers on Windows machines only. It is blank if the server is running on a UNIX machine.
r_process_user_name	string(64)	S	The operating system login name of the user executing the current process.
r_trace_file	string	S	Specifies a file or directory used to log API tracing. If a directory is specified, then the DMCL creates a file in the directory with a unique name. The file name begins with dmcl_trace_. (For UNIX or Windows only).

Attribute	Datatype	Single/ repeating	Description
r_trace_level	integer	S	The tracing level. 10 = api tracing 15 = lock tracing (for WIN32 only)
ref_binding_label	string(32)	S	Defines which document version to fetch when accessing a remote document. If this attribute is defined, its value overrides the binding specified in the dm_reference object that points to the remote object.
terminate_on_exception	Boolean	S	(Windows platforms only) Controls whether the DMCL shuts down or continues when an exception occurs or the DMCL cannot continue on exception.
trace_callback	integer	S	Used internally to integrate DMCL trace messages with DFC trace log files.
umask	string(4)	S	Modifies the default operating system permissions assigned to directories and files created by the client DMCL. This attribute, if set, affects clients on UNIX platforms only.
use_compression	Boolean	S	Indicates whether the server will use data compression when transferring files from storage to client local areas.

Attribute	Datatype	Single/ repeating	Description
use_content_server	Boolean	S	<p>Determines whether the client uses separate data and content servers or only the data server for all repository requests.</p> <p>TRUE directs the client to use data and content servers. FALSE directs the client to send requests only to the data server.</p> <p>The default is TRUE.</p>
use_local_always	Boolean	S	<p>Indicates whether the client is using a local area for files.</p>
use_local_on_copy	Boolean	S	<p>Used in conjunction with the use_local_always and nfs_enabled (api config) attributes to determine whether the server copies requested files to the client local area. The default is F.</p>

App Ref

Purpose An app ref object is used internally to reference an object used in an application.

Implementation

Supertype: SysObject
Subtypes: None
Internal name: dm_app_ref
Object type tag: 07

Objects of this type reference objects used in an DocApp. Users cannot create app ref objects directly. They are created and managed by the server, Documentum Application Builder, or the Documentum Application Installer. .

[Table 2-7, page 80](#), lists the only attribute defined for the type

Table 2-7. Attribute defined for the app ref type

Attribute	Datatype	Single/ repeating	Description
application_obj_id	ID	S	Object ID of the object referenced by the application.

Application

Purpose An application object represents a packaged application that can be installed by the DocApp Installer.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_application
 Object type tag: 08

Application objects represent packaged DocApps. Users cannot create application objects directly. They are created through the Documentum Application Builder or Documentum Application Installer when a user creates a DocApp.

Table 2–8, page 81, lists the attribute defined for the type.

Table 2-8. Attribute defined for the application type

Attribute	Datatype	Single/ repeating	Description
app_version	string(16)	S	Application version label.
application_object_id	ID	R	Object IDs of the objects in the application.
content_transfer_option	integer	R	Defines how to handle cabinets or folders when copying them to a target repository. Valid values are: 0, Copy all contained objects 1, Copy only the hierarchy (the directly or indirectly contained folders, but not the documents in the folders) 2, Copy just the cabinet or folder itself, but none of its contained objects 3, Copy only the cabinet

Attribute	Datatype	Single/ repeating	Description
			or folder and its directly contained documents.
			The value at an index position applies to the object at the corresponding index position in application_object_id
copyright_string	string(255)	S	User-defined.
def_alias_set_id	ID	S	The object ID of the default alias set for the application. This is set through Developer Studio
post_install_proc_id	ID	S	Chronicle ID of the post-installation procedure
post_install_proc_label	string(32)	S	Version label of the post-installation procedure
pre_install_proc_id	ID	S	Chronicle ID of the pre-installation procedure
pre-install_proc_label	string(32)	S	Version label of the pre-installation procedure
target_loc_alias	string(64)	R	An alias that resolves to the location to which to copy the object in the application. The value at an index position applies to the object at the corresponding index position in application_object_id.
target_perm_alias	string(64)	R	An alias that resolves to the name of a template ACL. The value at an index position applies to the object at the corresponding index position in application_object_id.

Attribute	Datatype	Single/ repeating	Description
target_owner_alias	string(64)	R	An alias that resolves to the name of the owner of the object. The value at an index position applies to the object at the corresponding index position in application_object_id.
upgrade_option	integer	S	<p>Defines how to handle the object when the application is upgraded. The value at an index position applies to the object at the corresponding index position in application_object_id.</p> <p>Valid values are:</p> <p>0, Overwrite the object 1, Ignore the object 2, Version the object</p>
user_runnable	Boolean	S	Reserved for future use.

Aspect Relation

Purpose Describes the compatibility of two aspects.

Implementation

Supertype: Relation
Subtypes: None
Internal name: dmc_aspect_relation
Object type tag: 37



Caution: This object type is for internal use only. It is not currently supported for external use.

An aspect relation object type describes the compatibility of two aspects.

[Table 2-9, page 84](#), lists the attributes defined for the type.

Table 2-9. Attributes defined for the aspect relation type

Attribute	Datatype	Single/ repeating	Description
child_aspect_name	string(64)	S	Reserved for internal use
parent_aspect_name	string(64)	S	Reserved for internal use
relation	integer	S	Reserved for internal use

Aspect Type

Purpose Stores the files that implement an aspect.

Implementation

Supertype: Module
 Subtypes: None
 Internal name: dmc_aspect_type
 Object type tag: 0b



Caution: This object type is for internal use only. It is not supported for external use.

An aspect type object is a folder that contains the implementation classes and other information about one aspect. (An aspect is a custom behavior implemented for a particular method or object type class.) Aspect type folders are stored in the repository under System/Business Objects/Aspects.

[Table 2-10, page 85](#), lists the attributes defined for the type.

Table 2-10. Attributes defined for the aspect type object type

Attribute	Datatype	Single/ repeating	Description
aspect_category	string(32)	R	Used internally
attaching_aspect_ policy	string(128)	R	Used internally
copy_policy	integer	S	Used internally
detaching_aspect_ policy	string(128)	R	Used internally
target_object_type	string(27)	R	Used internally
version_policy	integer	S	Used internally

Assembly

Purpose An assembly object describes a component of a virtual document.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dm_assembly
 Object type tag: 0d

Assembly objects are created when a user creates an assembly, a snapshot of the virtual document at particular point in time. Users must have at least Version permission for the object identified in the book_id attribute to modify an assembly object.

Table 2–11, page 86, lists the attributes defined for the type.

Table 2-11. Attributes defined for the assembly type

Attribute	Datatype	Single/ repeating	Description
a_contain_desc	string(255)	S	Used by Documentum clients to manage XML documents in assemblies. For more information, refer to XML support, page 165 in <i>Content Server Fundamentals</i> .
a_contain_type	string(32)	S	Used by Documentum clients to manage XML documents in assemblies. For more information, refer to XML support, page 165 in <i>Content Server Fundamentals</i> .

Attribute	Datatype	Single/ repeating	Description
book_id	ID	S	Object ID of the topmost containing document in the virtual document structure that contains the component described by this assembly object.
component _chron_id	ID	S	Chronicle ID of the component represented by the assembly object.
component_id	ID	S	Object ID of the component of the virtual document
depth_no	integer	S	Depth of the component within the levels of the virtual document identified by book_id.
order_no	integer	S	The ordering of the component within the virtual document identified by the book_id.
parent_id	ID	S	Object ID of the document that directly contains the virtual document described by this assembly object.
path_name	Oracle: string(740) SQLServer: string(765) Sybase string(600) DB2: string(740)	S	The path from the top-most containing document (identified in the book_id attribute) to the component. Path names are comprised of the object names of the nodes delimited by forward slashes. If the path is longer than the attribute, the path is truncated from the end of the path.

Audit Trail

Purpose An audit trail object stores information about one audited event.

Implementation

Supertype: Persistent Object
Subtypes: Audittrail Acl, Audittrail Group
Internal Name: dm_audittrail
Object type tag: 5f

Audit trail objects are created automatically by the server after a user initiates auditing for a particular event. Automatic auditing is provided for system events, such as a checkin or checkout, workflow events, and lifecycle events.

Users and applications can also create audit trail objects to record the occurrence of user-defined events.

Audit trail objects have ten attributes whose use is dependent on the event. These attributes are the `id_n` and `string_n` attributes. [Auditing, page 406](#) in the *Content Server Administrator's Guide* provides a complete list of auditable events and describes the use of the `id_n` and `string_n` attributes for server-generated audit trail objects.

[Table 2-12, page 88](#), lists the attributes defined for the type.

Table 2-12. Attributes defined for the audit trail type

Attribute	Datatype	Single/ repeating	Description
acl_domain	string(32)	S	Owner of the ACL associated with the object being audited. This is null for objects that are not SysObjects or SysObject subtypes.
acl_name	string(32)	S	Name of the ACL associated with the object being audited. This is null for objects that are not SysObjects or SysObject subtypes.

Attribute	Datatype	Single/ repeating	Description
application_code	string(64)	S	The application code or codes defined for the session in which the event occurred. Multiple codes are separated by commas. Note: The list of codes is truncated if necessary.
attribute_list	For all except DB2: string(2000) For DB2: string(1300)	S	Comma-separated list of audited attributes and their values. If the list is longer than the length of the attribute, the overflow is stored in the dmi_audittrail_attrs object identified in the attribute_list_id attribute.
attribute_list_id	ID	S	Object ID of the dmi_audittrail_attrs object that contains the overflow values from the attribute_list attribute.
audit_signature	string(255)	S	Signed hash of the audit trail entry.
audit_version	integer	S	Indicates how the signature was created if the entry is signed. The only valid value is 1, to indicate that 5.2 Content Server created the value in audit_signature. Otherwise, this is not set.

Attribute	Datatype	Single/ repeating	Description
audited_obj_id	ID	S	<p>Object ID of the object being audited</p> <p>For a checkin event, this is the object ID of the new version of the object.</p> <p>For workflow events, this is always the object ID of the process object.</p> <p>For dm_move_content events, this is always the object ID of the content object.</p>
audited_obj_vstamp	integer	S	The i_vstamp value of the object identified in the audited_obj_id attribute.
chronicle_id	ID	S	Chronicle ID of the object being audited if the object is a SysObject or SysObject subtype. Otherwise, this is the object ID of the object being audited.
controlling_app	string(32)	S	The object's controlling application. This is the value in the a_controlling_app attribute of the object being audited.
current_state	string(32)	S	Name of the current lifecycle state of the object being audited.
event_description	string(64)	S	Name of the event being audited in a user-friendly form.
event_name	string(64)	S	Actual name of the event being audited.

Attribute	Datatype	Single/ repeating	Description
event_source	string(64)	S	Source of the event. Some possible values are: lifecycle electronic signature system unspecific workflow
host_name	string(128)	S	Name of the host on which the Content Server that generated the audit trail entry resides
i_audited_obj_class	integer	S	Identifies what type of object is identified in audited_obj_id. Values are: -1, meaning the event has no audited object (for example, dm_audit or dm_unaudit events) 0, dm_sysobject and subtypes 1, for dm_acl 2, for dm_user and subtypes 3, for dm_group 4, for dmr_content

Attribute	Datatype	Single/ repeating	Description
i_is_archived	Boolean	S	T means that the audit trail entry has been archived. This is set to F, meaning that the entry has not been archived, when the audit trail entry is created and reset to T when the MARK_AS_ARCHIVED administration method is executed against the entry.
id_1	ID	S	event-dependent object ID
id_2	ID	S	event-dependent object ID
id_3	ID	S	event-dependent object ID
id_4	ID	S	event-dependent object ID
id_5	ID	S	event-dependent object ID
object_name	string(255)	S	Name of the object being audited. For SysObjects and ACLs, this is the object_name attribute value. For groups, this is the group_name value.
object_type	string(32)	S	Object type of the object being audited.
owner_name	string(32)	S	Name of the owner of the object being audited. The value may be an individual user or a group name.
policy_id	ID	S	Object ID of the lifecycle associated with the object being audited.
r_gen_source	integer	S	The source of the audit trail. Values are: 1, for a system-generated audit trail 0, for a user-generated audit trail

Attribute	Datatype	Single/ repeating	Description
registry_id	ID	S	Object ID of the registry object that stores the registration for the event that generated this audit trail entry.
session_id	ID	S	ID of the session in which the event occurred.
string_1	string(200)	S	event-dependent string
string_2	string(200)	S	event-dependent string
string_3	string(200)	S	event-dependent string
string_4	string(200)	S	event-dependent string
string_5	string(200)	S	event-dependent string
time_stamp	date	S	The local time at which the audit trail was generated
time_stamp_utc	date	S	The UTC (or GMT) time at which the event occurred
user_id	ID	S	Object ID of the dm_user object representing the user whose task caused the event to occur.
user_name	string(32)	S	Name of the user whose task caused the event to occur.
version_label	string(16)	S	The implicit (numeric) version label of the object being audited. This is null for objects that are not SysObjects or SysObject subtypes.
workflow_id	ID	S	Object ID of the dm_workflow for workflow events.

Audit Trail ACL

Purpose An audit trail acl object records information about one audited event for an ACL.

Implementation

Supertype: Audit Trail
Subtypes: None
Internal Name: dm_audittrail_acl
Object type tag: 5f

Audit trail acl objects are created when a dm_save, dm_saveasnew, or dm_destroy registered event occurs on an ACL. (The audited ACL is identified by the inherited audited_obj_id attribute.)

Table 2–13, page 94, lists the attributes defined for the type.

Table 2-13. Attributes defined for the audit trail acl type

Attribute	Datatype	Single/ repeating	Description
accessor_name	string(32)	R	The ACL user or group entry on which the operation at the corresponding index position in accessor_operation was performed.
accessor_operation	string(1)	R	The operation performed on the user or group entry identified in the corresponding position in accessor_name. Valid values are: U, for update I, for insert D, for delete

Attribute	Datatype	Single/ repeating	Description
accessor_permit	integer	R	Value in the ACL's r_accessor_permit attribute for the ACL entry identified in the dm_audittrail_acl.accessor_name attribute at the corresponding index position.
accessor_xpermit	integer	R	Value in the ACL's r_accessor_xpermit attribute for the ACL entry identified in the dm_audittrail_acl.accessor_name attribute at the corresponding index position.
acl_class	integer	S	Value in the audited ACL's acl_class attribute.
application_permit	string(128)	R	Value in the ACL's application_permit attribute for the ACL entry identified in the dm_audittrail_acl.accessor_name attribute at the corresponding index position.
description	string(128)	S	Value in the audited ACL's description attribute.
globally_managed	Boolean	S	Value in the audited ACL's globally_managed attribute.
is_group	Boolean	R	Value in the ACL's r_is_group attribute for the ACL entry identified in the dm_audittrail_acl.accessor_name attribute at the corresponding index position.

Attribute	Datatype	Single/ repeating	Description
is_internal	Boolean	S	Value in the ACL's r_is_internal attribute.
permit_type	integer	R	Value in the ACL's r_permit_type attribute for the ACL entry identified in the dm_audittrail_acl.accessor_name attribute at the corresponding index position.

Audit Trail Attrs

Purpose An audit trail attrs object stores the overflow attribute name and value pairs from the `attribute_list` attribute of an audit trail object.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal Name: `dmi_audittrail_attrs`
 Object type tag: 00

Audit trail attr objects are created automatically, when needed, by Content Server. They record overflow attribute names and their values if the list of attributes to be recorded is too long to fit into the `attribute_list` attribute of the associated audit trail object.

[Table 2-14, page 97](#), lists the attributes defined for the type.

Table 2-14. Attributes defined for the audit trail attrs type

Attribute	Datatype	Single/ repeating	Description
<code>attribute_list</code>	<code>string(2000)</code>	R	Comma-separated list of attribute name and value pairs to be audited.
<code>audit_obj_id</code>	ID	S	Object ID of the associated audit trail object.

Audit Trail Group

Purpose An audit trail group object records information about an audited event for a group.

Implementation

Supertype: Audit Trail
 Subtypes: None
 Internal Name: dm_audittrail_group
 Object type tag: 5f

Audit trail group objects are created when a dm_save, saveasnew, or dm_destroy registered event occurs on a group. (The audited group is identified in the inherited audited_obj_id attribute.)

Table 2-15, page 98 lists the attributes defined for the object type.

Table 2-15. Attributes defined for the audit trail group type

Attribute	Datatype	Single/ repeating	Description
alias_set_id	ID	S	Value in the audited group's alias_set_id attribute.
description	string(128)	S	Value in the audited group's description attribute.
globally_managed	Boolean	S	Value in the audited group's globally_managed attribute.
group_admin	string(32)	S	User or group allowed to modify the audited group.
group_address	string(80)	S	Value in the audited group's group_address attribute.
group_class	string(32)	S	Value in the audited group's group_class attribute.

Attribute	Datatype	Single/ repeating	Description
group_source	string(16)	S	Identifies the source of the audited group. The only valid value is LDAP, meaning the group was created by importing an LDAP group.
groups_names	string(32)	R	Names of groups that are users in the audited group on which an operation occurred. The operation is recorded in the groups_names_operation attribute in the corresponding index position.
groups_names_operation	string(1)	R	The operation that occurred on the group identified in the groups_name attribute at the corresponding index position. Valid values are: U, for update I, for insert D, for delete
is_dynamic	Boolean	S	T (TRUE) means the audited group is a dynamic group. F (FALSE) means the group is not a dynamic group.
is_dynamic_default	Boolean	S	Controls whether users in the audited group's list of potential users are considered members of the group by default when they connect to the repository. T (TRUE) means that users are treated as members of the group. F (FALSE) means that users are not treated as group members when they connect.

Attribute	Datatype	Single/ repeating	Description
is_private	Boolean	S	Value in the group's is_private attribute .
users_names	string(32)	R	Names of individual users in the audited group on which an operation occurred. The operation is recorded in the groups_names_operation attribute in the corresponding index position.
users_names_operation	string(1)	R	The operation that occurred on the user identified in the users_name attribute at the corresponding index position. Valid values are: U, for update I, for insert D, for delete

Auth Config

Purpose An auth config object contains the names of the primary and backup domain controllers for a particular domain.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal Name: dm_auth_config
 Object type tag: 00

The values in an auth config object are used when you authenticate users in repository on a UNIX platform against a Windows domain. A repository can have only one auth config object. It is created and its values set through Documentum Administrator, when `unix_domain_used` is defined in the `auth_protocol` attribute of the `docbase config` object.

[Table 2-16, page 101](#), lists the attributes defined for the type.

Table 2-16. Attributes defined for the auth config type

Attribute	Datatype	Single/ repeating	Description
domain_name	string (32)	R	The name of a Windows domain
primary_controller	string(32)	R	The name of the machine that is the primary domain controller for the domain named at the corresponding index position in domain_name.
backup_controller	string(32)	R	The name of the machine that is the backup domain controller for the domain named at the corresponding index position in domain_name.

Blob Store

Purpose A blob store object represents a blob storage area.

Implementation

Supertype: Store
Subtypes: None
Internal name: dm_blobstore
Object type tag: 40

Blob store objects represent blob store storage areas. Content stored in blob stores is stored in tables in the repository. You can store a maximum of 64 Kbytes of data in a blob store storage area.

[Table 2-17, page 102](#), lists the attributes defined for the type.

Table 2-17. Attributes defined for the blob store type

Attribute	Datatype	Single/ repeating	Description
ascii	Boolean	S	Indicates whether the contents are ASCII strings or arbitrary sequences of 8-bit characters.
selectable	Boolean	S	Not currently used.

Builtin Expr

Purpose A builtin expr object stores information needed to execute built-in server functions.

Implementation

Supertype: Expression
 Subtypes: None
 Internal name: dm_builtin_expr
 Object type tag: 54

Builtin expr objects store information needed to execute built-in server functions. Content Server creates and manages builtin expr objects. Users cannot create them. [Table 2-18, page 103](#) lists the attribute defined for the type.

[Table 2-18, page 103](#), lists the attribute defined for the type.

Table 2-18. Attribute defined for the builtin expr type

Attribute	Datatype	Single/ repeating	Description
builtin_tag	integer	S	Specifies which server function to invoke. Valid values are: 1, for NULL 2, for NOW 3, for TODAY 4, for TOMORROW 5, for YESTERDAY 6, for USER None of the functions require passed arguments.

CA Store

Purpose A CA store object represents a content addressable storage system known to Content Server.

Implementation

Supertype: Store
Subtypes: None
Internal name: dm_ca_store
Object type tag: 6d

A CA store object represents a content-addressable storage system. A content addressable storage system uses a content address instead of a directory path to locate content stored in the system.

[Table 2-19, page 104](#), lists the attributes defined for the type.

Table 2-19. Attributes defined for the CA store type

Attribute	Datatype	Single/ repeating	Description
a_content_attr_ description	string(128)	R	User-defined description of the attribute named at the corresponding position in a_content_attr_name
a_content_attr_ name	string(64)	R	Names of the content metadata fields whose values will be set using a Setcontentattrs method. The names may not contain spaces. This attribute can have a maximum of 62 values. Do not include the metadata field identified in a_retention_attr_name, if any.

Attribute	Datatype	Single/ repeating	Description
a_default_retention_date	Date	S	<p>The default retention value for content stored in this storage system.</p> <p>The value is ignored if a_retention_attr_required is set to T (TRUE) or a_retention_attr_name is set.</p>
a_plugin_id	ID	S	Object ID of the plugin object whose content is the plugin library that implements this ca storage system.
a_retention_attr_name	string(64)	S	<p>Identifies the content metadata field that stores the retention value for the content.</p> <p>The metadata field name may not contain spaces. Do not use a metadata field defined in a_content_attr_name.</p> <p>The value of this attribute may not be changed after the storage area is created.</p>
a_retention_attr_required	Boolean	S	<p>T (TRUE) means that the content stored in this system must have a retention period. If set to T, you must set a_retention_attr_name also.</p> <p>The default is F (FALSE).</p>
a_storage_params	string(1024)	R	<p>List of parameters specific to the content addressable storage system.</p> <p>The first index position ([0]) is reserved for the IP address or addresses for the Centera host machine or machines. The value may include the path to a Centera profile.</p> <p>The second index position ([1]) is reserved for embedded blob configuration. If set, the value is in the format:</p> <pre>pool_option:embedded_ blob:size_in_KB</pre> <p>This instructs the Centera system to store all content files less than or equal to the specified size as embedded blobs. (Refer to the</p>

Attribute	Datatype	Single/ repeating	Description
			<i>Content Server Administrator's Guide</i> for a complete description of this feature.)

Cabinet

Purpose The cabinet object represents the highest level of organization visible to end users in a repository.

Implementation

Supertype: Folder
 Subtypes: None
 Internal name: dm_cabinet
 Object type tag: 0c

The cabinet object represents the highest level of organization visible to end users in a repository. All folders, documents, and other objects (except cabinets) that are subtypes of the SysObject type are stored in cabinets. A cabinet is basically a folder that cannot be placed inside another folder or a cabinet. Users must have Superuser, Sysadmin, or Create Cabinet user privilege to create or destroy a cabinet. However, users can change a cabinet's attributes if they have Write permission for the cabinet.

[Table 2-20, page 107](#), lists the only attribute defined for the type.

Table 2-20. Attributes defined for the cabinet type

Attribute	Datatype	Single/ repeating	Description
is_private	Boolean	S	Indicates whether the cabinet is private or public. If set to TRUE, the cabinet is private. If set to FALSE, the cabinet is public. The default is FALSE. This attribute is not used by Content Server for security or any other use. It is intended for use by client applications.

Cache Config

Purpose A cache config object identifies a group of queries, objects, or both to be cached on the client and refreshed at defined intervals.

Implementation

Supertype: SysObject
Subtypes: None
Internal Name: dm_cache_config
Object type tag: 08

A cache config object identifies a group of queries, objects, or both to be cached on the client and refreshed at defined intervals.

[Table 2-21, page 108](#), lists the attributes defined for the type.

Table 2-21. Attributes defined for the cache config type

Attribute	Datatype	Single/ repeating	Description
cache_element_query	string(1500)	R	The queries that return the query results, objects or both to be cached. The value can be any valid query. Each index position can contain one query.
cache_element_type	string(32)	R	Indicates whether the value at the corresponding position in cache_element_query identifies a query or object to be cached. Valid values are: query object

Attribute	Datatype	Single/ repeating	Description
client_check _interval	integer	S	<p>Defines how often the client calls the server to obtain updated information about the cache config object.</p> <p>The default value is 0, meaning that the client requests updated information every time an API call specifies the cache config object.</p> <p>The value is interpreted in seconds.</p>
i_query_result _hash	string(100)	S	<p>Last computed hash of the query results. This value is used to compare executions of the cached queries.</p>
r_last_changed _date	Date	S	<p>Time at which the server last validated the queries and found a change.</p>
r_last_checked _date	Date	S	<p>Time at which the server last validated the queries.</p>
server_check _interval	integer	S	<p>Interval, in seconds, at which the server revalidates the queries in the cache configuration.</p> <p>The default value is 0, meaning that the server reruns all the queries every time an API call specifies the cache config object and the client_check_interval has expired.</p>

Category

Purpose Represents a taxonomy in a repository.

Implementation

Supertype: Folder
 Subtypes: Taxonomy
 Internal Name: dm_category
 Object type tag: 0b

A category object is a folder in the folder structure that represents a taxonomy in a repository. Every category object must be linked to at least one taxonomy object or other category object. (For more information about categories and taxonomies, refer the Content Intelligence Services documentation.)

[Table 2–22, page 110](#), lists the attributes defined for the category type and the inherited attributes whose use is specific to the category object type.

Table 2-22. Attributes defined for the category type

Attribute	Datatype	Single/ repeating	Description
a_status	string(16)	S	Describes the development and deployment status of the category. Valid values are: offline online
allowed_operations	string(32)	R	List of allowed operations for the category. Valid values are: user_browse user_subscribe user_subscribe_mandatory
candidate_threshold	integer	S	Relevance level at which a document is automatically routed for approval to the category owner. Values are interpreted as a percentage, from 1 to 100.

Attribute	Datatype	Single/ repeating	Description
category_evidence	string(255)	R	Identifies other categories whose evidence will be considered as evidence for this category. Values for this attribute are set using the Content Intelligence Client.
category_owner	string(32)	R	The owner of the category. Note: Only one owner is currently supported.
child_count	integer	S	Number of categories linked directly to this category.
class_id	ID	S	Object ID of the category class to which this category belongs. Defaults to the value of the class_id attribute in the taxonomy.
description	string(255)	S	User-defined description of the category.
definition_type	string(32)	S	Identifies the type of the definition. The definition is comprised of the values in the category_evidence, keyword_evidence, candidate_threshold, and on_target_threshold attributes. These values determine which documents can be automatically assigned to the category. Valid values are: none , which indicates that the category has no definition or the definition is to be ignored, meaning documents cannot be automatically assigned to the category simple , which indicates that the category has a simple definition
keyword_evidence	string(255)	R	List of keywords in document text considered to be evidence for the category. The values in this attribute are set using Content Intelligence Client.

Attribute	Datatype	Single/ repeating	Description
object_name	string(255)	S	The name of the category. The name must be unique among all categories within the category class.
on_target_threshold	integer	S	The relevance level at which a document is automatically assigned to the category. The value is interpreted as a percentage, from 1 to 100.
qualifiers	string(255)	R	List of initial conditions that a document must meet to qualify for testing against the evidence for the category. The values in this attribute are set using Content Intelligence Client.
supported_language	string(32)	R	List of language codes for all supported languages. The entry at each position must have a category name in the specified language at the matching index position in translated_name.
translated_name	string(255)	R	List of the category name in all supported languages. The name at each index position must be in the language specified by the language code in the matching index position in supported_language.

Category Assign

Purpose Records the assignment of an object to a particular category.

Implementation

Supertype: Relation
 Subtypes: None
 Internal Name: dm_category_assign
 Object type tag: 37

A category assign object records the assignment of an object to a particular category. Category assign objects are created automatically when an object is assigned or proposed for assignment to a category folder.

Note: The data dictionary label for this type is Category Assignment.

[Table 2–23, page 113](#), lists the attributes defined for the type and any inherited attributes that have a use specific to the object type.

Table 2-23. Attributes defined for the category assign type

Attribute	Datatype	Single/ repeating	Description
active_assignment	Boolean	S	Whether the object represented by this category assign object is visible to users. T (TRUE) means the object is available to end users. The value is T when assign_type value is active and the assign_status value begins with “assigned”. F (FALSE) means the object is not available.

Attribute	Datatype	Single/ repeating	Description
assign_status	string(32)	S	Describes the assignment operation. Valid values are: assigned_auto assigned_manual assigned_approved assigned_pending_remove assigned_final pending_assign removed_auto removed_manual removed_approved removed_final
assign_type	string(32)	S	Identifies whether the assignment took place during a test or production run. Valid values are: text active
child_id	ID	S	The i_chronicle_id value of the assigned object.
document_id	ID	S	Object ID of the assigned object.
modifier	string(32)	S	Name of the user or process that last modified the assignment. For all automatic assignment operations, this value is CIS.
parent_id	ID	S	Object ID of the category to which the object is assigned.
permanent_link	Boolean	S	Whether to maintain the assignment across versions of the assigned object. The default is F (FALSE).

Attribute	Datatype	Single/ repeating	Description
prev_assign_status	string(32)	S	<p>Describes the assigned object's previous assignment, if any. Valid values are:</p> <p>assigned_auto assigned_manual assigned_approved assigned_pending_remove assigned_final pending_assign removed_auto removed_manual removed_approved removed_final</p>
prev_modifier	string(32)	S	<p>Name of the user or process that performed the object's previous assignment. For all automatic assignments, this value is CIS.</p>
relation_name	string(32)	S	<p>Name of the relationship between the assigned object and the category. The only legal value is dm_category_assign.</p>
relevance	integer	S	<p>The relevance number of the assigned object. Values range from 1 to 100, with 100 meaning fully relevant.</p>
run_id	ID	S	<p>The object ID of the dm_docset_run object that identifies the run that classified and assigned the object.</p> <p>This attribute can be a NULL value.</p>

Category Class

Purpose Stores the definition of a class of categories and the default attribute values and behaviors for the included categories.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal Name: dm_category_class
 Object type tag: 08

A category class object stores the definition of a class of categories and the default attribute values and behaviors for the included categories. Category class objects are created using Content Intelligence Client.

[Table 2-24, page 116](#), lists the attributes defined for the type and those inherited attributes that have a use specific to the object type.

Table 2-24. Attributes defined for the category class type

Attribute	Datatype	Single/ repeating	Description
cat_evidence_conf	string(32)	S	Default confidence level for all explicit (not propagated) category evidence for categories in this class. Valid values are: certain high medium low auxilliary negative off <i>numeric</i> <i>numeric</i> is any value from 0 to 100.
description	string(255)	S	User-defined description of the category class.

Attribute	Datatype	Single/ repeating	Description
evidence_prop_conf	string(32)	S	<p>Default confidence level for all propagated category evidence for categories in this class. Valid values are:</p> <p>certain high medium low auxilliary negative <i>numeric</i></p> <p><i>numeric</i> is any value from 0 to 100.</p>
evidence_prop_type	string(32)	S	<p>Default value for all automatic (propagated) category evidence generation for categories in the class. Valid values are:</p> <p>off, which turns off evidence propagation @child, which propagates evidence from any child to the parent category @parent, which propagates any evidence for the parent category to immediate children</p>
implied_keyword_conf	string(32)	S	<p>Default keyword confidence level for all categories in this class. Valid values are:</p> <p>certain high medium low auxilliary negative off <i>numeric</i></p> <p><i>numeric</i> is any number from 0 to 100.</p>

Attribute	Datatype	Single/ repeating	Description
implied_keyword_style	string(64)	S	<p>Default implied keyword style for categories in this class. Valid values are combinations of the following:</p> <ul style="list-style-type: none"> active stem phrase_order_exact
keyword_conf	string(32)	S	<p>Default keyword confidence level for all categories in the class. Valid values are:</p> <ul style="list-style-type: none"> certain high medium low auxiliary negative off <i>numeric</i> <p><i>numeric</i> is an integer from 0 to 100.</p>
keyword_style	string(64)	S	<p>Default keyword style for categories in this class. Valid values are combinations of:</p> <ul style="list-style-type: none"> active stem phrase_order_exact
object_name	string(255)	S	<p>Category class name. The name must be unique among the categories that belong to the class.</p>

Attribute	Datatype	Single/ repeating	Description
source	string(32)	S	User-defined text identifying the source of the taxonomy and category objects in this class. For example: DCTM or Semio.
target_attribute	string(32)	S	Name of the attribute in a repository object type to be used for storing all document category assignments. The attribute must be a repeating attribute.

Change Record

Purpose Stores information used internally to ensure consistency of cached information.

Implementation

Supertype: Persistent Object

Subtypes: None

Internal Name: dmi_change_record

Object type tag: 33

A change record object stores information used internally to ensure consistency of cached information. There is one change record object for each repository.

[Table 2-25, page 120](#), lists the attributes defined for the type.

Table 2-25. Attributes defined for the change record type

Attribute	Datatype	Single/ repeating	Description
cache_change _count	integer	S	Number of changes affecting the server's global cache.
dd_change_count	integer	S	Number of times the data dictionary has been changed.
group_change_ count	integer	S	Number of times any group object has been changed.
reinit_change _count	integer	S	Number of times the main server thread (parent server) has been reinitialized.
sys_change_count	integer	S	Number of changes made to object types and format objects.
storage_change _count	integer	S	Number of times that a storage area has changed state.

Attribute	Datatype	Single/ repeating	Description
type_change_count	integer	S	Number of times that a type in the global cache has been changed.
user_change_count	integer	S	Number of times any user object has been changed.

CI Config

Purpose Stores Content Intelligence Services configuration options.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_ci_config
 Object type tag: 08

A CI config object stores Content Intelligence Services configuration options. CI config objects are created when a repository is enabled for Content Intelligence Services. Enabling occurs when Content Intelligence Client is installed or the DocApp for Web Publisher is installed.

[Table 2–26, page 122](#), lists the attributes defined for the type.

Table 2-26. Attributes defined for the CI config type

Attribute	Datatype	Single/ repeating	Description
assign_as_attribute	Boolean	S	Whether reflecting the category assignments as document attributes is enabled in the repository.
assign_as_link	Boolean	S	Whether assigning objects to folders is enabled in the repository.
auto_proc_enabled	Boolean	S	Whether automatic assignment is enabled in the repository.
auto_user	string(32)	S	User account that is used to generate and process the automatic document processing queue.
auto_workflow	Boolean	S	Whether automated object assignment initiates a confirmation workflow.
cat_owner_suggest	Boolean	S	Not currently used.

Attribute	Datatype	Single/ repeating	Description
category_object_type	string(32)	S	Name of the object type used to manage categories. This is set to dm_category by default.
ci_client_enabled	Boolean	S	Whether the Content Intelligence Client is enabled on the repository.
ci_enabled	Boolean	S	Whether Content Intelligence functionality is enabled on the repository.
ci_server_host_dev	string(32)	S	Host on which the Content Intelligence Services development server resides.
ci_server_host_prod	string(32)	S	Host on which the Content Intelligence Services production server resides.
impersonate_assign	Boolean	S	T (TRUE) allows both automatic and manual document assignments to be performed using the designated superuser account. Note: The superuser account is identified using Content Intelligence Client.
manual_proc_enabled	Boolean	S	Whether manual processing is enabled on the repository.
manual_user	string(32)	S	User account that is used to generate and process the manual document processing queue.
manual_workflow	Boolean	S	Whether manual object assignment requires a confirmation workflow.
root_admin_path	string(255)	S	Folder path of the root folder from which all administration information managed.
root_category_path	string(255)	S	Folder path of the root folder from which all categories and taxonomies are managed.

Attribute	Datatype	Single/ repeating	Description
sync_mode	string(32)	S	Identifies how data synchronization with the Content Intelligence server occurs. The only valid value is auto.
taxonomy_object_type	string(32)	S	Name of the object type used to manage taxonomies in the repository. This is set to dm_taxonomy by default.

Comment

Purpose Records a comment in a discussion

Implementation

Supertype: Richtext
 Subtypes: None
 Internal name: dmc_comment
 Object type tag: 08

A comment object represents a single comment in discussion. Comments are created and managed through Webtop. You must have installed Content Server with a Documentum Collaborative Services license to create comments in Webtop.

[Table 2-27, page 125](#), lists the attributes defined for the type.

Table 2-27. Attributes defined for the comment type

Attribute	Datatype	Single/ repeating	Description
comment_creation_date	date	S	Original date and time at which a comment was created. This is only set if the comment object represents a comment copied from the original comment.
comment_creator	string (128)	S	Name of the user who originally made the comment. This is only set if the comment object represents a comment copied from the original comment.
comment_id	integer	S	Identifying value for the comment within the topic.

Attribute	Datatype	Single/ repeating	Description
comment_modtag	integer	S	Value of the topic's last_update_modtag attribute after the most recent modification of this comment
comment_parentid	integer	S	The identifying value of the comment to which this comment is a reply or response. If the comment is not a reply to another comment, this is 0.

Completed Workflow

Purpose Records information about completed workflows

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dmc_completed_workflow
 Object type tag: 00

A completed workflow object stores information from audit trail records about a completed workflow. The objects are created by the dm_WfReporting job and used by Webtop's aggregate workflow reporting tool. To fully populate the attributes in these objects, you must be auditing all workflow events.

The object type is created by a script when Content Server is installed.

[Table 2-28, page 127](#), lists the attributes in the type.

Table 2-28. Attributes defined for the completed workflow type

Attribute	Datatype	Single/ repeating	Description
active_duration	double	S	Length of time the workflow was in the active state. This value is computed from the audit trail entries for the dm_startworkflow, dm_changestateworkflow, dm_finishworkflow, and dm_abortworkflow events.
complete_date_utc	Date	S	The date and time on which the workflow is finished or aborted. This value is taken from the audit trail entries for the dm_finishworkflow or dm_abortworkflow events.

Attribute	Datatype	Single/ repeating	Description
complete_type	integer	S	<p>Whether the workflow was finished or aborted. Valid values are:</p> <p>0, meaning the workflow was finished</p> <p>1, meaning the workflow was aborted</p> <p>The value is taken from the audit trail entries for the dm_finishworkflow or dm_abortworkflow events.</p>
component_id	ID	S	<p>Object ID of the first component in the first package.</p> <p>The value is taken from audit trail entries for the dm_addpackage event.</p>
component_name	string(80)	S	<p>Name of the first component in the first package.</p> <p>The value is taken from audit trail entries for the dm_addpackage event.</p>
creator_name	string(32)	S	<p>Name of the workflow's creator.</p> <p>The value is taken from the audit trail entries for the dm_createworkflow event.</p>
paused_duration	double	S	<p>Length of time the workflow was in the paused state.</p> <p>The value is computed by subtracting the value of active_duration from total_duration.</p>

Attribute	Datatype	Single/ repeating	Description
process_id	ID	S	Object ID of the template used to create the workflow. The value is taken from the audit trail entries for the dm_createworkflow event.
process_name	string(255)	S	Name of the workflow template used to generate the workflow instance.
start_date_utc	Date	S	Date and time of the workflow's start. The value is taken from the audit trail entries for the dm_startworkflow event.
supervisor_name	string(32)	S	Name of the workflow's supervisor The value is taken from the audit trail entries for the dm_startworkflow and dm_changeworkflowsupervisor events.
total_duration	double	S	Total length of the time the workflow existed. The value is computed by subtracting start_date_utc from complete_date_utc.
total_user_cost	double	S	Sum of the user_cost values for the completed work items in the workflow. The value is computed from the dmc_completed_workitem objects.

Attribute	Datatype	Single/ repeating	Description
total_user_time	integer	S	Sum of the user_time values for the completed work items in the workflow. The value is computed from the dmc_completed_workitem objects.
workflow_id	ID	S	Object ID of the completed workflow. The value is taken from dm_createworkflow events.
workflow_name	string(64)	S	Name of the workflow. The value is taken from the audit trail entries for the dm_createworkflow event.

Completed Workitem

Purpose Records information about completed work items in workflows.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dmc_completed_workitem
 Object type tag: 00

A completed workitem object stores information from audit trail records about a completed work item. The objects are created by the dm_WfReporting job and used by Webtop's aggregate workflow reporting tool. To fully populate the attributes in these objects, you must be auditing all workflow events.

The object type is created by a script when Content Server is installed.

[Table 2-29, page 131](#), lists the attributes of the type.

Table 2-29. Attributes defined for the completed workitem type

Attribute	Datatype	Single/ repeating	Description
act_name	string(32)	S	Name of the activity that generated the work item The value is taken from the audit trail entries for dmd_startedworkitem events
act_seqno	integer	S	Sequence number of the activity that generated the work item The value is taken from the audit trail entries for dmd_startedworkitem events

Attribute	Datatype	Single/ repeating	Description
active_duration	double	S	<p>Length of time that the work item was in the active state</p> <p>The value is computed from the audit trail entries for the dm_selectedworkitem, dm_completedworkitem, and dm_changestateactivity events.</p>
complete_action	integer	S	<p>Identifies the action chosen by the user when completing the work item. Valid values are:</p> <p>-1, meaning unknown</p> <p>0, meaning forward</p> <p>1, meaning reject</p> <p>The value is derived from the audit trail entries for the dm_setoutput event..</p>
complete_date_utc	Date	S	<p>Date and time at which the work item was completed</p> <p>The value is taken from the audit trail entries for dm_completedworkitem events.</p>
complete_state	integer	S	<p>State of the work item when the workflow was finished or aborted. Valid values are:</p> <p>0, meaning dormant</p> <p>1, meaning acquired</p> <p>2, meaning finished</p>

Attribute	Datatype	Single/ repeating	Description
creation_date_utc	Date	S	Date and time when the work item was created The value is taken from the audit trail entries for dm_startedworkitem events.
dormant_duration	double	S	Length of time, in seconds, that the work item was in the dormant state. The value is derived from the audit trail entries for dm_startedworkitem and dm_selectedworkitem events.
paused_duration	integer	S	Length of time, in seconds, that the work item was in the paused state The value is derived from the values for total_duration, dormant_duration, and active_duration.
performer_name	string(32)	S	Name of the task performer The value is taken from the audit trail entries for the dm_startedworkitem event.
process_id	ID	S	Object ID of the workflow template used to create the workflow that generated this work item. The value is taken from audit trail entries for the dm_startedworkitem event.

Attribute	Datatype	Single/ repeating	Description
process_name	string(255)	S	Name of the workflow template used to generate the workflow instance that contained this work item. The value is taken from audit trail entries for the dm_startedworkitem event.
task_priority	integer	S	The final priority of the work item. The value is taken from the audit trail entries for the dm_startedworkitem and dm_changepriority-workitem events.
total_duration	integer	S	Total length of time, in seconds, that the work item existed This is computed using start_date_utc and complete_date_utc values.
user_cost	double	S	The user cost argument value specified in the Complete method. The value is taken from audit trail entries for the dm_completedworkitem event.
user_time	integer	S	Amount of time the user spent on the work item (specified as an argument to the Complete method) The value is taken from audit trail entries for the dm_completedworkitem event.

Attribute	Datatype	Single/ repeating	Description
workflow_id	ID	S	Object ID of the workflow that generated the work item The value is taken from audit trail entries for the dm_startedworkitem events.
workitem_id	ID	S	Object ID of the work item The value is taken from audit trail entries for the dm_startedworkitem events.
wq_doc_profile	string(64)	S	Value of the a_wq_doc_profile attribute of the work item. The value is taken from audit trail entries for the dm_completedworkitem events.
wq_flag	integer	S	Value of the a_wq_flag attribute of the work item The value is taken from audit trail entries for the dm_completedworkitem events.
wq_name	string(32)	S	Value of the a_wq_name attribute of the work item The value is taken from audit trail entries for the dm_completedworkitem events.

Component

Purpose Represents a component, a set of related functionality that is used by client applications.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_component
 Object type tag: 08

A component object represents a component, a set of related functionality that is used by client applications.

[Table 2–30, page 136](#), lists the attributes defined for the type.

Table 2-30. Attributes defined for the component type

Attribute	Datatype	Single/ repeating	Description
build_technology	integer	S	The build technology used to create the component. Valid values are: 1, for ACX 16, for WIN 32 Exes 32, for Data module
com_class_id	string(38)	S	The component's COM class ID. This is required for desktop applications. It is not required for Web-based applications.
component_version	string(16)	S	The component's version.
uniq_cont_ticket	string(128)	S	Used internally.

Composite Predicate

Purpose Represents one route case condition for a workflow activity.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dmc_composite_predicate
 Object type tag: 00

A composite predicate object records a route case condition for an activity that has a transition type of automatic. Composite predicate objects are created internally when the addConditionRouteCase method is used to add route case definitions to an activity. You cannot create these objects directly.

[Table 2-31, page 137](#), lists the attributes defined for the type.

Table 2-31. Attributes defined for the composite predicate type

Attribute	Datatype	Single/repeating	Description
predicate_id	ID	R	Object IDs of transition condition objects representing the individual comparison expressions that make up the route case.
r_aspect_name	string(64)	R	Names of the TBO associated with the predicate.

Cond Expr

Purpose Stores a list of one or more Boolean expressions that are called by IF conditions in a routine or containing expression.

Implementation

Supertype: Func Expr
Subtypes: Cond ID Expr
Internal name: dm_cond_expr
Object type tag: 56

A cond expr object stores a list of one or more Boolean expressions that are called by IF conditions in a routine or containing expression. Content Server creates and manages cond expr objects. Users cannot create them.

[Table 2–32, page 138](#), lists the attribute defined for the cond expr type.

Table 2-32. Attributes defined for the cond expr type

Attribute	Datatype	Single/ repeating	Description
expression_list	ID	R	List of object IDs representing dm_func_expr objects for expressions that return a Boolean result.

Cond ID Expr

Purpose Stores a list of one or more Boolean expressions and a list of corresponding object IDs.

Implementation

Supertype: Cond Expr
 Subtypes: None
 Internal name: dm_cond_id_expr
 Object type tag: 57

A cond id expr object stores a list of one or more Boolean expressions and a list of corresponding object IDs. (The list of expressions is stored in the inherited `expression_list` attribute.) The information stored in a cond ID expr object is part of the data dictionary. A cond id expr object is created when a routine or larger expression includes one or more if conditions that return an object ID. Content Server creates and manages cond id expr objects. Users cannot create them.

[Table 2-33, page 139](#), lists the attributes defined for the type.

Table 2-33. Attributes defined for the cond ID expr type

Attribute	Datatype	Single/ repeating	Description
default_id	ID	S	Object ID to return if none of the expressions evaluate to TRUE.
id_list	ID	R	Object IDs to be returned when the corresponding expression evaluates to TRUE. For example, if the expression specified in <code>expression_list[3]</code> evaluates to TRUE, the object ID specified in <code>id_list[3]</code> is returned.

Connection Config

Purpose Describes a session's connection to a single repository.

Implementation

The connection config object type is a non-persistent type.

A connection config object describes a session's connection to a single repository. Access to this object is through the Get and Set methods, using the object's alias, connectionconfig. This object is intended for use by system administrators.

Table 2-34, page 140, lists the attributes defined for the type.

Table 2-34. Attributes defined for the connection config tType

Attribute	Datatype	Single/ repeating	Description
client_cache_size	integer	S	Defines the size of the client cache size for the subconnection. The value is taken from the session config's client_cache_size attribute setting
client_cache_write_interval	integer	S	Controls how often periodic refreshes of the client persistent cache occurs. The default value is 60 minutes.
connection_id	ID	S	Object ID of the session object for the subconnection.
connection_name	string(5)	S	Connection identifier for the subconnection
force_coherency_checks	Boolean	S	T disables the use of consistency check rules (for client persistent caches) defined in queries or a cache config object. The default is F (FALSE).

Attribute	Datatype	Single/ repeating	Description
network_requests	integer	S	Number of RPC calls sent by the client session to the server. This updates throughout the session, whenever an RPC call occurs.
nfs_enabled	Boolean	S	Indicates whether the subconnection will use NFS as its filesharing protocol. The value is derived from the nfs_enabled settings in the session config and server config objects.
r_date_format	string(40)	S	The date format that will be used to return dates to the user.
r_docbase_id	ID	S	repository ID of the repository to which the subconnection is connected.
r_docbase_name	string(120)	S	Name of the repository to which the subconnection is connected.
r_events_location	string(32)	S	Name of the location object in the repository that points to the repository's events directory.
r_mac_protocol	string(32)	S	Identifies the Macintosh file-sharing protocol used by the repository.
r_persistent_caching	Boolean	S	Whether persistent client caching is enabled for the session. T (TRUE) indicates that caching is enabled. F (FALSE) indicates that it is disabled.
r_security_mode	string(32)	S	Identifies the security mode under which the repository is running.

Attribute	Datatype	Single/ repeating	Description
r_user_name	string(32)	S	Name of the current user. This value is always the same as that found in the session config object.
secure_channel	Boolean	S	Whether the session is using a secure connection for its repository connection. T (TRUE) means the connection is secure. F (FALSE) means it is not.

Containment

Purpose Stores information about a component of a virtual document.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dmr_containment
 Object type tag: 05

A containment object stores information about a component of a virtual document. Each time a user adds a component to a virtual document, the server creates a containment object for that component. The attributes for this type are set by the Appendpart, Insertpart, and Updatepart methods. Users can query these attributes using DQL.

[Table 2–35, page 143](#), lists the attributes defined for the type.

Table 2-35. Attributes defined for the containment type

Attribute	Datatype	Single/ repeating	Description
a_contain_desc	string(255)	S	User-defined. Used by Documentum clients to manage XML documents. For more information, refer to XML support, page 165 in <i>Content Server Fundamentals</i> .
a_contain_type	string(255)	S	User-defined. Used by Documentum clients to manage XML documents. For more information, refer to XML support, page 165 in <i>Content Server Fundamentals</i> .

Attribute	Datatype	Single/ repeating	Description
component_id	ID	S	Chronicle ID of the component. (The chronicle ID is the object ID of the original version of an object. If the object has no versions, then its object ID and chronicle ID are the same.)
copy_child	integer	S	<p>Defines client behavior when the document containing the component is copied. Valid values are:</p> <p>0, meaning the decision whether to copy or reference the component is left to the user when the document is copied.</p> <p>1, meaning when the document is copied, the component is referenced in the new copy rather than actually copied.</p> <p>2, meaning when the document is copied, the component is also copied.</p>
follow_assembly	Boolean	S	If set to TRUE, directs the system to resolve a component using the component's assembly (if the component has an assembly).
order_no	double	S	Number representing the component's position within the components of the virtual document identified by parent_id.

Attribute	Datatype	Single/ repeating	Description
parent_id	ID	S	Object ID of the object that directly contains the component.
use_node_ver_label	Boolean	S	If set to TRUE for early-bound components, the server uses the early-bound symbolic label to resolve late-bound descendants of the component during assembly.
version_label	string(32)	S	Version label for the component.

Content

Purpose Stores information about a content file.

Implementation

Supertype: Persistent Object

Subtypes: None

Internal name: dmr_content

Object type tag: 06

A content object stores the information about the format and location of a content file. It also contains the information that links the content to an object. A content object also has five attributes used by Media Transformation Services to store metadata values generated by the Media Transformation Services server. If the object is stored in a content-addressed storage area, these attributes are used to record metadata values to be stored in the storage system with the content.

[Table 2-36, page 146](#), lists the attributes defined for the type.

Table 2-36. Attributes defined for the content type

Attribute	Datatype	Single/ repeating	Description
content_attr _data_type	integer	R	Data type of the media property in the corresponding index position in content_attr_name. Possible values are: 2, for a string datatype 4, for a date/time datatype 5, for a double datatype

Attribute	Datatype	Single/ repeating	Description
content_attr _date_value	date	R	The value of the metadata field identified in the corresponding index position in content_attr_name if the property is a date data type. For all other data types, the value in the corresponding index position in content_attr_date_value is NULLDATE.
content_attr_name	string(64)	R	Names of the metadata fields generated by the Media Server or specified in a Setcontentattrs or SET_CONTENT_ATTRS method for the content.
content_attr _num_value	integer	R	The value of the metadata field identified in the corresponding index position in content_attr_name if the property is an integer data type. For all other data types, the value in the corresponding index position in content_attr_num_value is NULLINT.
content_attr value	string(255)	R	The value of the metadata field identified in the corresponding index position in content_attr_name if the property is a string data type. For all other data types, the value in the corresponding index position in

Attribute	Datatype	Single/ repeating	Description
content_size	integer	S	<p>content_attr_value is NULLSTRING.</p> <p>Size, in bytes, of the content file</p> <p>This attribute cannot record a size greater than 2GB. If the file is large, examine the full_content_size attribute to obtain the full size.</p> <p>The attribute is not set if the content is stored in external storage.</p>
data_ticket	integer	S	<p>Value used internally to retrieve the content.</p> <p>An actual data ticket is created for content stored in file stores and blob stores.</p> <p>If the content is stored in turbo storage, in the content object, data_ticket is 0.</p> <p>If the content is stored in turbo storage and is too large for the i_contents attribute in the content object, the content is stored in repeating values of the i_contents attribute of a subcontent object. The data_ticket value in the content object contains the number of values in that attributes used to store the content.</p> <p>If the content is stored in external storage, data_ticket is 0.</p> <p>If the content is stored in a distributed storage area component, the data ticket value identifies a</p>

Attribute	Datatype	Single/ repeating	Description
			dmi_replica_record object for the content.
			If the content is stored in content-addressed storage, the data_ticket is a number used internally only and has no external meaning.
encoding	string(10)	S	The encoding for the format, if any
format	ID	S	Object ID of the format object describing this content's format
full_content_size	double	S	Size, in bytes, of the content file
			The attribute is not set if the content is stored in external storage.
full_format	string(64)	S	Full format specification for the content
fulltext_index	ID	R	obsolete
index_format	ID	S	obsolete
index_formats	ID	R	obsolete
index_operations	integer	R	obsolete
index_pages	integer	R	obsolete
index_parent	ID	S	obsolete
index_parents	ID	R	obsolete
index_set_times	DATE	R	obsolete
index_subtypes	string(27)	R	obsolete
is_archived	Boolean	S	Indicates whether the content has been archived.
is_offline	Boolean	S	Indicates whether the content is in the storage area.

Attribute	Datatype	Single/ repeating	Description
i_contents	For all databases except Sybase: string(2000) For Sybase: string(255)	S	<p>If the content is stored in turbo storage, this attribute contains the actual content file. If the content is too large for this attribute, the content is stored in a dmi_subcontent object and this attribute is unused.</p> <p>If the content is stored in content-addressed storage, this attribute contains the content address.</p> <p>If the content is stored in external storage, this attribute contains the token used to retrieve the content.</p> <p>The attribute cannot be selected using the API or DQL.</p>
i_encoding	string(10)	R	Contains format information used internally in the management of distributed repositories.
i_format	ID	R	Contains the information from the page and format attributes
i_full_format	string(64)	R	Contains format information used internally in the management of distributed repositories.
i_index_format	ID	R	obsolete
i_px	integer	R	Contains format information used internally in the management of distributed repositories.

Attribute	Datatype	Single/ repeating	Description
i_py	integer	R	Contains format information used internally in the management of distributed repositories.
i_pz	integer	R	Contains format information used internally in the management of distributed repositories.
i_rendition	integer	R	Contains the information in the rendition attribute. Used internally in the management of distributed repositories.
loss	integer	S	The transformation loss for the format, if any.
other_ticket	integer	S	Value used internally to retrieve the content
page	integer	R	Position of the content in each of the objects that contain it
page_modifier	string(16)	R	User-defined string to disambiguate renditions having the same format that are associated with a particular content page of a document.
parent_count	integer	S	Total number of objects that contain this content
parent_id	ID	R	Object IDs of the objects that contain the content represented by this content object

Attribute	Datatype	Single/ repeating	Description
r_content_hash	string(256)	S	Hashed value of the associated content file. This is only set if content is stored in a file store storage area and the storage area's content_hash_mode attribute is set to 1.
rendition	integer	S	Provides information about a rendition. Valid values are: 0, for original content 1, for a rendition generated by the server 2, for a rendition generated by the client 3, meaning keep the rendition when the content with which it is associated is updated or removed from the document or repository
resolution	integer	S	The resolution specification for the content's format.
set_client	string(64)	S	Name of the client machine on which the setfile was executed.
set_file	string(255)	S	Source file on the client machine that contained the content.
set_time	DATE	S	The initial value is the time at which the Setfile was executed. The time value is the time on the server machine.
storage_id	ID	S	Object ID of the store object representing the storage area that contains the content.

Attribute	Datatype	Single/ repeating	Description
transform_path	string(32)	S	The transformation path for this format
update_count	integer	R	Number of the update operation in which this content will be indexed.
x_range	integer	S	The format's range along the x axis.
y_range	integer	S	The format's range along the y axis.
z_range	integer	S	The format's range along the z axis.

Cryptographic Key

Purpose Stores a private cryptographic key.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal Name: dm_cryptographic_key
 Object type tag: 08

A cryptographic key object stores a private cryptographic key. The cryptographic key object is created automatically by Content Server. It is used by Content Server to encrypt instructions regarding content file availability for an ACS server or BOCS server. There is only one cryptographic key object in a repository. The permissions on this object give Delete permission to the owner and the members of the dm_superuser group.



Caution: These objects are for internal use only. Do not modify, remove, or add these objects.

Table 2-37, page 154, lists the attributes defined for the type.

Table 2-37. Attributes defined for the cryptographic key type

Attribute	Datatype	Single/repeating	Description
key_identifier	string(40)	S	Base 64-encoded SHA1 digest of the DER private key
key_type	integer	S	Identifies what the key used for by Content Server. The only valid value is: 1, meaning used for ACS encryption

Attribute	Datatype	Single/repeating	Description
key_value	string(1000)	S	Base 64-encoded and DBK-encrypted DER key
public_key_identifier	string(40)	S	Base 64-encoded SHA1 digest of the DER private key value in the dm_public_key_certificate.key_identifier attribute.

DD Attr Info

Purpose Contains the published data dictionary information for an attribute.

Implementation

Supertype: DD Common Info
 Subtypes: None
 Internal Name: dmi_dd_attr_info
 Object type tag: 6a

A dd attr info object contains the published data dictionary information for an attribute. You cannot create dd attr info objects directly. They are created or modified as needed when data dictionary information is published.

[Table 2–38, page 156](#) lists the basic attributes of a dd attr info object. With one exception (*attr_name*), each attribute in the table has a corresponding attribute in the type named *i_attribute_name*. The *i_* attributes are used internally by Content Server.

Table 2-38. Attributes defined for the DD attr info type

Attribute	Datatype	Single/ repeating	Description
allowed_search _ops	integer	R	<p>The search operators available for the attribute. Valid values depend on the attribute's data type.</p> <p>For string:</p> <ul style="list-style-type: none"> 1, meaning = 2, meaning <> 3, meaning > 4, meaning < 5, meaning >= 6, meaning <= 7, meaning begins with 8, meaning contains 9, meaning does not contain 10, meaning ends with 11, meaning in 12, meaning not in

Attribute	Datatype	Single/ repeating	Description
allowed_search _ops (continued)			<p>14, meaning is null 15, meaning is not null</p> <p>For Integer:</p> <p>1, meaning = 2, meaning <> 3, meaning > 4, meaning < 5, meaning >= 6, meaning <= 11, meaning in 12, meaning not in 13, meaning between 14, meaning is null 15, meaning is not null</p> <p>For Date:</p> <p>1, meaning = 2, meaning <> 3, meaning > 4, meaning < 5, meaning >= 6, meaning <= 13, meaning between 14, meaning is null 15, meaning is not null</p> <p>For Boolean and ID:</p> <p>1, meaning = 2, meaning <> 14, meaning is null 15, meaning is not null</p>
attr_name	string(32)	S	<p>Name of the attribute described by the dd attr info object.</p> <p>This attribute has no corresponding i_ attribute in the type.</p>
category_name	string(64)	S	User-defined.

Attribute	Datatype	Single/ repeating	Description
computed_dep_usr	Boolean	S	Indicates whether the values in computed_expr_dep were set by the user or calculated. TRUE means the values are user defined; FALSE means they are calculated.
computed_expr_dep	string(32)	R	List of attributes on which the computed expression referenced in cond_computed_expr depends.
cond_computed_expr	ID	R	Object ID of the dm_cond_id_expr object that contains the conditional computed expressions and the associated object IDs.
cond_value_assist	ID	S	Object ID of the cond ID expr object that contains the conditional expressions for the attribute's value assistance.
def_value_length	integer	S	Length of the attribute's default value. A zero indicates that the value is unspecified.
default_expr_builtin	integer	R	Used internally.
default_expr_kind	integer	R	Used internally.
default_expr_value	string(255)	R	Used internally to optimize default value handling.
default_search_arg	integer	S	Default value to display in conjunction with the default search operator.
default_search_op	string(255)	S	Default search operator to display to users when they search on the attribute.

Attribute	Datatype	Single/ repeating	Description
default_value	ID	R	Object ID of an expression object that resolves to the default value for the attribute. For single-valued attributes, only one value can be defined. For repeating attributes, multiple values can be defined.
domain_length	integer	S	For string attributes, the maximum length of a value.
domain_type	integer	S	Identifies the datatype of the attribute. Valid values are: 0, Boolean 1, Integer 2, String 3, ID 4, Time/Date 5, Double
format_pattern	string(64)	S	For date attributes, the pattern used to interpret values for the attribute.
format_pattern_tag	integer	S	Currently unused. (This value is always 1.)
ignore_immutable	Boolean	S	Indicates whether the attribute is changeable even if the containing object is immutable. This setting is effective only for objects of type dm_sysobject or its subtypes.
is_hidden	Boolean	S	For use by client applications.

Attribute	Datatype	Single/ repeating	Description
is_required	Boolean	S	Indicates whether users must provide a value for the attribute.
map_data_string	string(128)	R	List of possible data values for the attribute.
map_description	string(255)	R	Descriptions of the data values at the corresponding index levels in map_data_string.
map_display_string	string(128)	R	The character string to display for the data value at the corresponding index level in map_data_string.
not_null	Boolean	S	Indicates whether the attribute has a NOT NULL constraint defined for it.
not_null_enf	integer	S	Indicates who is responsible for enforcing a NOT NULL constraint. Valid values are the same as for foreign_key_enfs.
not_null_msg	string(255)	S	Error message to display when the NOT NULL constraint is violated.
read_only	Boolean	S	Indicates whether the attribute is read only.
reference_kind	integer	S	Currently unused.
super_domain_id	ID	S	Object ID of the domain from which the domain of this attribute is derived, if any.
super_domain_name	string(32)	S	Name of the domain from which the domain of this attribute is derived. If blank, the domain is the built-in domain indicated by domain_type.

Attribute	Datatype	Single/ repeating	Description
value_assist_dep	string(32)	R	Attributes on which the value assistance expressions depend.
value_assist_dep_usr	Boolean	S	Indicates whether the user defined the list of values in value_assist_dep. TRUE means the user defined the values; FALSE means the values were calculated.

DD Common Info

Purpose Contains the published data dictionary information that an object type and attribute have in common.

Implementation

Supertype: Persistent Object
 Object Subtypes: DD Attr Info; DD Type Info
 Internal Name: dmi_dd_common_info
 Object type tag: 68

A dd common info object contains the published data dictionary information that an object type and attribute have in common. You cannot create a dd common info object directly. They are created as needed when data dictionary information is published.

With four exceptions, each attribute listed in [Table 2-39, page 162](#) has a corresponding attribute named *i_attribute_name* in the object type definition. The *i_* attributes are used internally by Documentum products. The four exceptions are the attributes that identify the object type or attribute with which the dd common info object is associated. These four attributes are:

- business_policy_id
- nls_key
- state_name
- type_name

[Table 2-39, page 162](#), lists the attributes defined for the object type.

Table 2-39. Attributes defined for the DD common info type

Attribute	Datatype	Single/ repeating	Description
business_policy_id	ID	S	Object ID of a dm_policy object This attribute has a value only if the information in the dd common info object is associated with a particular lifecycle and state.

Attribute	Datatype	Single/ repeating	Description
			This attribute has no corresponding i_ attribute in the type.
comment_text	string(255)	S	User-defined.
constraint_dep_usr	Boolean	S	TRUE indicates that the values in val_constraint_dep are user-defined. FALSE means that the values are computed by Content Server.
foreign_keys	ID	R	Object IDs of any foreign key objects associated with the attribute.
foreign_key_enfs	integer	R	Indicates who is responsible for enforcing the corresponding foreign key. Valid values are: 1, Enforcement disabled 2, Enforcement by application
foreign_key_msgs	string(255)	R	Error message to display when the corresponding foreign key constraint is violated.
help_text	string(255)	S	Help text to use for the attribute.
i_dd_flags	integer	R	Reserved for future use.
ignore_constraints	Boolean	S	TRUE indicates that constraints inherited from parent types are ignored. FALSE means the inherited constraints are enforced. The default is FALSE.

Attribute	Datatype	Single/ repeating	Description
is_searchable	Boolean	S	Indicates whether the attribute is searchable. The default is TRUE, meaning the attribute can be searched.
label_text	string(64)	S	Label for the attribute or object type. Documentum provides default labels for all attributes and object types. For numerous attributes, the default is defined in a data dictionary population file. If no default is defined in the data dictionary, the default for an object type is its name. For example, for dm_sysobject, the default is dm_sysobject. The default for an attribute is the attribute's name.
life_cycle	integer	S	Indicates the current life cycle status of the type or attribute. Valid values are: 1, Currently in use 2, For future use 3, Obsolete The default is 1 (currently in use).
nls_key	string(5)	S	Locale of the information in the dd common info object. This attribute has no corresponding i_ attribute in the type.

Attribute	Datatype	Single/ repeating	Description
primary_key	ID	S	Object ID of dm_key object that contains primary key constraints for the type or attribute.
primary_key_enf	integer	S	Identifies who is responsible for enforcing the corresponding primary key. Valid values are: 1, Enforcement disabled 2, Enforcement by application
primary_key_msg	string(255)	S	Error message to display when the primary key is violated.
resync_needed	Boolean	S	Indicates whether there are unpublished changes to the object or attribute. TRUE indicates that there are unpublished changes.
state_name	string(32)	S	Name of the lifecycle state in which this data dictionary information applies. This attribute has a value only if the information in the dd common info object is associated with a particular lifecycle and state. This attribute has no corresponding i_ attribute in the type.

Attribute	Datatype	Single/ repeating	Description
type_name	string(32)	S	Name of the object type with which the information in the dd common info object is associated. This attribute has no corresponding i_ attribute in the type.
unique_keys	ID	R	Object IDs of the key objects representing any unique keys that apply to the type or attribute. The unique keys defined for the type or attribute are listed first, followed by those inherited from the supertype or super domain.
unique_key_enfs	integer	R	Indicates who is responsible for enforcing the corresponding unique key. Valid values are the same as for foreign_key_enfs.
unique_key_msgs	string(255)	R	Error message to display when the corresponding unique key is violated.
val_constraint	ID	R	Object IDs of the expression objects representing any check constraints that apply to the attribute.
val_constraint_dep	string(32)	R	Names of the attributes referenced by the constraint at the corresponding index level in val_constraint.

Attribute	Datatype	Single/ repeating	Description
val_constraint_enf	integer	R	Indicates who is responsible for enforcing the corresponding validation constraint. Valid values are the same as for foreign_key_enfs.
val_constraint_msg	string(255)	R	Error message to display when the corresponding validation constraint is violated

DD Info

Purpose Contains data dictionary information for an object type or attribute that is not dependent on locale.

Implementation

Supertype: Persistent Object

Subtypes: None

Internal name: dm_dd_info

Object type tag: 4e

A dd info object contains data dictionary information for an object type or attribute that is not dependent on locale. Content Server creates and manages dd info objects. Attributes in dd info objects are set when users add or change data dictionary information.

[Table 2-40, page 168](#) lists the attributes defined for dd info that are applicable to both object types and attributes. [Table 2-41, page 170](#) lists the attributes that apply only to object types. [Table 2-42, page 171](#) lists the dd info attributes that apply only to attributes.

Table 2-40. Attributes applicable to object types and attributes

Attribute	Datatype	Single/ repeating	Description
auditable _appevents	string(64)	R	The application-defined events that can be audited.
auditable _sysevents	string(64)	R	The system-defined events that can be audited.
foreign_keys	ID	R	Object IDs of the foreign key objects for the type or attribute.
foreign_key_enfs	integer	R	Indicates how the corresponding foreign keys are enforced. Valid values are: 1, Enforcement disabled 2, Enforcement by application

Attribute	Datatype	Single/ repeating	Description
ignore_constraints	integer	S	<p>Indicates whether the validation, key, and NOT NULL constraints defined for the type or attribute are ignored. Valid values are:</p> <p>0, Follow the constraints 1, Ignore the constraints -1, Inherit the value from the corresponding attribute in the type or attribute's supertype</p>
is_searchable	integer	S	<p>For client use. Indicates whether the attribute or type should appear in pick lists for Find dialogs. Valid values are:</p> <p>0, FALSE 1, TRUE -1, Inherit the value from the corresponding attribute in the type or attribute's supertype</p>
life_cycle	integer	S	<p>Indicates the state of the object type or attribute in the repository. Valid values are:</p> <p>1, Currently in use 2, For future use 3, Obsolete</p>
unique_keys	ID	R	<p>Objects IDs of the key objects that define unique keys for this type or attribute.</p>

Attribute	Datatype	Single/ repeating	Description
unique_key_enfs	integer	R	Indicates how the corresponding unique keys are enforced. Valid values are the same as for foreign_key_enfs.
val_constraint	ID	R	Object IDs of the expression objects corresponding to the constraints applied to the type or attribute.
val_constraint_dep	string(32)	R	The attributes on which the expression validation constraints depend.
val_constraint_enf	integer	R	Indicates how the corresponding validation constraints are enforced. Valid values are the same as for foreign_key_enfs.

Table 2-41. Attributes applicable only to object types

Attribute	Datatype	Single/ repeating	Description
comp_classifier	string(128)	R	Qualified components that can be called for this type.
default_policy_id	ID	S	Chronicle ID of the default policy object for the type.
icon_index	integer	S	The index that locates the type's icon in the icon resource file.
policy_ver_label	string(32)	S	Version label that identifies which version of the default policy object to use for the type.
primary_key	ID	S	Object ID of the primary key for the type. The primary key must be one of the unique keys defined in unique_key.

Attribute	Datatype	Single/ repeating	Description
primary_key_enf	integer	S	Indicates how the primary key is enforced. Valid values are the same as for foreign_key_enfs.
qual_comp_id	ID	R	Object IDs corresponding to the qualified components identified in comp_classifier.

Table 2-42. DD info attributes applicable to attributes

Attribute	Datatype	Single/ repeating	Description
allowed_search_ops	integer	R	A list of integers representing the valid search operators for this attribute. Valid values are: 1, for = 2, for <> 3, for > 4, for < 5, for >= 6, for <= 7, for begins with 8, for contains 9, for does not contain 10, for ends with 11, for in 12, for not in 13, for between 14, for is null 15, for is not null 16, for not
computed_expr_dep	string(32)	R	Currently unused.
cond_computed_expr	ID	S	Currently unused.

Attribute	Datatype	Single/ repeating	Description
cond_value_assist	ID	S	Object ID of the cond id expr object that contains the conditional value assistance.
default_search_arg	string(255)	S	The default value to use in conjunction with the default search operator (defined in default_search_op)
default_search_op	integer	S	The default search operator for the attribute. The operator specified must be in allowed_search_ops.
default_value	ID	R	Object IDs of the expression objects representing the attribute's default values. For single-valued attributes, default_value[0] must be the object ID of a dm_expression object corresponding to a literal of the appropriate datatype for the attribute or the NULL value appropriate for the type (for example, NULLSTRING). For repeating attributes, multiple expression object IDs are allowed, representing multiple default values for the attribute. However, object IDs representing expressions that evaluate to NULL are not allowed.

Attribute	Datatype	Single/ repeating	Description
ignore_immutable	integer	S	<p>Indicates whether the value in <code>r_immutable_flag</code> controls the attribute's changeability. This attribute setting affects only attributes of objects of type <code>dm_sysobject</code> or <code>SysObject</code> subtypes.</p> <p>If set to 1 (TRUE), the attribute is changeable regardless of the object's <code>r_immutable_flag</code> setting. Valid values are:</p> <p>0, FALSE 1, TRUE -1, Inherit value from super domain or type</p>
is_hidden	integer	S	<p>Provided for use by client applications. Valid values are the same as for <code>ignore_immutable</code>.</p>
is_required	integer	S	<p>Indicates whether a value is required for the attribute.</p> <p>If set to 1 (TRUE), the attribute must have a value before saving the object. Valid values are the same as for <code>ignore_immutable</code>.</p>
not_null	integer	S	<p>Indicates whether the attribute has the NOT NULL constraint defined for it.</p> <p>If set to 1 (TRUE), the NOT NULL constraint is specified for the attribute. Valid values are the same as for <code>ignore_immutable</code>.</p>

Attribute	Datatype	Single/ repeating	Description
not_null_enf	integer	S	Indicates how the NOT NULL constraint is enforced. Valid values are the same as for the foreign_key_enfs attribute.
parent_id	ID	S	Object ID of the aggr domain object that references the dd info object.
read_only	integer	S	Indicates whether users can read and write the attribute or only read it. If set to 1 (TRUE), users can only read this attribute, they cannot write to it. Valid values are the same as for ignore_immutable.
reference_kind	integer	S	Currently unused.
value_assist_dep	string(32)	R	Attributes on which the value assistance depends.

DD Type Info

Purpose Contains the published data dictionary information for an object type.

Implementation

Supertype: DD Common Info
 Subtypes: None
 Internal Name: dmi_dd_type_info
 Object type tag: 69

A dd type info object contains the published data dictionary information for an object type. You cannot create or modify dd type info objects directly. They are created or modified as needed when data dictionary information is published.

[Table 2-43, page 175](#), lists the attributes of the object type.

Table 2-43. Attributes defined for the DD type info type

Attribute	Datatype	Single/ repeating	Description
attr_domain_name	string(32)	R	Currently unused.
attr_domain_id	ID	R	Currently unused.
auditable _appevents	string(64)	R	The application-defined events that can be audited.
auditable _sysevents	string(64)	R	The system-defined events that can be audited.
comp_classifier	string(128)	R	The classifiers for the qualified components that can be executed against instances of the type.
default_policy_id	ID	S	Object ID of the dm_policy object representing the default lifecycle for the type.
i_attr_domain_id	integer	S	For internal use.
i_attr_domain _name	integer	S	For internal use.

Attribute	Datatype	Single/ repeating	Description
i_comp_classifier	integer	S	For internal use.
i_default_policy_id	Boolean	S	For internal use.
i_icon_index	Boolean	S	For internal use.
i_policy_version_label	Boolean	S	For internal use.
i_qual_comp_id	integer	S	For internal use.
icon_index	integer	S	The index that locates the icon for the type in the icon resource file.
map_data_string	string(128)	R	List of possible data values for the attribute.
map_description	string(255)	R	Descriptions of the data values at the corresponding index levels in map_data_string.
map_display_string	string(128)	R	The character string to display for the data value at the corresponding index level in map_data_string
policy_ver_label	string(32)	S	Version label of the object identified in default_policy_id. Note: This value is provided by the user, not derived from the object by Content Server
qual_comp_id	ID	R	Object IDs of the qual comp objects representing the component routines that can be executed against instances of the type.
r_has_check	Boolean	R	For internal use
r_has_constraint	Boolean	R	For internal use
r_has_default	Boolean	R	For internal use
r_has_dependency	Boolean	R	For internal use

Attribute	Datatype	Single/ repeating	Description
r_has_foreign_key	Boolean	R	For internal use
r_has_ignore _immutable	Boolean	R	For internal use
r_has_not_null	Boolean	R	For internal use
r_has_primary_key	Boolean	R	For internal use
r_has_unique_key	Boolean	R	For internal use
r_has_value_assist	Boolean	R	For internal use
scope_config	ID	R	Object IDs of the scope config object for the object type.

Display Config

Purpose Defines the display configuration for a group of attributes.

Implementation

Supertype: None
 Subtypes: None
 Internal name: dm_display_config
 Object type tag: 6b

A display config object defines the display configuration for a group of attributes. Display config objects are referenced by scope config objects, which define the context within which the display configuration is used. Both display config objects and scope config objects are used by client applications. Content Server does not use these objects.

Table 2-44, page 178, lists the attributes defined for the type.

Table 2-44. Attributes defined for the display config type

Attribute	Datatype	Singe/ repeating	Description
attribute_ display_hint	integer	R	<p>Controls the display of attributes in the user interface. These values are interpreted by client application, not Content Server.</p> <p>Values used by Desktop Client and Webtop are:</p> <ul style="list-style-type: none"> • 0, meaning do not display a separator before an attribute • 1, meaning display a separator before an attribute <p>Values used by Webtop only are:</p> <ul style="list-style-type: none"> • 2, meaning display the attribute if the user requests it and do not use a separator • 3, meaning display the attribute if the user requests it and use a separator <p>The hint at a particular index position is applied to the attribute named at</p>

Attribute	Datatype	Singe/ repeating	Description
			the corresponding index position in attribute_name.
attribute_name	string(40)	R	List of attributes names. The attributes must be defined for the object type identified in attribute source. The order in which the attributes are listed determines the order in which they are displayed in the client application.
attribute_source	string(27)	S	The object type for which the attributes listed in attribute_name are defined.
fixed_display	Boolean	S	Indicates whether attributes can be added to attribute_name or display_hints changed. T means that the attribute list and hint cannot be changed. F means that the attribute list and hint can be changed. The default is F.
i_config_identifier	string(20)	S	Uniquely identifies a display config object across repositories. The format of the value is <i>dm_r_object_id</i> , where <i>r_object_id</i> is the object ID of the display config object.
			This is used internally to manage load operations.
object_name	string(64)	S	Name of the display config object. This must be unique within the scope in which the display config object is used. Note: Uniqueness is enforced by the client, not Content Server.

Distributed Store

Purpose Contains information about a storage area.

Implementation

Supertype: Store
Subtypes: None
Internal name: dm_distributedstore
Object type tag: 2c

A distributed store object contains information about a storage area. A distributed storage area points to component storage areas and is used to implement a repository that has distributed content. All but three of its attributes are inherited from its supertype, dm_store.

Note: For information about distributed storage areas, refer to [Distributed storage areas](#), page 26 in the *Documentum Distributed Configuration Guide*.

[Table 2-45](#), page 180 lists the attributes defined for the distributed store type.

Table 2-45. Attributes defined for the distributed store type

Attribute	Datatype	Single/ repeating	Description
change_record_id	ID	S	Used internally to manage structural changes.
epoch_number	integer	S	Records a count of how many structural changes (adding or removing component areas) have been made to the distributed storage area.
only_fetch_close	Boolean	S	Indicates if a server can fetch files from component storage areas in the distributed storage area that are defined as far for the server.

Docbase Config

Purpose Contains configuration information about a repository.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_docbase_config
 Object type tag: 3c

A docbase config object contains configuration information about a repository. Each repository must have a single docbase config object whose object name matches the name of the repository.

[Table 2-46, page 181](#), lists the attributes defined for the type.

Table 2-46. Attributes defined for the docbase config type

Attribute	Datatype	Single/ repeating	Description
a_bpaction_run_as	string(32)	S	<p>Defines which user account is used to run lifecycle (business policy) actions. Valid values are:</p> <ul style="list-style-type: none"> • session_user (default) • superuser • lifecycle_owner • <i>user name</i> <p>The first three values are keywords that must be entered as shown. For the last, <i>user name</i>, specify the user name of a repository user.</p>

Attribute	Datatype	Single/ repeating	Description
auth_deactivation_ interval	integer	S	<p>Length of time between a user's account deactivation and automatic re-activation. If this is 0, the account is not automatically re-activated.</p> <p>The value is specified in minutes.</p> <p>The default is 0.</p>
auth_failure_ interval	integer	S	<p>Length of time, in minutes, in which consecutive failed login authorizations will cause a user's account to be deactivated. The number of failed attempts that must occur within the interval to trigger deactivation is determined by the max_auth_attempts attribute.</p> <p>The default is 0, meaning that deactivation always occurs when the maximum number of consecutive failed login attempts is reached, regardless of how long that takes.</p>
auth_protocol	string(32)	S	<p>On Windows platforms, if set to domain_required, it indicates that the repository is running in domain-required mode. If the repository is not using domain-required mode, this is blank.</p> <p>On UNIX platforms, if you are authenticating users against a Windows domain, set this to unix_domain_used</p> <p>Otherwise, this attribute is blank for repositories</p>

Attribute	Datatype	Single/ repeating	Description
check_client_version	Boolean	S	<p>running on a UNIX platform.</p> <p>T means that the repository servers will not accept connections from clients older than the version level specified in the oldest_client_version attribute.</p> <p>F means that the servers accept connections from any client version.</p> <p>The default is F.</p>
client_pcaching_change	integer	S	<p>Controls persistent client cache flushing. Incrementing this value forces clients to flush all persistent caches on start up.</p>
client_pcaching_disabled	Boolean	S	<p>T (TRUE) disables persistent client caching for sessions with the repository. The default is F (caching is allowed).</p>
dd_locales	string(5)	R	<p>Data dictionary locales recognized by the server.</p> <p>Setting this attribute requires you to execute a Reinit method on the server to make the change visible.</p>

Attribute	Datatype	Single/ repeating	Description
default_app_permit	integer	S	<p>Default user permission level for application-controlled objects accessed through an application that doesn't own the object. Values are:</p> <p>2, Browse permission 3, Read permission 4, Relate permission 5, Version permission 6, Write permission 7, Delete permission</p> <p>The default value is 3, Read permission.</p>
dir_user_sync_on_demand	Boolean	S	Reserved for future use
effective_date	date	S	<p>Obsolete in 5.2 repositories. In pre-5.2 repositories, this is used to force persistent client cache flushes. Refer to Using persistent client caching with a pre-5.2 Content Server, page 200 in the <i>Content Server Administrator's Guide</i> for details. The default value is NULLDATE.</p>
folder_security	Boolean	S	<p>Indicates whether the repository is running with folder security on or off. The default is T, meaning that security is turned on.</p>
fulltext_install_locs	string(32)	R	Name of the location object that points to a fulltext installation.
i_crypto_key	string(255)	S	The encryption key for the repository

Attribute	Datatype	Single/ repeating	Description
i_ticket_crypto_key	string(255)	S	The encrypted login ticket key
index_store	string(80)	S	Name of the RDBMS tablespace where you want to store type indexes.
login_ticket_cutoff	Date	S	Defines the earliest possible creation date for valid login tickets. Tickets issued before this date are not valid in this repository. The default value is NULLDATE, meaning there is no cutoff date for login tickets.
mac_access_protocol	string(32)	S	Identifies the type of file sharing protocols in use for Macintosh clients. Valid values nt, double, ushare, or none. Use none only if you have no Macintosh clients.
macl_security_disabled	Boolean	S	If T (TRUE), Content Server enforces only the AccessPermit and ExtendedPermit entries in an ACL. If the server is installed with a Trusted Content Services license, the default is F (FALSE), meaning the server enforces all entries an ACL If the server is not installed with a Trusted Content Services license, the default is T.

Attribute	Datatype	Single/ repeating	Description
max_auth_attempt	integer	S	<p>Maximum number of unsuccessful login attempts allowed.</p> <p>The default is 0, which means the feature is disabled.</p>
object_name	string(255)	S	<p>Contains the name of the repository. (This attribute is inherited from dm_sysobject.)</p> <p>A repository name must be all ASCII characters and <=32 characters in length. The name docu is reserved by Documentum.</p>
offline_checkin_flag	integer	S	<p>Used by some Documentum clients to determine whether a client dialog box is used to check in content or the user's local check-in setting takes precedence. Valid values are:</p> <p>0, meaning use a client dialog box for check ins</p> <p>1, meaning allow the user's local check-in setting to take precedence</p> <p>The default value is 0.</p>

Attribute	Datatype	Single/ repeating	Description
offline_sync_level	integer	S	<p>Identifies the level of repository access for offline synchronization. This is used primarily by some Documentum clients. Valid values are:</p> <p>0, meaning none</p> <p>1, meaning one-way access</p> <p>2, meaning role-based access</p> <p>The default value is 0.</p> <p>Note: Refer to the client documentation for information about the actual use of this attribute and role-based synchronization.</p>
oldest_client_version	string(32)	S	<p>Version number of the oldest Documentum client that will access this repository.</p> <p>This must be set manually. It is used by the DFC to determine how to store chunked XML documents. If check_client_version is set to T, then this value is also used to identify the oldest client version level that may connect to the repository.</p>
r_address_partitions	integer	S	<p>This is no longer used.</p>

Attribute	Datatype	Single/ repeating	Description
r_dbms_name	string(32)	S	Name of the RDBMS. Valid values on Windows platforms are: Oracle, SQL Server, and DB2. Valid values on Unix platforms are: Oracle, Sybase, and DB2.
r_docbase_id	integer	S	The repository ID as a decimal value.
r_ending_partition	integer	S	This is no longer used.
r_federation_name	string(120)	S	Name of the federation to which the repository belongs, if any. The name is taken from the value of the object_name attribute in the dm_federation object.
r_module_mode	integer	R	Used internally
r_module_name	string(32)	R	Used internally
r_starting_partition	integer	S	This is no longer used.
r_storage_mode	integer	S	Used internally
richmedia_enabled	Boolean	S	Indicates whether the Content Server for this repository can process rich-media content. This value is set to TRUE during installation of the Media Server.
security_mode	string(32)	S	Defines the security level for the repository. Valid values are acl or none.

Attribute	Datatype	Single/ repeating	Description
trust_by_default	Boolean	S	<p>Whether the repository accepts login tickets and application access control tokens from other repositories that have the same login ticket key (LTK) as this repository.</p> <p>T (TRUE) means the repository accepts login tickets and tokens from other repositories that share its LTK.</p> <p>F (FALSE) means that it only accepts login tickets and tokens from other repositories sharing its LTK if they appear in the trusted_docbases list.</p> <p>The default value is F.</p>
trusted_docbases	string(255)	R	<p>Name of the repositories from which login tickets and tokens are accepted. The repositories must have the same login ticket key as the repository represented by this docbase config object.</p>
wf_package_control_enabled	Boolean	S	<p>Controls whether Content Server exposes the object names of components in workflow packages.</p> <p>T allows Content Server to set the r_component_name attribute in dmi_package objects. F disallows setting the r_component_name attribute in package objects.</p> <p>If this attribute is F, the setting at the workflow level, in the package_control attribute of the dm_process object is ignored.</p>

Attribute	Datatype	Single/ repeating	Description
-----------	----------	----------------------	-------------

The default value is F.

Docbase Locator

Purpose Contains information about the repositories known to a connection broker.

Implementation

The docbase locator type is a non-persistent type.

A docbase locator object is an object that is constructed and returned by a connection broker in response to a Getdocbasemap method call. The information for a single repository appears at corresponding index levels in the repeating attributes. For example, the name of the repository whose ID appears in `r_docbase_id[0]` is found in `r_docbase_name[0]`, and its description is found in `r_docbase_description[0]`.

[Table 2-47, page 191](#), lists the attributes defined for the type.

Table 2-47. Attributes defined for the docbase locator type

Attribute	Datatype	Single/ repeating	Description
auth_protocol	string(32)	S	<p>On Windows platforms, if set to domain required, indicates that the repository is running in domain-required mode. If the repository is not using domain-required mode, this attribute is blank.</p> <p>On UNIX platforms, if you are authenticating users against a Windows domain, this is set to <code>unix_domain_used</code>.</p> <p>Otherwise, this attribute is blank for</p>

Attribute	Datatype	Single/ repeating	Description
			repositories running on a UNIX platform.
i_docbroker _version	string(32)	S	Version number of the responding connection broker.
i_host_addr	string(32)	S	The IP address of the host on which the responding connection broker resides.
i_host_name	string(128)	S	Name of the host machine on which the responding connection broker resides.
i_port_number	integer	S	Port number of the responding connection broker.
r_docbase _description	string(128)	R	Verbose name or description of the repository.
r_docbase_id	ID	R	The internal ID of the repository.
r_docbase_name	string(32)	R	Name of the repository.
r_federation_name	string(32)	R	Name of the federation to which this docbase belongs, if any.
r_govern_docbase	string(32)	R	For each repository participating in a federation, the name of the governing repository in the federation.

Attribute	Datatype	Single/ repeating	Description
r_object_id	ID	S	Object ID of the docbase locator object.
r_server_version	string(32)	R	Version numbers of the servers.

Docbroker Locator

Purpose Contains information about each connection broker that the client DMCL can access.

Implementation

The docbroker locator type is a non-persistent type.

A docbroker locator object contains information about each connection broker that the client DMCL can access. The object is constructed and returned by the client library in response to a `Getdocbrokermap` method call. The information for a single connection broker appears at corresponding index levels in the attributes. For example, the protocol for the connection broker whose `host_name` appears in `host_name[0]` is found in `network_protocol[0]` and its port number is found in `port_number[0]`.

[Table 2-48, page 194](#), lists the attributes defined for the type.

Table 2-48. Attributes defined for the docbroker locator type

Attribute	Datatype	Single/ repeating	Description
<code>host_name</code>	<code>stringstring((128)</code>	R	Name of the host machine on which the connection broker resides
<code>network_protocol</code>	<code>string(12)</code>	R	Protocol for a given connection broker
<code>port_number</code>	<code>integer</code>	R	Number of the port on the host machine that the connection broker uses for communication
<code>time_out</code>	<code>integer</code>	R	Time, in seconds, that a server waits for a response from the connection broker before forwarding the request to a backup connection broker.

Docset

Purpose Represents a set of documents to be processed by Content Intelligence Services.

Implementation

Supertype: Document
 Subtypes: None
 Internal name: dm_docset
 Object type tag: 09

A docset object represents a set of documents to be processed by Content Intelligence Services. The documents are defined by a query recorded in the docset content. The content's format is crtext.

[Table 2-49, page 195](#), lists the attributes defined for the type.

Table 2-49. Attributes defined for the docset type

Attribute	Datatype	Single/ repeating	Description
description	string(255)	S	User-defined description of the document set
last_run_id	ID	S	Object ID of the docset run object last used to process this document set.
object_name	string(255)	S	Document set name
qualifiers	string(255)	R	List of conditions that documents must meet for inclusion in the docset.

Attribute	Datatype	Single/ repeating	Description
queue_bound	Boolean	S	Controls whether documents placed on the processing queue can be considered for inclusion in this docset. Setting this to T allows documents on the processing queue to be considered for inclusion in this docset.
query_type	string(32)	S	Identifies the type of query. The only valid value is dql.

Docset Run

Purpose Represents an executable collection of documents.

Implementation

Supertype: Document
 Subtypes: None
 Internal name: dm_docset_run
 Object type tag: 09

A docset run object represents an executable collection of documents. Docset run objects are used by Content Intelligence Services.

[Table 2-50, page 197](#), lists the attributes defined for the type and those inherited attributes that have a meaning specific to this type.

Table 2-50. Attributes defined for the docset run type

Attribute	Datatype	Single/ repeating	Description
docset_id	ID	S	Object ID of the document set associated with the processing run
estimated_count	integer	S	Estimated number of documents to process in the run
object_name	string(255)	S	Name of the docset run
processed_count	integer	S	Actual number of documents processed by the run
run_interval	integer	S	Not currently used
run_mode	integer	S	Not currently used
run_now	Boolean	S	T (TRUE) invokes processing immediately. The default is F (FALSE).
run_owner	string (32)	S	Name of the category manager who started the processing.

Attribute	Datatype	Single/ repeating	Description
run_status	string(32)	S	Status of the processing run. Valid values are: not_started started scheduled completed failed
run_type	string(32)	S	Indicates whether this is a test run or an actual run. Valid values are: test active
start_time	Date	S	Not currently used
taxonomy_id	ID	R	Object IDs of all taxonomies included in the processing. Currently, only one taxonomy in a run is supported.
title	string(255)	S	Description of status state. The description may be user or system-defined.

Document

Purpose Contains information about a document.

Implementation

Supertype: SysObject

Subtypes: Email Message, Esign Template, XML Config, XML Custom Code, XML Style Sheet, XML Zone, Notepage, TCF Activity, TCF Activity Template, XFM Form, XFM Instance

Internal name: dm_document

Object type tag: 09

Documents are the objects most commonly manipulated by users in the Documentum system. Documents can be simple documents or virtual documents. In a simple document, the content generally seen by a user is in one or more content files associated with the document. A virtual document is a document that is composed of components that are either simple documents or other virtual documents, and the content that users see is the content files associated with these components.

A virtual document can also have associated content files. That is, in addition to its components, a virtual document can have an associated content file (or files). However, when you open a virtual document for viewing or editing through WorkSpace, WorkSpace opens the component.

All content files associated directly with a document must have the same file format. The components of virtual documents can have a mixture of formats.

Documents inherit all of their attributes from their supertype, the SysObject type.

Domain

Purpose Describes the properties of an attribute's domain.

Implementation

Supertype: Persistent Object

Subtypes: Aggr Domain

Internal name: dm_domain

Object type tag: 50

A domain object describes the properties of an attribute's domain. The information contained in a domain object is part of the data dictionary. The domain information includes the attribute's datatype, length (if a string type), localized label text, and any expressions used to populate the attribute. Domain objects are created and managed by the server and cannot be created by users.

[Table 2-51, page 200](#), lists the attributes defined for the type.

Table 2-51. Attributes defined for the domain type

Attribute	Datatype	Single/ repeating	Description
dd_info	ID	S	Object ID of the dd info object containing the domain's information that does not depend on locale.
domain_length	integer	S	For string domains, the maximum length of the domain.
domain_name	string(32)	S	Currently unused.
domain_type	integer	S	Indicates the domain's datatype. Valid values are: 0, Boolean 1, Integer 2, String 3, ID 4, Time/Date 5, Double

Attribute	Datatype	Single/ repeating	Description
nls_dd_info	ID	R	Object IDs of the nls dd info objects containing the domain's information that is dependent on locale.
nls_default	integer	S	Currently unused.
nls_keys	string(64)	R	Identifies the locales for which localized information is available. The values are the standard locale codes; for example, en or ja. The localized domain information for each locale is recorded in the nls dd info object identified in the corresponding index position of the nls_dd_info attribute. This attribute is set when a locale is populated.
parent_id	ID	S	Object ID of the aggr domain object that references this domain in its attr_domain_id attribute.
re_sync_dd	Boolean	S	Indicates whether the domain's attribute values accurately reflect the values in the associated type definition. If TRUE, indicates that the domain may not accurately reflect the information in the type definition.
super_domain_id	ID	S	Object ID of the domain object from which this domain is derived, if any.
super_domain_name	string(32)	S	Currently unused.

DSM Application

Purpose Contains a eCTD application.

Implementation

Supertype: Folder
Subtypes: None
Internal name: dmc_dsm_application
Object type tag: 0b

A dsm application object contains all the objects that comprise an eCTD application submitted and managed by EMC Documentum Submissions Manager. The object type is installed when you install the DSM DocApp in a repository.



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

[Table 2-52, page 202](#), lists the attributes defined for the type.

Table 2-52. Attributes defined for the dmc dsm application type

Attribute	Datatype	Single/ repeating	Description
last_published_name	string(255)	S	Name of the last published submission

DSM Backbone

Purpose Represents the root document of the virtual document that represents a submission.

Implementation

Supertype: dsm section
 Subtypes: DSM Stf Backbone, DSM M1 Backbone
 Internal name: dmc_dsm_backbone
 Object type tag: 09

A dsm backbone object stores the index.xml file for a submission. This file is the root document of the virtual document that represents the submission in the repository. The object type is installed when the EMC Documentum Submissions Manager DocApp is installed in a repository.



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

[Table 2-53, page 203](#)

Table 2-53. Attributes defined for the dmc dsm backbone type

Attribute	Datatype	Single/ repeating	Description
document_id	ID	R	Used internally
dsm_data_model_version	string(32)	S	Used internally
is_import_post_processed	Boolean	S	Used internally
is_published	Boolean	S	Used internally
last_used_leaf_id	string(128)	S	Used internally
leaf_id	string(128)	R	Used internally

DSM Doc Properties

Purpose Stores the values of the eCTD attributes for an eCTD content file.

Implementation

Supertype: Relation
 Subtypes: None
 Internal name: dmc_dsm_doc_properties
 Object type tag: 37

A dsm doc properties object records the eCTD attributes of an eCTD content file. The values are version-specific, so there is one dsm doc properties object for each version of the document that contains the file. The object type is installed when the EMC Documentum Submissions Manager DocApp is installed.



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

[Table 2-54, page 204](#)

Table 2-54. Attributes defined for the dmc dsm doc properties type

Attribute	Datatype	Single/ repeating	Description
application_version	string(128)	S	Version of the software application used to create the file
checksum_type	string(128)	S	Algorithm used to generate the checksum
checksum_value	string(128)	S	Checksum value for the file
document_keywords	string(4000)	S	Keywords specified for the file
document_title	string(128)	S	Title of the document

Attribute	Datatype	Single/ repeating	Description
font_library	string(128)	S	Commercial name of the font or font set used to create the document
last_checksum_time	Date	S	Timestamp of the last checksum for this document
version_identifier	string(128)	S	Submitter's internal version number or version identification for the report

DSM Drug Product

Purpose Records information about a drug product specified in an eCTD section element.

Implementation

Supertype: DSM Section
 Subtypes: None
 Internal name: dmc_dsm_drug_product
 Object type tag: 09

A dsm drug product object records the attribute values specified in a <m2-3-drug-product> or <m3-2-p-drug-product> section element. The object type is installed when the EMC Documentum Submissions Manager DocApp is installed.



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

[Table 2-55, page 206](#), lists the attributes defined for the type.

Table 2-55. Attributes defined for the dmc dsm drug product type

Attribute	Datatype	Single/ repeating	Description
dosage_form	string(128)	S	The form in which the drug was given to study subjects.
manufacturer_name	string(128)	S	Name of the drug's manufacturer
product_name	string(128)	S	Name of the product

DSM Drug Substance

Purpose Records information about a drug substance specified in an eCTD section element.

Implementation

Supertype: DSM Section
 Subtypes: None
 Internal name: dmc_dsm_drug_substance
 Object type tag: 09

A dsm drug substance object records the attribute values specified in a <m2-3-s-drug-substance> or <m3-2-s-drug-substance> section element. The object type is installed when the EMC Documentum Submissions Manager DocApp is installed.



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

[Table 2-56, page 207](#), lists the attributes for the type.

Table 2-56. Attributes defined for the dmc dsm drug substance type

Attribute	Datatype	Single/ repeating	Description
manufacturer_name	string(128)	S	Name of the drug's manufacturer
substance_name	string(128)	S	Name of the substance

DSM Excipient

Purpose Records information about an excipient substance specified in an eCTD section element.

Implementation

Supertype: DSM Section

Subtypes: None

Internal name: dmc_dsm_excipient

Object type tag: 09

A dsm excipient object records the attribute values specified in a <m3-2-a-1-control-of-excipients> section element. The object type is installed when the EMC Documentum Submissions Manager DocApp is installed.



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

[Table 2-57, page 208](#)

Table 2-57. Attributes defined for the dmc dsm excipient type

Attribute	Datatype	Single/ repeating	Description
excipient_substance	string(128)	S	Name of the excipient substance

DSM Facilities Equip

Purpose Records the attribute values specified in an eCTD section element.

Implementation

Supertype: DSM Section
 Subtypes: None
 Internal name: dmc_dsm_facilities equip
 Object type tag: 09

A dsm facilities equip object records the attribute values in a <m3-2-a-1-facilities-and-equipment> eCTD section element. The object type is installed when the EMC Documentum Submissions Manager DocApp is installed.



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

[Table 2-58, page 209](#)

Table 2-58. Attributes defined for the dmc dsm facilities equip type

Attribute	Datatype	Single/ repeating	Description
dosage_form	string(128)	S	The form in which the drug was given to study subjects.
manufacturer_name	string(128)	S	Name of the drug's manufacturer
product_name	string(128)	S	Name of the product
substance_name	string(128)	S	Name of the substance

DSM Indication

Purpose Records the attribute values specified in an eCTD section element.

Implementation

Supertype: DSM Section
Subtypes: None
Internal name: dmc_dsm_indication
Object type tag: 09

A dsm indication object records the attribute values specified in a <m5-3-5-reports-of-
efficacy-and-safety-studies> or <m2-7-3-summary-of-clinical-efficacy> eCTD section
element. The object type is installed when the EMC Documentum Submissions Manager
DocApp is installed.



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

[Table 2-59, page 210](#)

Table 2-59. Attributes defined for the dmc dsm indication type

Attribute	Datatype	Single/ repeating	Description
indication_name	string(128)	S	Name of the indication

DSM M1 Backbone

Purpose Represents as regional backbone.

Implementation

Supertype: DSM Backbone
Subtypes: None
Internal name: dmc_dsm_m1_backbone
Object type tag: 09

A dsm m1 backbone object represents a regional backbone file. The object type is installed when the EMC Documentum Submissions Manager DocApp is installed. The object type inherits all its attributes from its parent type. It has no defined attributes.



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

DSM Safety Eval

Purpose Records the attribute values specified in an eCTD section element.

Implementation

Supertype: DSM Section
 Subtypes: None
 Internal name: dmc_dsm_safety_eval
 Object type tag: 09

A dsm safety eval object records the attribute values in a <m3-2-a-2-adventitious-agents-safety-evaluation> eCTD section element. The object type is installed when the EMC Documentum Submissions Manager is installed.



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

[Table 2-60, page 212](#)

Table 2-60. Attributes defined for the dmc dsm safety eval type

Attribute	Datatype	Single/ repeating	Description
dosage_form	string(128)	S	The form in which the drug was given to study subjects.
manufacturer_name	string(128)	S	Name of the drug's manufacturer
product_name	string(128)	S	Name of the product
substance_name	string(128)	S	Name of the substance

DSM Sect Doc Attributes

Purpose Stores the values of attributes specific to a section-document relationship.

Implementation

Supertype: Relation
 Subtypes: None
 Internal name: dmc_dsm_sect_doc_attributes
 Object type tag: 37

A dsm sect doc attributes object stores the values of attributes specific to a section-document relationship. The relation_name for objects of this type is always dmc_dsm_section_to_doc. The object type is installed when the EMC Documentum Submissions Manager DocApp is installed.



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

[Table 2-61, page 213](#)

Table 2-61. Attributes defined for the dmc dsm sect doc attributes type

Attribute	Datatype	Single/ repeating	Description
folder_path	string(1024)	S	Repository location of the document
leaf_id	string(128)	S	Unique identifier for this file in the XML instance.
modified_file	string(128)	S	Identifies the location of the document being modified. The value in this attribute points the the index.xml file and the leaf ID of the file being altered.

Attribute	Datatype	Single/ repeating	Description
modified_file_id	string(32)	S	Object ID of the document specified in modified_file
operation_name	string(32)	S	Identifies the operation to be performed on the file to be modified. Valid values are: new append replace delete
submission_id	string(32)	S	Object ID of the submission folder to which the document belongs.

DSM Section

Purpose Records information used to display a submission in Virtual Document Manager.

Implementation

Supertype: Document
 Subtypes: DSM Backbone, DSM Drug Product, DSM Drug Substance, DSM Excipient, DSM Facilities Equip, DSM Safety Eval, DSM Indication, DSM Section Extension, DSM Stf Section
 Internal name: dmc_dsm_section
 Object type tag: 09

A dsm section object contains information used by EMC Documentum Submissions Manager to manage the display of the submission in the Virtual Document Manager. The object type is installed when the EMC Documentum Submissions Manager DocApp is installed..



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

[Table 2–62, page 215](#), lists the attributes defined for the type.

Table 2-62. Attributes defined for the dmc dsm section type

Attribute	Datatype	Single/ repeating	Description
ectd_profile	string(32)	S	Identifies the associated DSM configuration profile governing DSM processing This attribute may be empty. If so, the value is derived from the first parent in the virtual document that contains a value for this attribute.

Attribute	Datatype	Single/ repeating	Description
section_id	string(128)	S	Name of the section. If the section has attributes, the attribute values are concatenated with the section name.
section_name	string(128)	S	The eCTD section name. The exact value depends on whether the section has attributes.

DSM Section Extension

Purpose Records eCTD element extensions.

Implementation

Supertype: DSM Section
 Subtypes: None
 Internal name: dmc_dsm_section_extension
 Object type tag: 08

A dsm section extension object records eCTD element extensions. Each pair at the same index position in the attributes represents a nested level of <node-extension>. The object type is installed when the EMC Documentum Submissions Manager DocApp is installed.



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

[Table 2-63, page 217](#)

Table 2-63. Attributes defined for the dmc dsm section extension type

Attribute	Datatype	Single/ repeating	Description
extension_id	string(128)	R	List of extension identifiers.
extension_title	string(128)	R	Titles of the extensions. The title at any particular index position is associated with the identifier in the corresponding position in extension_id.

DSM Stf Backbone

Purpose Represents the backbone of a study tagging (stf.xml) file.

Implementation

Supertype: DSM Backbone
 Subtypes: None
 Internal name: dmc_dsm_stf_backbone
 Object type tag: 09

A dsm stf backbone object represents the root document of the virtual document that represents an stf.xml file in the repository. That virtual document is a component of the virtual document whose root is represented by a dsm backbone object. The object type is installed when the EMC Documentum Submissions Manager DocApp is installed.



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

[Table 2-64, page 218](#), lists the attributes defined for the type. For the repeating attributes, the values recorded at a particular index position represent one category name attribute element.

Table 2-64. Attributes defined for the dmc dsm stf backbone type

Attribute	Datatype	Single/ repeating	Description
category_name	string(128)	R	Category names identified in the <category name> attributes in a <study-identifier> element.
category_info_type	string(32)	R	Information type identified in <category name> attributes in a <study-identifier> element.

Attribute	Datatype	Single/ repeating	Description
category_value	string(128)	R	The value specified in <category name> attributes in a <study-identifier> element.
study_id	string(128)	S	Value of the <study-id> attribute in a <study-identifier> element.
study_title	string(128)	S	Value of the <title> attribute in a <study-identifier> element.

DSM Stf Section

Purpose Records the attributes of an eCTD content block within an STF document.

Implementation

Supertype: DSM Section
 Subtypes: None
 Internal name: dmc_dsm_stf_section
 Object type tag: 09

A dsm sft section object stores the values of attributes of an eCTD content block from an sft.xml document. The object type is installed when the EMC Documentum Submissions Manager is installed.



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

[Table 2-65, page 220](#), lists the attributes defined for the type.

Table 2-65. Attributes defined for the dmc dsm stf section type

Attribute	Datatype	Single/ repeating	Description
block_title	string(128)	S	The title value of the content block.
property_info_type	string(32)	R	The information types specified in property attributes.
property_name	string(128)	R	The names specified in n property attributes.
property_value	string(128)	R	The values specified in property attributes.
subject_name	string(128)	R	Name of the subject
subject_info_type	string(32)	R	The information type for the subject.

DSM Study Attributes

Purpose Records the study-related attributes for an STF study report.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dmc_dsm_study_attributes
 Object type tag: 08

A dsm study attributes object records the values of the study attribute for an STF study report. The objects are related to the document that stores the report. The relation name is always dmc_dsm_doc_to_study_attri. The object type is installed when the EMC Documentum Submissions Manager DocApp is installed.



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

Table 2-66, page 221 lists the attributes defined for the type.

Table 2-66. Attributes defined for the dmc dsm study attributes type

Attribute	Datatype	Single/ repeating	Description
property_info_type	string(32)	R	The information types specified in property attributes
property_name	string(128)	R	The names specified in n property attributes
property_value	string(128)	R	The values specified in property attributes
study_title	string(128)	S	The title of the study

Attribute	Datatype	Single/ repeating	Description
subject_name	string(128)	R	Names of the subjects
subject_info_type	string(32)	R	The information types for the subjects.

DSM Study Report

Purpose Relates a study report within an STF to a logical content file in the main backbone.

Implementation

Supertype: Relation
 Subtypes: None
 Internal name: dmc_dsm_study_report
 Object type tag: 37b

A dsm study report object establishes a relationship between a study report within an STF and a logical content file in the main backbone. The objects are related to the document that stores the report. The relation name is always dmc_dsm_doc_to_stf. The object type is installed when the EMC Documentum Submissions Manager DocApp is installed.



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

[Table 2-67, page 223](#), lists the attributes defined for the type.

Table 2-67. Attributes defined for the dmc dsm study report type

Attribute	Datatype	Single/ repeating	Description
study_report_status	string(32)	S	Only valid value is Delete, to indicate that the logical content has been deleted from the STF.

DSM Submission

Purpose

Implementation

Supertype: Folder
Subtypes: None
Internal name: dmc_dsm_submission
Object type tag: 0b

A dsm submission object contains all the documents related to a particular submission. The object type is installed when the EMC Documentum Submissions Manager DocApp is installed.



Caution: This object type is used by EMC Documentum Submissions Manager. You must not modify the object type, nor can you create instances of the type directly.

[Table 2-68, page 224](#), lists the attribute defined for the type.

Table 2-68. Attributes defined for the dmc dsm submissionrt type

Attribute	Datatype	Single/ repeating	Description
previous_name	string(32)	S	Previous submission name before "lock"

Dump Object Record

Purpose Contains information about an object that has been copied from a repository into an external file using the Dump utility.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dmi_dump_object_record
 Object type tag: 30

A dump object record object contains information about an object that has been copied from a repository into an external file using the Dump utility. The information in a dump object record object is primarily useful if the dump process is interrupted and must be restarted.

[Table 2-69, page 225](#), lists the attributes defined for the type.

Table 2-69. Attributes defined for the dump object record type

Attribute	Datatype	Single/ repeating	Description
extra_data _dumped	Boolean	S	Used internally by the dmarchive utility.
dump_object	ID	S	Object ID of the dump record object associated with the dump procedure that generated this dump object record object.
object_id	ID	S	Object ID of the object that was copied from the repository.
version	integer	S	Version stamp of the dumped object.

Dump Record

Purpose Contains information about a specific dump execution.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dm_dump_record
 Object type tag: 2f

A dump record object contains information about a specific dump execution. The server uses this information to start a dump execution. Additionally, some of the attributes contain recovery information if the dump process is interrupted and must be restarted.

Note: To start a dump process, you create a dump record object and save it. Saving a dump record object automatically starts the dump process. You must have Sysadmin or Superuser privileges to create and save a dump record object.

[Table 2-70, page 226](#), lists the attributes defined for the type.

Table 2-70. Attributes defined for the dump record type

Attribute	Datatype	Single/ repeating	Description
dump_ operation	string(255)	S	Dumps the entire repository if set to full_docbase_dump. If NULL, dumps only the types specified by the type attribute.
dump_ parameter	string(255)	R	Defines parameters for the dump operation, such as the cache size or content compression. Valid values are: <ul style="list-style-type: none"> • compress_content=T F T directs the operation to compress the content. The default is F. • cache_size=<i>integer</i> <i>integer</i> defines the cache size in megabytes

Attribute	Datatype	Single/ repeating	Description
file_name	string(255)	S	Name of the external file that the system is writing the dump into.
include_content	Boolean	S	Indicates whether or not to include the actual content files in the dump file. The default is false.
predicate	string(255)	R	Predicate expression that specifies which objects of the specified types are dumped.
predicate2	string(255)	R	Extends the predicate expression defined in the predicate attribute at corresponding index levels. For example, predicate2[0], if defined, is concatenated at the end of predicate[0] when the dump is executed.
r_current_object_count	integer	S	Number of objects dumped. This is updated continuously throughout the process, as objects are copied to the file.
r_current_pos	integer	S	Position in the dump file at which you wrote the last entry, if file is less than 2 GB.
r_current_root_count	integer	S	Indicates how many of the objects qualified by the predicate have been dumped. This is constantly refreshed during the dump process as the types specified in the predicate are dumped.
r_end_time	TIME	S	Time at which the dump ended.

Attribute	Datatype	Single/ repeating	Description
r_is_complete	Boolean	S	<p>Set at the completion of the dump operation. Values are:</p> <p>T (TRUE), meaning the operation completed successfully</p> <p>F (FALSE), meaning the operation terminated due to an error</p> <p>Note: The value is NULL if the dump record object was created prior to upgrading to version 5.3.</p>
r_is_more	Boolean	S	<p>This value is set when r_is_complete is set to T. Valid values are:</p> <p>T (TRUE), meaning the dump operation completed because it hit the objects_per_transfer limit, but there are more objects to dump</p> <p>F (FALSE), meaning that the dump operation completed and there are no more object to dump.</p> <p>The value of this attribute is undefined if r_is_completed is set to F (FALSE).</p>
r_root_count	integer	S	Number of objects qualified by the predicate.
r_start_time	TIME	S	Time at which the dump was started.
type	string(33)	R	Identifies the types to be dumped. (Note that the predicate and type repeating attributes are associated in that the predicate in position x is applied to the type in position x, the predicate in position y is applied to the type in position y, and so forth.)
r_current_offset	string(20)	S	Position in the dump file at which the server wrote the last entry if file is 2 GB or greater.

Email Address Table

Purpose Records all unique, full email addresses for an email message.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal Name: dm_email_address_table
 Object type tag: 00

An email address table object records a unique email address found in the header of an email message. A unique address is an address not currently represented in the repository by another email address table object. Each time a unique address is found in an archived message, an email address table object is created to record the address. Email message table objects are primarily used by personal and compliance archiving applications.

[Table 2-71, page 229](#), lists the attributes defined for the type.

Table 2-71. Attributes defined for the email address table type

Attribute	Datatype	Single/ repeating	Description
addr_id	string(16)	S	Hash value of the email address
email_addr	string(1024)	S	Fully qualified email address of the user
friendly_name	string(128)	S	User's name as it appears prepended to the full email address. This name typically appears in double quotes prior to the actual email address in angle brackets.
mail_user	string(128)	S	User's name as it appears in the actual email address.
primary_domain	string(512)	S	Primary domain of the email address.

Attribute	Datatype	Single/ repeating	Description
sub_domain	string(512)	S	Subdomains in the email address.
user_name	string(32)	S	User's user_name value if there is a repository user who has the email address recorded in email_addr.

Email Message

Purpose Stores the content of an email message.

Implementation

Supertype: Document
 Subtypes: None
 Internal Name: dm_email_message
 Object type tag: 09

An email message object stores an electronic message as content. Any user can create an email message object.

[Table 2-72, page 231](#), lists the attributes defined for the type.

Table 2-72. Attributes defined for the email message type

Attribute	Datatype	Single/ repeating	Description
date_received	Date	S	The date and time at which the message was received.
date_published	Date	S	Date and time on which the message was sent.
media_type	string(32)	S	Identifies what kind of message this is. For example, possible values are email, IM (instant message), note, or calendar.
message_identifier	string(32)	S	The unique message identifier provided by the messaging system.
other_recipients	string(48)	R	Message recipients identified in the CC list.
originating_org	string(128)	S	The sender of the message
recipients	string(48)	R	Message recipients identified in the To list.

Esign Template

Purpose Stores a signature page template as content

Implementation

Supertype: Document
Subtypes: None
Internal name: dm_esign_template
Object type tag: 09

An esign template object is used to store a signature page template in the repository. A default signature page template is provided with Content Server, and you can create custom signature page templates. Each template is represented in the repository by one esign template object.

Table 2-73, page 232, lists the attributes defined for the type.

Table 2-73. Attribute defined for the esign template type

Attribute	Datatype	Single/ repeating	Description
append_to_body	Boolean	S	Whether to append or pre-pend the signature page to the content. T (TRUE) means to append the content. F (FALSE) means to pre-pend the content. The default is F.
begin_tag	string(1)	S	Identifies the beginning delimiter for tags on the signature page template. The default is <.
document_type	string(32)	R	Names of the object types that can be signed using the template.

Attribute	Datatype	Single/ repeating	Description
end_tag	string(1)	R	Identifies the ending delimiter for tags on the signature page template. The default is >.
max_signatures	integer	S	Defines the maximum number of signatures that may added to a particular version of the object. The default is 0. There is no upper limit.

Expr Code

Purpose Stores generated expression source code and pcode.

Implementation

Supertype: SysObject
Subtypes: None
Internal name: dmi_expr_code
Object type tag: 58

An expr code object stores generated expression source code and pcode. Content Server creates and manages expr code objects. Users cannot create them.

[Table 2-74, page 234](#), lists the attribute defined for the type.

Table 2-74. Attribute defined for the expr code type

Attribute	Datatype	Single/ repeating	Description
i_type_name	string(32)	S	Name of the type on whose behalf the source code and pcode are collected.
parent_id	ID	S	Object ID of an aggrdomain object. Used internally by dump and load operations.

Expression

Purpose Stores information needed to execute expressions defined in the data dictionary.

Implementation

Supertype: Persistent
 Object Subtypes: Func Expr, Literal Expr, Builtin Expr
 Internal name: dm_expression
 Object type tag: 52

An expression object stores information needed to execute expressions defined in the data dictionary. Expression objects are created and managed by Content Server and cannot be created by users.

[Table 2-75, page 235](#) lists the attributes defined for the type.

Table 2-75. Attribute defined for the expression type

Attribute	Datatype	Single/ repeating	Description
expression_name	string(32)	S	Currently unused. For the current release, this is always NULL.
expression_text	string(255)	R	The text of the expression. If the expression text is greater than 255 characters, the first 255 characters are put in expression_text[0], the second 255 in expression_text[1] and so forth.

Attribute	Datatype	Single/ repeating	Description
expression_type	integer	S	<p>The data type of the expression.</p> <p>If the expression subtype is literal expr or builtin expr, valid values are:</p> <ul style="list-style-type: none"> 0, Boolean 1, Integer 2, String 3, ID 4, Time/Date 5, Double <p>For the func expr subtype, valid values are:</p> <ul style="list-style-type: none"> 0, Boolean 1, Integer 3, ID
parent_id	ID	S	Object ID of the aggr domain or policy object that references the expression object.

External File Store

Purpose Represents an external file store.

Implementation

Supertype: External Store
 Subtypes: None
 Internal name: dm_extern_file
 Object type tag: 61

An external file store object represents an external file store. The server uses external file store objects to locate content stored in external file stores.

[Table 2-76, page 237](#) lists the attributes defined for dm_extern_file.

Table 2-76. Attributes defined for the external file store type

Attribute Name	Datatype	Single/ repeating	Description
a_config_name	string(64)	R	Name of the server config object to use for a_location[position].
a_location	string(64)	R	Name of a location object to be used as root_name for a_config_name[position].
def_client_root	string(64)	S	Name of the client-specific location object. Default is NULL.
def_server_root	string(64)	S	Name of the server-specific location object. Default is NULL.

External Free Store

Purpose Represents an external storage area accessed through a user-defined content token.

Implementation

Supertype: External Store
Subtypes: none
Internal name: dm_extern_free
Object type tag: 63

The external free store object type allows you to specify a token for content retrieval and storage that does not follow the standard for external file store or external URL store. You define your own token standard and means of retrieving the content associated with the token.

Configuration of plug-in attributes for the external free store type is fully under the discretion of users. Depending on the accessibility of the content, users decide whether to run the plug-in on the server or client.

The external free store object type inherits all its attributes from its supertype. No attributes are defined specifically for it.

External Store

Purpose Represents an external storage area accessible to the server but whose physical contents are stored outside the server.

Implementation

Supertype: Store

Subtypes: External File Store, External URL Store, External Free Store

Internal name: dm_extern_store

Object type tag: 60

An external store object represents an external storage area accessible to the server but whose physical contents are stored outside the server. An external store object eliminates the need to transfer the content from client to server during a save operation. All subtypes of dm_extern_store operate in token mode.

If a valid value is specified for root_name, and the client can access the full path, set a_exec_mode to TRUE and specify an a_plugin_id-a_platform value pair for the client platform. Otherwise, set a_exec_mode to FALSE and create an a_plugin_id-a_platform value pair for the server platform.

[Table 2-77, page 239](#), lists the attributes defined for the external store object type.

Table 2-77. Attributes defined for the external store type

Attribute Name	Datatype	Single/ repeating	Description
a_content_static	Boolean	S	Controls how Getfile methods behave when fetching content if the storage area is an external storage area. For an explanation of the behavior, refer to Configuring for optimal performance on retrieval, page 255 . The default is F.

Attribute Name	Datatype	Single/ repeating	Description
a_exec_mode	Boolean	S	Controls where to execute the plug-in. If TRUE, execute the plug-in on the client. If FALSE, execute the plug-in on the server. The default is FALSE
a_plugin_id	ID	R	ID of the document whose content is the plug-in DLL (Windows) or shared library (UNIX). The storage type for this document must be dm_filestore.
a_platform	integer	R	Identifies the client platform. Valid values are: 1, for Windows 2, for Solaris 3, for AIX 4, for HP-UX 5, for Macintosh 6, for Linux 7, for Itanium

External URL Store

Purpose Represents an external storage area whose content is accessed using a token that follows the URL standard.

Implementation

Supertype: External Store
Subtypes: None
Internal name: dm_extern_url
Object type tag: 62

An external URL store object represents objects whose content is stored externally and accessed using token-mode operation, in which the token follows the URL standard. For the external URL store object type, it is best to configure the plug-in for content retrieval to be executed on the client side. URLs are universal and can be accessed from anywhere. Content Server does not validate the URL.

The external URL store object type inherits all its attributes from its supertype. No attributes are defined specifically for it.

Federation

Purpose Contains the description of a federation.

Implementation

Supertype: SysObject
Subtypes: None
Internal name: dm_federation
Object type tag: 5e

A federation object contains the description of a federation. Federation objects are created and managed by Content Server when you create a federation using Documentum Administrator. Each repository belonging to a federation will have a federation object that describes the federation.

[Table 2-78, page 242](#) lists the attributes defined for the type.

Table 2-78. Attribute defined for the federation type

Attribute	Datatype	Single/ repeating	Description
r_govern_docbase	string(120)	S	Name of the governing repository in the federation. This value serves as the default name of the federation unless the object's object_name attribute is explicitly set. This attribute is set in the federation objects for all members of the federation.

Attribute	Datatype	Single/ repeating	Description
r_govern_is_active	Boolean	S	<p>Indicates whether the governing repository is active. If this is FALSE (the default), federation-related jobs cannot be executed.</p> <p>This value is set only in the federation object in the governing repository.</p>
r_govern_suspended	Boolean	S	<p>Indicates whether transactions or changes to configuration objects can occur in the governing repository. The default is FALSE, meaning that transactions are allowed. (This is used to manage synchronization and integrity between repositories in the federation while jobs are executing.)</p> <p>This value is set only in the federation object in the governing repository.</p>
r_member_docbases	string(120)	R	<p>Names of the repositories that belong to the federation. Each name must be unique within the list.</p> <p>This value is set only in the federation object in the governing repository.</p>

Attribute	Datatype	Single/ repeating	Description
r_member_is_active	Boolean	R	<p>Indicates whether the member repository is active in the management of configuration objects. The value at each index level is associated with the repository at the corresponding index level in r_member_docbases.</p> <p>The default is TRUE, meaning that the repository participates.</p> <p>This value is set only in the federation object in the governing repository.</p>
r_member_refresh	date	R	<p>Indicates the last refresh date of the configuration objects in the member repositories. The date at each index level is associated with the repository at the corresponding index level in r_member_docbases.</p> <p>This value is set only in the federation object in the governing repository.</p>
user_subtypes	string(32)	R	<p>dm_user subtypes to propagate.</p> <p>The federation update job propagates users represented by the users subtypes listed in this attribute (in addition to users represented by dm_user objects). The subtypes must exist on all member repositories.</p>

File Store

Purpose Contains information about a file store storage area.

Implementation

Supertype: Store
 Subtypes: none
 Internal name: dm_filestore
 Object type tag: 28

A file store object contains information about a file store storage area. A storage area of type dm_filestore is one of the most common types of file storage areas in a Content Server installation. It is used to store files that have a wide variety of formats.

[Table 2-79, page 245](#), lists the attributes defined for the type.

Table 2-79. Attributes defined for the file store type

Attribute	Datatype	Single/ repeating	Description
is_public	Boolean	S	Indicates if the area is accessible to the public with no restrictions. Valid values are: 1, for is accessible 0, for is not accessible
root	string(64)	S	Object name of the location object representing this storage area
use_extensions	Boolean	S	Indicates whether server should append a DOS extension to the file when writing it into the storage area.

Folder

Purpose Used, in conjunction with cabinets, to organize the contents of your repository.

Implementation

Supertype: SysObject
 Subtypes: Cabinet, Category, Room, Module, Topic,
 DSM Application, DSM Submission, XML Application
 Internal name: dm_folder
 Object type tag: 0b

The folder object serves, in conjunction with cabinets, to organize the contents of your repository. All SysObjects and SysObject subtypes (except cabinets) must be stored in either directly in a cabinet or in a folder. Folders, in turn, are stored in cabinets or other folders. Ultimately, every SysObject or subtype is stored in a cabinet.

Table 2–80, page 246, lists the attributes defined for the folder type.

Table 2-80. Attributes defined for the folder type

Attribute	Datatype	Single/ repeating	Description
i_ancestor_id	ID	R	Object ID of the folders or cabinets that contain this folder directly or indirectly.
r_folder_path	With Oracle and DB2: string(740) With Sybase: string(600) With SQL Server: see Description	R	Folder paths for all locations to which the folder is linked. The length of this attribute will vary for SQL Server customers: For a new SQL Server repository, the length is 450. For an upgraded SQL Server repository not using Unicode, the length is 765.

Foreign Key

Purpose Describes the set of attributes that define a foreign key.

Implementation

Supertype: Relation Type
 Subtypes: None
 Internal name: dm_foreign_key
 Object type tag: 65

A foreign key object describes the set of attributes that define a foreign key. Content Server creates foreign key objects when users define foreign keys for a type. Users cannot create foreign key objects.

[Table 2-81, page 247](#), lists the attributes defined for the type.

Table 2-81. Attributes defined for the foreign key type

Attribute	Datatype	Single/ repeating	Description
child_attributes	string(32)	R	Names of the child key attributes for the foreign key.
parent_attributes	string(32)	R	Names of the parent key attributes for the foreign key.

Format

Purpose Records information about a file format recognized by Content Server.

Implementation

Supertype: Persistent Object

Subtypes: None

Internal name: dm_format

Object type tag: 27

A format object contains information about a file format recognized by Content Server. A predefined set of file formats is installed by default when a repository is configured.

[Table 2-82, page 248](#) lists the attributes defined for the type.

Table 2-82. Attributes defined for the format type

Attribute	Datatype	Single/ repeating	Description
asset_class	string(32)	S	Identifies the kind of asset (video, audio, and so forth) represented by this format. This is used by applications.
can_index	Boolean	S	Indicates whether this format can be full-text indexed. This attribute is set by an entry in a configuration file.
com_class_id	string(38)	S	The class ID (CLSID) recognized by the Windows registry for a content type.
default_storage	ID	S	Identifies the default storage area for contents having this format.
description	string(64)	S	User-defined description of the format.

Attribute	Datatype	Single/ repeating	Description
dos_extension	string(10)	S	The DOS extension to use when copying a file in the format into the common area, client local area, or storage.
filename_modifier	string(16)	S	The modifier to append to a file name, to create a unique file name.
format_class	string(32)	R	<p>Content Server may set this to either:</p> <ul style="list-style-type: none"> • ftalways, meaning that content files in this format are always indexed • ftpreferred, meaning if there are multiple renditions of the content, this format is the preferred format for indexing <p>May be also be set to a user-defined value to identify the class or classes of formats to which the format belongs. For example, the xml, xsd, and xsl formats belong to the XML and MSOffice classes.</p>
icon_index	integer	S	The index that locates the icon for the format in the icon resource file.
is_hidden	Boolean	S	Used by some Documentum clients.
mac_creator	string(4)	S	Information used internally for managing Macintosh resource files.

Attribute	Datatype	Single/ repeating	Description
mac_type	string(4)	S	Information used internally for managing Macintosh resource files.
mime_type	string(64)	S	The Multimedia Internet Mail Extension (MIME) for the content type.
name	string(64)	S	Name of the format; for example, doc or tiff.
richmedia_enabled	Boolean	S	Indicates whether thumbnails, proxies, and metadata are generated for content in this format. This is set to TRUE by default for the following formats: jpeg, mpeg, pdf, gif, and avi. It is FALSE for all other formats by default.
topic_filter	string(64)	S	Obsolete
topic_format	ID	S	The ID of the format object representing the format to which this format must be transformed for indexing if topic_transform is T (TRUE)
topic_format_name	string(64)	S	Name of the format object representing the format to which this format must be transformed for indexing if topic_transform is T (TRUE) For example, if this format object represents the PDF format and its topic_transform attribute is T, then topic_format_name for the PDF format is (typically) PDFText.

Attribute	Datatype	Single/ repeating	Description
topic_transform	Boolean	S	Use the name as defined in the format's format object. Indicates if it is necessary to transform the format before indexing
win31_app	string(12)	S	Used by Documentum clients. It identifies the application to launch when users select a document in the format represented by the format object.

FT Index Agent Config

Purpose Stores information about an index agent.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_ftindex_agent_config
 Object type tag: 08

An ft index agent config object stores information about one index agent.

[Table 2–83, page 252](#), lists the attribute defined for the type plus two that are inherited and used in a specific manner by this type.

Table 2-83. Attributes defined for the FT index agent config type

Attribute	Datatype	Single/ repeating	Description
active_connectors	string(32)	S	Whether the index agent is running in normal or migration mode. Valid values are; normal , indicating normal mode reindex , indicating migration mode If the attribute is not set or is blank, the default is normal.
checkpoint_interval	integer	S	Reserved for future use
connectors_batch_size	integer	S	Number of queue items picked up by the queue reader in each polling interval. The default is 1000.

Attribute	Datatype	Single/ repeating	Description
docbase_ connector_wait_ time	integer	S	Frequency, in seconds, at which the index agent polls for events The default is 60.
exporter_queue_ threshold	integer	S	Used to control additions to the index agent's internal, in-memory queue. If the number of objects in the queue is equal to or exceeds the value in this attribute, then no more objects may be added to the queue. The default value is 500.
exporter_thread_ count	integer	S	Number of concurrent exporter threads being run by the index agent The default is 3.
force_inactive	integer	S	Indicates whether the agent is operating normally or shutdown and waiting for a manual start. Valid values are: 0, meaning the agent is operating normally 1, meaning the server will not start the agent until this attribute is set to 0 manually. The default is 0.
index_name	string(64)	S	Name of the fulltext index object associated with this index agent.

Attribute	Datatype	Single/ repeating	Description
indexer_queue_ threshold	integer	S	Used to control additions to the index agent's internal, in-memory queue. If the number of objects in the queue is equal to or exceeds the value in this attribute, then no more objects may be added to the queue. The default value is 500.
object_name	string(255)	S	This is an inherited attribute. For ft index agent config objects, the value must be unique among all ft index agent config objects in the repository. The default value for the index agent installed with Content Server is <i>hostname_indexagentname</i> .
queue_user	string(64)	S	User name used by the index agent to look for queued items The default is <i>dm_fulltext_index_user</i> .
r_fail_time	Date	S	Reserved for future use
r_is_active	integer	S	Whether the index agent is active or inactive. Valid values are 0, meaning the agent is inactive, and 1, meaning the agent is active.
r_last_done_time	Date	S	Reserved for future use
r_last_work_time	Date	S	Reserved for future use

Attribute	Datatype	Single/ repeating	Description
r_modify_date	Date	S	This is an inherited attribute. For ft index agent config objects, this is set to the last time the associated index agent updated the ft index agent config object.
r_start_time	Date	S	Reserved for future use
r_stop_time	Date	S	Reserved for future use
runaway_item_timeout	integer	S	How long a queue item remains in the internal queue before it is abandoned. The value is interpreted in seconds. The default value is 600.
save_queue_items	integer	S	Whether successfully processed queue items are saved or destroyed. 0 means to mark successfully processed queue items "done" and save them. 1 means to destroy successfully processed queue items. The default is 1.
shut_down_requested	integer	S	Reserved for future use

FT Engine Config

Purpose Records configuration information for an index server.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_ftengine_config
 Object type tag: 08

An ft engine config object stores the parameters necessary to use a particular index server for indexing or querying. There is one ft engine config associated with each fulltext index object in a repository. Permissions to modify objects of this type are controlled by the ACL named dm_fulltext_admin_acl.



Caution: Changing the values in the param_name or param_value attributes manually may result in undefined behavior during indexing or querying. It is recommended that you do not change these attributes unless told to do so by technical support.

Table 2-84, page 256, lists the attributes defined for the type.

Table 2-84. Attributes defined for the FT engine config type

Attribute	Datatype	Single/ repeating	Description
param_name	string(64)	R	Names of the parameters passed to the engine.
param_value	string(255)	R	Values of the parameters passed to the engine. The value at each index position is associated with the name at the same index position in param_name.

Fulltext Index

Purpose Represents a full-text index for the repository.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dm_fulltext_index
 Object type tag: 3b

A fulltext index object represents the index associated with the repository. The object is created internally when a 'normal mode' index agent is configured for the repository. Only the `ft_engine_id` attribute can be modified by a user. Superuser or Sysadmin user privileges are required to change this attribute.

[Table 2-85, page 257](#), lists the attributes defined for the type.

Table 2-85. Attributes defined for the fulltext index type

Attribute	Datatype	Single/ repeating	Description
codepage	string(64)	S	Identifies the code page in use for strings sent to and received from the Index Agent. Typically, this is UTF-8.
ft_engine_id	ID	S	Object ID of the <code>dm_ftengine_config</code> object used for this index.
index_name	string(64)	S	Name of the fulltext index object. This must consist of ASCII characters.

Attribute	Datatype	Single/ repeating	Description
install_loc	string(32)	S	Name of the location object that identifies the location of the fulltext installation for this index. The default name is the value of the fulltext_location attribute in the server config object.
is_standby	Boolean	S	Reserved for internal use.
r_content_count	integer	S	Reserved for future use
r_index_type	integer	S	Indicates the type of index. The default value is 4.
r_last_clean	integer	S	Reserved for future use
r_last_update	DATE	S	Time and date of the last update for this index.

Func Expr

Purpose Stores the data dictionary information needed to execute expressions written in a third-party expression language.

Implementation

Supertype: Expression
 Subtypes: Cond Expr
 Internal name: dm_func_expr
 Object type tag: 55

A func expr object stores the data dictionary information needed to execute expressions written in a third-party expression language. Content Server creates and manages func expr objects. Users cannot create them directly.

Table 2-86, page 259, lists the attributes defined for the type.

Table 2-86. Attributes defined for the func expr type

Attribute	Datatype	Single/ repeating	Description
attr_object_index	integer	R	<p>For each attribute in attribute_name, provides an index into the type_name list that identifies the source type of the attribute.</p> <p>The value in attr_object_index[0] is the index value to use for the attribute specified in attribute_name[0]. For example, if attr_object_index[0] is 3, then the attribute specified in attribute_name[0] is taken from the type specified in type_name[3].</p>

Attribute	Datatype	Single/ repeating	Description
attribute_name	string(32)	R	List of attribute names that are passed to the expression. The names are passed in the order in which they are listed.
code_page	integer	R	Content page number in the object specified in routine_id that contains the generated code. Currently, this value is always 1.
expression_kind	integer	S	Indicates whether the expression is a complete routine or an expression. Valid values are: 1, Expression 2, User routine
expression_lang	integer	S	Identifies the language of the expression or routine. The only valid value is 1, meaning Docbasic.
object_alias	string(32)	R	Identifies an alias for each type named in type_name. The aliases are used to map attributes in the expression_text to their associated types. The alias at each index level is associated with the type specified at the corresponding index level in type_name.

Attribute	Datatype	Single/ repeating	Description
repeat_attr_index	integer	R	<p>For each repeating attribute in <code>attribute_name</code>, this provides the index into the attribute's list of values, to identify the desired value. Valid values are:</p> <p>0 or greater, Index into the repeating attribute -2, Any value in the attribute -3, All values in the attribute -4, First value in the attribute -5, Last value in the attribute</p> <p>For single-valued attributes specified in <code>attribute_name</code>, the corresponding value in <code>repeat_attr_index</code> is always -1.</p>
routine_id	ID	S	Object ID of the SysObject containing the source and pcode content pages.
routine_name	string(255)	S	Name of the entry point in the routine.
source_page	integer	S	Content page number in the object specified in <code>routine_id</code> containing the source code.
type_name	string(32)	R	<p>Currently, this value is always 0.</p> <p>Names of the object types that are the sources of the attributes specified in <code>attribute_name</code>.</p>

Group

Purpose Contains information about a group in the repository.

Implementation

Supertype: Persistent Object

Subtypes: None

Internal name: dm_group

Object type tag: 12

A group object contains information about a group in the repository.

[Table 2–87, page 262](#), lists the attributes defined for the group type.

Table 2-87. Attributes defined for the group type

Attribute	Datatype	Single/ repeating	Description
alias_set_id	ID	S	The object ID of an alias set associated with the group.
description	string (255)	S	User-defined description of the group.
globally_managed	Boolean	S	Indicates whether the group is globally or locally managed. The default is FALSE, meaning that it is locally managed.
			Requires at least Sysadmin privileges to change.
group_address	string(80)	S	Electronic mail address for the group.
group_admin	string(32)	S	Name of a user or group who can modify the group.

Attribute	Datatype	Single/ repeating	Description
group_class	string(32)	S	Indicates whether the group is a standard group, a role group, or a domain. Valid values are: group role domain
group_directory_id	ID	S	Object ID of the LDAP config object representing the LDAP directory used to synchronize this group.
group_display_name	string(255)	S	Name of the group as it appears in a room. The name must be unique within the room. The default is the group_name.
group_global_unique_id	string(255)	S	Reserved for future use
group_name	string(32)	S	Name of the group. If group_class is set to role, then group_name is the name of the role. If group_class is set to domain, then group_name is set the name of the domain. The name must consist of characters compatible with the server os code page of the Content Server and must be unique among the user and group names in the repository.

Attribute	Datatype	Single/ repeating	Description
group_native_ room_id	ID	S	Object ID of the room that contains this private group.
group_source	string(16)	S	Identifies the source of the group. The only valid value is LDAP, meaning the group was created by importing an LDAP group.
groups_names	string(32)	R	Names of any groups that are users in this group.
i_all_users_names	string(32)	R	List of all users in the group, including those users that belong to groups contained within the group. This attribute value is computed when the attribute is first queried in a session and cached on the server and client.
i_supergroups _names	string(32)	R	Name of the group and all groups that contain this group.
is_dynamic	Boolean	S	T (TRUE) means the group is a dynamic group. F (FALSE) means the group is not a dynamic group. The default is F.
is_dynamic_default	Boolean	S	Controls whether users in the group's list of potential users are considered members of the group by default when they connect to the repository. T (TRUE) means that users are treated as members of the group. F (FALSE) means that users are not treated as group members when they connect. The default is F (FALSE).

Attribute	Datatype	Single/ repeating	Description
is_private	Boolean	S	Indicates whether the group is private (T) or public (F).
owner_name	string(32)	S	Name of the user or group who owns the group.
r_has_events	Boolean	S	Whether someone has registered the group for auditing.
r_object_id	ID	S	Object ID of the group.
users_names	string(32)	R	Names of the directly contained users in the group.

Index

Purpose Stores information about an RDBMS index created in the repository.

Implementation

Supertype: Persistent Object

Subtypes: None

Internal name: dmi_index

Object type tag: 1f

An index object stores information about an RDBMS index created in the repository.

[Table 2–88, page 266](#) lists the attributes defined for the type.

Table 2-88. Attributes defined for the index type

Attribute	Datatype	Single/ repeating	Description
attribute	integer	R	Position of the attributes being indexed in the type's definition.
attr_count	integer	S	Number of attributes (columns) in the index
data_space	string(64)	S	Name of the Oracle tablespace or MS SQL Server or Sybase database in which the index resides.
index_type	ID	S	Object ID of the dm_type object that represents the type being indexed.
is_unique	Boolean	S	Indicates if the rows in the index are unique
name	string(64)	S	Name of the index. This is generated internally.
rebuilding	Boolean	S	This is TRUE if the index is currently being moved from one location to another.

Attribute	Datatype	Single/ repeating	Description
repeating	Boolean	S	Indicates whether the index is built on the type's single-valued attribute table or the repeating attribute table. Values are F for the single-valued attribute table and T for the repeating attribute table.
use_id_col	Boolean	S	Used by server for server-generated indexes.
use_pos_col	Boolean	S	Used by server for server-generated indexes.
use_tag	integer	R	Used by server for server-generated indexes.

Jar

Purpose Represents a jar file stored in the repository.

Implementation

Supertype: SysObject
Subtypes: None
Internal name: dmc_jar
Object type tag: 08

A jar object contains information about a jar file stored in the repository as the object's content. The object type is installed by a script when Content Server is installed. Instances of the type are created through Documentum Application Builder.

[Table 2-89, page 268](#), lists the attributes defined for the type.

Table 2-89. Attributes defined for the jar type

Attribute	Datatype	Single/ repeating	Description
jar_type	integer	S	Identifies what type of jar file this object represents. Valid values are: 1, meaning interface 2, meaning implementation 3, meaning both interface and implementation
minimum_vm_version	string(32)	S	Specifies the VM version level required by this file.

Java Library

Purpose Represents a third-party Java library stored in the repository.

Implementation

Supertype: Folder
 Subtypes: None
 Internal name: dmc_java_library
 Object type tag: 0b

Java library objects represent java libraries stored in the repository. Each object represents one library. The object type is installed by a script when Content Server is installed. Instances of the type are created through Documentum Application Builder.

[Table 2-90, page 269](#), lists the attribute defined for the type.

Table 2-90. Attributes defined for the java library type

Attribute	Datatype	Single/ repeating	Description
sandbox_library	Boolean	S	Indicates how the library is to be loaded T (TRUE) means the library is loaded by the Business Object class loader. F (FALSE) means the library loaded by the Shared class loader.

Job

Purpose Used to run a program at regularly scheduled intervals.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_job
 Object type tag: 08

A job object represents a program that you want to run at regularly scheduled intervals. A job object stores information about the program, such as the name of the method object for the program, how often or when to run it, the number of times the program has been run, the date of its last execution, and so forth. Programs represented by job objects are launched by a special utility called the dm_agent_exec utility, which is itself started by Content Server.

Table 2-91, page 270, lists the attributes defined for the type.

Table 2-91. Attributes defined for the job type

Attribute	Datatype	Single/ repeating	Description
a_continuation _interval	integer	S	Indicates how often to re-invoke the job, in minutes. If set to a number greater than zero, the job is automatically re-started after the specified interval. The default is 0.
a_current_status	string(255)	S	Contains status information provided by the application or program run by the job.
a_is_continued	Boolean	S	Indicates whether the current execution of the job is a result of an automatic re-invocation of the job. The default is FALSE.

Attribute	Datatype	Single/ repeating	Description
a_iterations	integer	S	Number of times the job has been invoked and completed.
a_last_completion	date	S	Contains the date and time when the last invocation of the job was completed.
a_last_document_id	ID	S	Object ID of the status document created for the last invocation of the job.
a_last_invocation	date	S	Contains the date and time when the last invocation of the job was started.
a_last_process_id	ID	S	The process id of the last-launched dm_method.
a_last_return_code	integer	S	Return status of the last dm_method executed.
a_next_continuation	date	S	Date of next automatic invocation of the job. The default is NULLDATE.
a_next_invocation	date	S	Computed time of next invocation.
a_special_app	string(32)	S	This attribute is inherited from dm_sysobject. It is set by the server when the job is checked out by the dm_agent_exec utility for execution.
expiration_date	date	S	Date and time on which the job expires. The job will not be executed after the specified date and time. The default is NULLDATE.
inactivate_after_failure	Boolean	S	Indicates if the job is inactive due to failure. The default is FALSE.

Attribute	Datatype	Single/ repeating	Description
is_inactive	Boolean	S	Indicates whether the job is unavailable for execution. The default is FALSE, which indicates that the job can be executed.
max_iterations	integer	S	Maximum number of times to execute the job. The default is 1. A value of zero (0) indicates that there is no maximum.
method_arguments	string(255)	R	Arguments to pass to the method. These arguments are passed only if the pass_standard_arguments attribute is set to F.
method_data	string(255)	R	Used by procedure associated with job. This attribute is available for the program to write/read as needed during its execution.
method_name	string(255)	S	Name of the method to invoke when the job is executed. This is a required attribute.
method_trace_level	integer	S	Turns on tracing for the method.
object_name	string(255)	S	Name of the job object. This is inherited from dm_sysobject.

Attribute	Datatype	Single/ repeating	Description
pass_standard_arguments	Boolean	S	<p>T means that the standard arguments are passed to the method and the arguments in method_arguments are not passed.</p> <p>(The standard arguments are: <i>-docbase_name repositoryname.serverconfig</i> <i>-user_name username</i> <i>-job_id jobid</i> <i>-method_trace_level methodtracelevel</i>)</p> <p>F means that the standard arguments are not passed to the method, but the arguments in the method_arguments value are passed instead.</p> <p>The default is F.</p>
run_interval	integer	S	<p>Used in conjunction with run_mode to determine how often to invoke the job. For example, if you enter 3 for this attribute and 4 for run_mode, the job is invoked every three weeks. The default is zero (0).</p>
run_mode	integer	S	<p>Used in conjunction with run_interval to determine how often to invoke the job. Run_mode specifies the unit of measure for the value in run_interval. Valid values are:</p> <ul style="list-style-type: none"> 1, for every x minutes 2, for every x hours 3, for every x days 4, for every x weeks 5, for every x months 7, for every x day of week

Attribute	Datatype	Single/ repeating	Description
			8, for every x day of month 9, for every x day of year x is the value in run_interval
run_now	Boolean	S	The default value is zero (NULL). If set to T, the job is executed the next time the agent exec program polls the repository.
start_date	date	S	Date and time of the first required invocation. This is a required attribute.
target_server	string(255)	S	Specifies a particular server to run the job. Specify the server name using the format: <i>repository[.server_config_name] [@machine]</i> The default value is NULLSTRING.

Job Request

Purpose Used internally by Documentum Administrator.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_job_request
 Object type tag: 08

A job request object records information needed by Documentum Administrator to execute certain jobs. Users cannot create these objects manually.



Caution: Users, or applications, must not modify these objects. However, they may be deleted if request_completed is T, indicating that the associated job has been invoked.

Table 2-92, page 275, lists the attributes defined for the object type.

Table 2-92. Attributes defined for the job request type

Attribute	Datatype	Single/rRepeating	Description
arguments_keys	string(255)	R	Internal keys used in conjunction with arguments_values. The key at any particular index position is used in conjunction with the value at the corresponding index position in arguments_values.
arguments_values	string(255)	R	Argument values passed to the job.
job_name	string(255)	S	Name of the requested job
method_name	string(255)	S	Name of the method invoked by the job

Attribute	Datatype	Single/rRepeating	Description
priority	integer	S	A priority value internally assigned by Documentum Administrator
request_completed	Boolean	S	Whether the job has been executed. T means the job has been invoked. F means the job is awaiting execution.

Job Sequence

Purpose Identifies a job that belongs to a set of jobs executed in a particular sequence.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_job_sequence
 Object type tag: 08

A job sequence object stores information about a job that belongs to a set of jobs executed in a particular sequence. Sequenced jobs are executed using the `dm_run_dependent_jobs` method, which can be invoked by another, controlling job or on the command line. Job sequence objects are created using Documentum Administrator.

[Table 2-93, page 277](#), lists the attributes defined for the object type and the one inherited attribute that has a specific use for job sequences.

Table 2-93. Attributes defined for the job sequence type

Attribute	Datatype	Single/ repeating	Description
<code>job_display_name</code>	<code>string(255)</code>	S	Reserved for use by Documentum Administrator
<code>job_docbase_name</code>	<code>string(255)</code>	S	Identifies the repository in which the job resides. This string is used to match an entry in the repository connection file if that file is used. Valid values for this string are: <i>repository_name</i> <i>repository_name.content_server_name</i> <i>repository_name@host_name</i>

Attribute	Datatype	Single/ repeating	Description
			<i>repository.content_server_name@host_name</i>
job_id	ID	S	Object ID of the job
job_login_domain	string(255)	S	Login domain of the user identified in job_login_user_name
job_login_user_name	string(32)	S	Login name of the user as whom the job identified in job_id will run
object_name	string(255)	S	This inherited attribute must be set to the name of the job sequence that includes the job identified in this job sequence object.
predecessor_id	ID	R	Identifies the job or jobs that must complete successfully before the job specified in job_id is run. The jobs are identified by their object IDs. This attribute may be empty.

Key

Purpose Contains information describing a key for an object type.

Implementation

Supertype: Persistent Object

Subtypes: None

Internal name: dm_key

Object type tag: 59

A key object contains information describing a key for an object type. Content Server creates key objects when users define keys for a type. Users cannot create key objects.

[Table 2-94, page 279](#) lists the attributes defined for the type.

Table 2-94. Attributes defined for the key type

Attribute	Datatype	Single/ repeating	Description
is_unique	Boolean	S	Indicates whether the key is a unique key.
key attributes	string(32)	R	Names of the attributes in the key definition.
type_name	string(32)	S	Name of the object type containing the key attributes.
parent_id	ID	S	Object ID of an aggrdomain object. Used internally by dump and load operations.

LDAP Config

Purpose Stores the set-up values that configure a repository's use of an LDAP-compliant directory server.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_ldap_config
 Object type tag: 08

An ldap config object stores the set-up values that configure a repository's use of an LDAP-compliant directory server. The set-up values are defined through Documentum Administrator when the repository is set up to use the LDAP directory server. Setting the values in an ldap config object requires Superuser user privileges.

[Table 2-95, page 280](#) lists the attributes defined for the ldap config type.

Table 2-95. Attributes defined for the LDAP config type

Attribute	Datatype	Single/ repeating	Description
a_last_run	date	S	Used internally
a_last_no	string(20)	S	Used internally
bind_dn	string(255)	S	A distinguished name for binding
bind_pwd	string(32)	S	<p>This is a required attribute.</p> <p>Password for the binding name</p> <p>This is a required attribute.</p>

Attribute	Datatype	Single/ repeating	Description
bind_type	string(16)	S	<p>Defines the binding mode used to authenticate LDAP users. Valid values are:</p> <p><i>bind_search_dn</i>, which uses the user's operating system name to find the Distinguished name for authentication.</p> <p><i>bind_by_dn</i>, which uses the Distinguished name stored in the user's <code>user_ldap_dn</code> attribute to perform authentication.</p>
certdb_location	string(255)	S	Name of the location object that points to the location of the certificate database directory.
deactivate_user_option	Boolean	S	TRUE causes the synchronization job to set users to inactive in the repository if they are inactivated in the Directory Server; FALSE disables the repository update. The default is FALSE.
first_time_sync	Boolean	S	<p>Whether the LDAP directory must perform a first-time synchronization of the users and groups in the repository that match users and groups in the directory.</p> <p>T means to perform such a synchronization. F means not to perform the synchronization. The default is F.</p>

Attribute	Datatype	Single/ repeating	Description
group_obj_class	string(64)	S	Identifies the object class of groups, as defined in the LDAP directory server. This is optional. The default is groupofuniquemembers.
grp_search_base	string(256)	S	Defines the starting point in the directory server's schema for group searches. (Refer to the documentation provided by the directory server vendor for information about search bases.)
grp_search_filter	string(256)	S	Defines a filter for group searches. (Refer to the documentation provided by the directory server vendor for information about the format of a filter specification.)
import_mode	string(7)	S	Controls whether LDAP synchronization is performed for users, groups, or both. Valid values are: user, group, and all. The default is all.
ldap_host	string(128)	S	The host name of the machine on which the LDAP directory server is running. This is required.
map_attr	string(32)	R	Identifies object attributes that are set when the user or group is imported into the repository. The attribute at a particular index position is set to the value identified at the same index position in map_val.

Attribute	Datatype	Single/ repeating	Description
map_attr_type	string(32)	R	Object type for which the attribute identified at the corresponding index position in map_attr is defined.
map_const_attr	string(32)	R	currently unused
map_const_val	string(32)	R	currently unused
map_val	string(64)	R	The name of a Directory server attribute or a hard-coded value to which the attribute identified in the corresponding index position in map_attr is mapped.
map_val_type	string(1)	R	Indicates whether the value in the corresponding index position in map_val is a Directory Server attribute name or an actual value. If map_val value is a Directory Server attribute, then map_val_type is set to A. If map_value is a value, then map_val_type is set to V.
per_search_base	string(256)	S	Defines the starting point in the directory server's schema for user searches. (Refer to the documentation provided by the directory server vendor for information about search bases.)

Attribute	Datatype	Single/ repeating	Description
per_search_filter	string(256)	S	Defines a filter for user searches. (Refer to the documentation provided by the directory server vendor for information about the format of a filter specification.)
person_obj_class	string(64)	S	Identifies the object class of users as defined in the LDAP directory server. This is optional. The default is person.
port_number	integer	S	The port number on the LDAP directory server's host machine which Content Server will use to communicate with the directory server.
rename_group_option	Boolean	S	TRUE updates group names in the repository during synchronization if they have changed in the Directory Server; FALSE disables this option. The default is FALSE.
rename_user_option	Boolean	S	TRUE updates user names in the repository during synchronization if they have changed in the Directory Server; FALSE disables this option. The default is FALSE.

Attribute	Datatype	Single/ repeating	Description
ssl_mode	integer	S	Whether LDAP user authentication takes place over a secure socket layer (SSL). Values are 0, meaning SSL is not used and 1, meaning that SSL with server authentication is used. The default is 0.
ssl_port	integer	S	Port number of the LDAP SSL port. The default is 636.
use_ext_auth_prog	Boolean	S	TRUE directs Content Server to use an external password checking program to authenticate LDAP users. FALSE directs the server to authenticate LDAP users itself. The default is FALSE.
user_subtype	string(64)	S	Identifies the user's object type if the user is a subtype of dm_user.

Linked Store

Purpose A linked store object represents a storage area that links a common directory accessible to all clients and an actual file directory.

Implementation

Supertype: Store
Subtypes: None
Internal name: dm_linkedstore
Object type tag: 2a

A linked store object contains information about the link between a common directory accessible to all clients and an actual file directory.

[Table 2-96, page 286](#), lists the attributes defined for the type.

Table 2-96. Attributes defined for the linked store type

Attribute	Datatype	Single/ repeating	Description
link_location	string(64)	S	Name of the location object that represents the directory that contains the logical link.
symbolic_links	Boolean	S	Indicates if the links defined in the link directory are symbolic or hard links. TRUE indicates symbolic links; FALSE indicates hard links.

Link Record

Purpose Contains information about established links to content storage areas.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dmi_linkrecord
 Object type tag: 2b

A link record object contains information about established links to content storage areas.

[Table 2-97, page 287](#), lists the attributes defined for the type.

Table 2-97. Attributes defined for the link record type

Attribute	Datatype	Single/ repeating	Description
parent_id	ID	S	Object ID of the object with which the content is associated.
component_id	ID	S	Object ID of the content object associated with the content.
format_id	ID	S	Object ID of the format object representing the format of the content for which the link was established.
data_ticket	integer	S	Used internally to fetch and save the content.
other_ticket	integer	S	Used internally to fetch and save the content.

Attribute	Datatype	Single/ repeating	Description
session_count	integer	S	Number of sessions that are accessing the content storage area identified by this link record object.
session_id	ID	R	Session IDs of the sessions accessing the content storage area.

Literal Expr

Purpose Describes a literal value that is an integer, string, Boolean, ID, or date datatype.

Implementation

Supertype: Expression

Subtypes: None

Internal name: dm_literal_expr

Object type tag: 53

A literal expr object describes a literal value that is an integer, string, Boolean, ID, or date datatype. Literal expr objects are created and managed by Content Server and cannot be created by users. The literal expr type has no defined attributes. All of its attributes are inherited from its supertype, dm_expression.

Note: A date literal value specified in a literal expr object cannot require a pattern definition.

Load Object Record

Purpose Records information about objects that have been loaded from a dump file into a repository.

Implementation

Supertype: Persistent Object
Subtypes: None
Internal name: dmi_load_object_record
Object type tag: 32

A load object record object contains information about one object that has been loaded from a dump file into a repository.

[Table 2-98, page 290](#), lists the attributes defined for the type.

Table 2-98. Attributes defined for the load object record type

Attribute	Datatype	Single/ repeating	Description
extra_offset	string(20)	S	Position in the dump file at which the server wrote the last entry, if file is 2 GB or greater.
extra_pos	integer	S	Specifies the position in the dump file where you can find additional information about the object. This is expressed as the number of characters from the beginning of the file.
file_offset	string(20)	S	Position in the dump file at which the server wrote the last entry, if the file is less than 2 GB.

Attribute	Datatype	Single/ repeating	Description
is_synonym	Boolean	S	Indicates if the new_id already exists in the new repository. This is used for objects such as content storage areas and partitions, where you do not want to copy the old object into the new repository, but merely use a matching object in the new repository.
load_object	ID	S	Object ID of the load record object that contains the information about the dump file containing the loaded object.
new_id	ID	S	Object ID of the loaded object in the new repository.
old_id	ID	S	Object ID of the loaded object in the old repository.

Load Record

Purpose Stores information about dump files that are being loaded into a new repository.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dm_load_record
 Object type tag: 31

A load record object stores information about a dump file that is being loaded into a new repository. You can use the information in a load record object, in conjunction with the information in the associated load object record objects, to restart the loading process if it is interrupted.

Table 2-99, page 292, lists the attributes defined for the type.

Table 2-99. Attributes defined for the load record type

Attribute	Datatype	Single/ repeating	Description
file_name	string(255)	S	Name of the dump file that you are loading.
load_operation	string(255)	S	Used internally.
load_parameter	string(255)	R	Defines parameters for the load operation. Valid parameters are: preserve_replica=T F generate_event=T F generate_event is TRUE by default. Setting it to F (FALSE) turns off notification to the fulltext index user for the save event when an object is loaded into the target repository.
predicate	string(255)	R	Used internally.

Attribute	Datatype	Single/ repeating	Description
relocate	Boolean	S	If set to TRUE, this directs the server to assign new object IDs to all the objects in the dump file during the load process. If set to FALSE, this directs the server to use the objects' previous IDs. The default is TRUE.
r_end_time	date	S	Date and time at which the load process ended.
r_first_phase	Boolean	S	Indicates whether the load process has completed the first phase.
r_position	integer	S	Indicates how far into the dump file the load process has gone. This value is updated continually as the load process proceeds.
r_start_time	TIME	S	Time at which the load process was started.
type	string(33)	R	Used internally.
file_offset	string(20)	S	Position in the dump file at which the server wrote the last entry, if file is less than 2 GB.
extra_offset	string(20)	S	Position in the dump file at which the server wrote the last entry, if file is 2 GB or greater.

Location

Purpose Records a file system location for a specific file or directory.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_location
 Object type tag: 3a

A location object contains a file system location for a specific file or directory. The server uses the information in location objects to find the files and directories that it needs for successful operation.

Table 2-100, page 294, lists the attributes defined for the type.

Table 2-100. Attributes defined for the location type

Attribute	Datatype	Single/ repeating	Description
file_system_path	string(255)	S	<p>Specifies the actual location of the directory or file represented by the location object. The specification's syntax must be appropriate for the machine on which the server resides.</p> <p>For example, if the server is on a Windows host, the specification must be expressed as a Windows path specification. If on a UNIX host, the specification must be a UNIX path.</p> <p>The directory path and name must consist of ASCII characters.</p>
mount_point_name	string(32)	S	<p>Identifies the mount point underneath which this location resides. Use the</p>

Attribute	Datatype	Single/ repeating	Description
			name of the mount point object that describes the mount point.
no_validation	Boolean	S	Identifies whether applications create a location object. If set to TRUE, the server does not validate the directory path specified by file_system_path and does not create the directory. The default is FALSE.
object_name	string(255)	S	Name of the location object. The name must be unique among the location objects in a repository. This attribute is inherited from its SysObject supertype, but is included here because its setting has specific requirements for the location type.
path_type	string(16)	S	Indicates whether the location represents a directory or a file. Legal values are directory and file.
security_type	string(32)	S	Defines the security level applied to the directory or file. Legal values are: public_open, public, and private. The default setting is private.

Locator

Purpose Stores a Documentum Resource Locator (DRL), which points to an object in a repository.

Implementation

Supertype: SysObject
Subtypes: None
Internal name: dm_locator
Object type tag: 08

A locator object stores a Documentum Resource Locator (DRL), which points to an object in a repository. The DRL is stored in the one attribute defined for the type.

[Table 2-101, page 296](#), lists that attribute.

Table 2-101. Attributes defined for the locator type

Attribute	Datatype	Single/ repeating	Description
resource_locator	string(255)	S	Contains a text string that resolves to an object in a repository.

Mail Message

Purpose Stores an email message.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_mail_message
 Object type tag: 08

A mail message object is used to store an email message. The header information in the message is stored in the object's attributes and the actual content of the message is stored as content. Mail message objects are primarily used by personal and compliance archiving applications.

Table 2-102, page 297, lists the attributes defined for the type.

Table 2-102. Attributes defined for the mail message type

Attribute	Datatype	Single/ repeating	Description
application_id	integer	S	Identifies the application that sent the message data. Valid values are: 0, meaning the default compliance application 1, meaning the default personal archiving application 3, meaning DCO

Attribute	Datatype	Single/ repeating	Description
attachment_ extract_seq	integer	R	Identifies the attachment's order in the sequence of extracted attachments. The value at a particular index position applies to the attachment identified at the corresponding index position in attachment_name. This is set to 0 for attachments that are not extracted from the message.
attachment_name	string(1000)	R	Names of the attachments.
attachment_type	string(32)	R	Identifies the format of the attachments. For example, text or word. The format at a particular index position applies to the attachment identified at the corresponding index position in attachment_name.
attachments_stored	integer	S	Total number of message attachments stored as renditions or components
bcc_addr	string(512)	S	First 512 bytes of the message's bcc list. Note: The remainder of the list, if any, is stored in the associated route table object.
cc_addr	string(512)	S	First 512 bytes of the message's cc list. Note: The remainder of the list, if any, is stored in the associated route table object.

Attribute	Datatype	Single/ repeating	Description
components_flag	string(1)	S	A bitmask that manages the email's content when the content is stored as virtual document components
connector_type	string(1)	S	Identifies the module used to retrieve the email. Valid values are: 0, meaning a module for MAPI emails 1, meaning a module for Lotus Notes emails 2, meaning a module for Directory emails
datamodel_version	string(6)	S	Used internally to manage data model changes
from_addr	string(512)	S	First 512 bytes of the message's from list. Note: The remainder of the list, if any, is stored in the associated route table object.
ft_flag	integer	S	Controls how the message and its virtual document components, if any, are indexed. Valid values are: 1, Content of the root document and all components are indexed as one document. Searches on any will return the root document object ID. 2, Content of the root document and the components are indexed separately. Searches may return the root document and components.

Attribute	Datatype	Single/ repeating	Description
is_message_edited	Boolean	S	Used internally
is_root_message	Boolean	S	Whether the mail message is a root message. Root messages are those that are the root of a virtual document. T means the message is a root message. F means that it is not a root message.
message_class	string(1)	S	Identifies the class of message represented by the mail message object. Valid values are: 0, meaning email message 1, meaning contact 2, meaning appointment
message_copy_count	integer	S	Set to 1 or 2 if the message represents an original instance or duplicate instance of an embedded message. The value 1 means that this mail message object represents the first instance of an embedded message. The value 2 means that the object represents a second, duplicate copy of an embedded mail message.

Attribute	Datatype	Single/ repeating	Description
message_doctype	string(1)	S	Identifies the kind of email message. Valid values are: 0, meaning MAPI mail message 1, meaning SMTP message 2, Lotus Notes email message
message_id	string(64)	S	Message identifier extracted from the email message
message_importance	string(1)	S	The level of importance of the email message. Valid values are: 0, meaning normal 1, meaning low 3, meaning medium 5, meaning high
message_lang	string(5)	S	The language of the message.
message_link_count	integer	S	The message's link number in a SysObject chain.
message_sensitivity	string(1)	S	Specifies the sensitivity of the message. Valid values are 1 to 255.
message_size	integer	S	Size of the complete email message, including headers and routing information, in bytes

Attribute	Datatype	Single/ repeating	Description
message_source	string(1)	S	Whether the message is extracted, journaled, or classified by the user. Valid values are: 0, meaning journaled 1, meaning extracted 2, meaning directory
original_message_ doc_id	ID	S	Object ID of the first mail message object with this message ID. This is blank if the original mail message is not segmented into multiple mail message objects or if this mail message object represents the first mail message created in the chain of mail message objects.
parent_message_id	string (24)	S	Message identifier of the root message if this message is an embedded message.
primary_content	string(1)	S	Identifies which email content is stored as the primary rendition. Valid values are: 0, meaning native message 1, meaning the message body 2, meaning message data
receive_date	Date	S	GMT date and time at which the message was received

Attribute	Datatype	Single/ repeating	Description
renditions_flag	string(1)	S	Manages email content stored as secondary renditions (using a bitmask). Valid values are: 1, meaning native message 2, meaning attachments 4, meaning message data 8, meaning subject
sent_date	Date	S	GMT date and time at which the message was sent.
to_addr	string(512)	S	The first 512 bytes of the addresses in the message's To list. Note: The remainder of the list, if any, is stored in the associated route table object.
total_attachments	integer	S	Total number of attachments in the message data

Media Profile

Purpose Stores a transformation profile as content.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_media_profile
 Object type tag: 08

A media profile object stores a transformation profile as content. Transformation profiles are XML documents that describe the transformations that can be performed on content, to change one format to another. A default set of profiles is provided with the Media Server. Additional profiles can be created by system administrators.

Table 2-103, page 304, lists the attributes defined for the type.

Table 2-103. Attributes defined for the media profile type

Attribute	Datatype	Single/ repeating	Description
cmd_file_paths	string(255)	R	Used internally by Documentum Media Transformation Services
source_formats	string(64)	R	Names of the source formats identified in the transformation profile.
target_formats	string(64)	R	Names of the target formats identified in the transformation profile. The format named in any given index position in source_formats can be transformed to the format identified in the corresponding index position in target_formats.

Method

Purpose Represents an external procedure that is invoked through the DO_METHOD function using either the DQL EXECUTE statement or the Apply method.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_method
 Object type tag: 10

A method object represents an external procedure that is invoked through the DO_METHOD function using either the DQL EXECUTE statement or the Apply method.

Table 2-104, page 305, lists the attributes defined for the type.

Table 2-104. Attributes defined for the method type

Attribute	Datatype	Single/ repeating	Description
launch_async	Boolean	S	<p>Indicates whether the procedure is to run asynchronously or not.</p> <p>If this is set to TRUE and the method is launched on the application server, setting SAVE_RESPONSE on to TRUE on the command line is ignored.</p> <p>This attribute setting is ignored if the method is launched on the method server or Content Server and SAVE_RESULTS is set to TRUE on the command line. The method is always launched synchronously.</p>
launch_direct	Boolean	S	<p>Used by Content Server. This is ignored when the method executed on</p>

Attribute	Datatype	Single/ repeating	Description
			the application server or method server.
			Controls whether the program is executed by the operating system's system or exec API call. If set to TRUE, the server uses the exec call to execute the procedure. In such cases, the method_verb must be a fully qualified path name. If set to FALSE, the server uses the system call to execute the procedure.
			Note: If SAVE_RESULTS is set to TRUE on the command line, the server ignores the setting of launch_direct and always uses the system call to execute the procedure.
method_args	string(255)	R	List of command line arguments for the procedure. Not currently used.
method_type	string(32)	S	If you are executing the method using Content Server or the method server and the executable is stored as content for the method, then setting this to dmawk or dmbasic, directs the server to add -f in front of the file name and to pass all arguments specified on the DO_METHOD command line to the program.

Attribute	Datatype	Single/ repeating	Description
method_verb	string(255)	S	<p>If you are executing the method on an application server, set this to java.</p> <p>Command-line name of the procedure or the name of the interpretive language that will execute the program file.</p> <p>Use dmbasic if the program is written in Docbasic.</p> <p>On Windows, use dmawk32 if the program is written in dmawk. On UNIX, use dmawk if the program is written in dmawk.</p> <p>Use the namespace in which to launch the method if the method is executing on the application server. For example:</p> <pre>com.foo.<package_name>.<class></pre> <p>or</p> <pre>fooPayMethod</pre>
run_as_server	Boolean	S	<p>If set to TRUE, it indicates that you want the method to run as the server account. The default is FALSE.</p> <p>run_as_server must be TRUE to execute a method on the method server or application server.</p>

Attribute	Datatype	Single/ repeating	Description
success_return_codes	integer	R	<p>Valid values for the a_last_return_code attribute of the job.</p> <p>If this is set, the a_last_return_code attribute must be set to one of these values or the dm_run_dependent_jobs method assumes that the job failed.</p> <p>The attribute is set to 0 for replication jobs.</p> <p>The attribute is ignored if it is not set.</p>
success_status	string(255)	S	<p>Defines the valid value for a_current_status in the completed job.</p> <p>If set, the job's a_current_status attribute value must match the success_status value after the job completes. If it does not, the dm_run_dependent_jobs method considers the job to have failed.</p> <p>For replication jobs, the value is set to "Replicate Operation Complete."</p> <p>The attribute is ignored if it is not set.</p>
timeout_default	integer	S	<p>The default time-out for this procedure. Used if no other time-out is specified on the command line. The default is 60 seconds.</p>

Attribute	Datatype	Single/ repeating	Description
timeout_max	integer	S	The maximum time-out that you can specify on the command line for this procedure. The default is 300 seconds.
timeout_min	integer	S	The minimum time-out that you can specify on the command line for this procedure. The default is 30 seconds.
trace_launch	Boolean	S	Controls whether internal trace messages generated by the executing program are logged.

If set to TRUE, the messages are put in the session log. If set to FALSE, the messages are not logged. This is FALSE by default.

Notes:

If trace_launch is TRUE and SAVE_RESULT is set to TRUE on the command line, the trace output is put into the result document.

If the method is executing on the application server, the trace records information about the method launch. Internal trace messages are stored in a result document if SAVE_RESULTS is set to TRUE.

Attribute	Datatype	Single/ repeating	Description
use_method _content	Boolean	S	<p>Indicates whether the program that you want to run is stored as content in the method object or as an external file. Set this to TRUE if you have specified an interpretive language for method_verb and the program you want to run is stored as the method's content. Set this to FALSE if method_verb contains a full file path to a program or script.</p> <p>The value is ignored if the method is executing on an application server.</p>
use_method _server	Boolean	S	<p>Whether to use the Method Server or Application Server to execute Dmbasic or Java methods. TRUE means to use the Method Server or application server. FALSE means to use the Content Server.</p> <p>If this is TRUE, the value is used in conjunction with the value of method_type to determine which server to use.</p> <p>Setting this attribute requires Superuser privileges.</p>

Module

Purpose Represents a business object module.

Implementation

Supertype: Folder
 Subtypes: Aspect Type, Validation Module
 Internal name: dmc_module
 Object type tag: 0b

A module object stores information about one business object module. The name of the module is recorded in the `object_name` attribute of the module object. The name must be unique. Module objects are stored in the repository in `/System/Modules/module_type`, where `module_type` is the value also found in the `a_module_type` attribute of the module.

The object type is installed by a script when Content Server is installed. Instances of the type are created and managed through Documentum Application Builder.

[Table 2-105, page 311](#), lists the attributes defined for the type.

Table 2-105. Attributes defined for the module type

Attribute	Datatype	Single/ repeating	Description
<code>a_bof_version</code>	<code>string(32)</code>	S	Version level of the BOF module. This is defined by the module's creator.
<code>a_interfaces</code>	<code>string(255)</code>	R	Fully qualified class names of the interfaces implemented by the BOF module

Attribute	Datatype	Single/ repeating	Description
a_module_type	string(32)	S	<p>Identifies the type of the module. For user-defined modules, the a_module_type value is also user defined. Modules provided by EMC Documentum typically have one of the following values:</p> <p>TBO</p> <p>SBO</p> <p>Aspect</p> <p>Note: Module objects that have Aspect as the module type are created and used internally only.</p>
a_req_module_interfaces	string(255)	R	<p>Object IDs of the dmc_jar objects associated with the modules identified in req_module_names. The IDs at a particular index position are associated with the module at the corresponding index position in req_module_names.</p> <p>Multiple object IDs at one index position are separated by commas.</p>
contact_info	string(256)	S	<p>Contact information about a party who can provide information about this module. This value is user-defined.</p>
implementation_technology	string(32)	S	<p>Identifies the technology used to create the module.</p>

Attribute	Datatype	Single/ repeating	Description
min_dfc_version	string(32)	S	Identifies the minimum DFC version level required to implement this module. The minimum value is 5.3.
module_description	string(255)	S	User-defined description of the module's functionality
primary_class	string(255)	S	The main implementation class. For .NET implementations, the value may consist of multiple comma-separated values.
req_module_names	string(64)	R	Names of modules on which this module depends

Module Config

Purpose Identifies a business object module to the workflow Timer servlet.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dmc_module_config
 Object type tag: 00

A module config object identifies a business object module to a workflow Timer servlet and stores argument values used by the servlet. The object type is installed using a script when Content Server is installed.

[Table 2-106, page 314](#) lists the attributes defined for the type.

Table 2-106. Attributes defined for the module config type

Attribute	Datatype	Single/ repeating	Description
id_attr_name	string(40)	R	Names of attributes having object ID values to be used by the servlet.
id_attr_value	ID	R	Object IDs stored in the attributes in the corresponding index position in id_attr_name.
r_aspect_name	string(64)	R	List of BOF module names
string_attr_name	string(40)	R	Names of attributes having string values to be used by the servlet
string_attr_value	string(2000)	R	Values of the attributes identified in string_attr_name. The value in a particular index position is the value of the attribute identified in the corresponding

Attribute	Datatype	Single/ repeating	Description
			index position in string_attr_name.

Mount Point

Purpose A mount point object describes a mounted directory in a Content Server installation.

Implementation

Supertype: SysObject
Subtypes: None
Internal name: dm_mount_point
Object type tag: 3e

A mount point object describes a mounted directory in a Content Server installation.

[Table 2-107, page 316](#), lists the attributes defined for the type.

Table 2-107. Attributes defined for the mount point type

Attribute	Datatype	Single/ repeating	Description
file_system_path	string(255)	S	Specifies the full path location of the mount point. The specification's syntax must be appropriate for the machine on which the location resides. For example, if the location is on a Windows host, the specification must be a Windows path specification.
host_name	string(128)	S	Identifies the machine on which this location resides. Use the machine's host name.
mac_preferred_alias	string(32)	S	Defines the alias that you prefer the system to use when a Macintosh client references the mount point. For example, a valid value might be Documentum.

Attribute	Datatype	Single/ repeating	Description
object_name	string(255)	S	Names the mount point object. This attribute is actually inherited from its sysobject supertype. However, it is included here because you are required to set this attribute when you create a mount point object.
security_type	string(32)	S	Specifies the security level applied to this directory location. Legal values are: public_open, public, and private. Defaults to private if not explicitly set.
unix_preferred_alias	string(32)	S	Defines the alias that you prefer the system to use when the client references the mount point. When the server is a UNIX server, you are not required to set this attribute.
win_preferred_alias	string(32)	S	Defines the alias that you prefer the system to use when a Windows client references the mount point. For example, valid values might be t:\ or k:\.

Network Location Map

Purpose Records information about network locations of client hosts.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_network_location_map
 Object type tag: 08

A network location object describes the network location of client sites. Instances of the object type are used to determine a user's location relative to the Content Servers that might provide them with content.

Table 2-108, page 318, lists the attributes defined for the type.

Table 2-108. Attributes defined for the network location map type

Attribute Name	Datatype	Single/ repeating	Description
begin_near_ip_ address	string(80)	R	The beginning of the IP address range representing IP addresses near to this network location Each index position represents the beginning of one range. The end of that particular range is recorded in the corresponding index position in end_near_ip_address.
default_netloc	Boolean	S	Whether this network location can serve as a default network location for a client, to be used when no other mapping for a user's IP address exists.

Attribute Name	Datatype	Single/ repeating	Description
end_near_ip_ address	string(80)	R	The end of the IP address range representing IP addresses near to this network location Each index position represents the end of one range. The beginning of that particular range is recorded in the corresponding index position in begin_near_ip_address.
netloc_display_ name	string(80)	S	User-defined display name or description of this network location.
netloc_ident	string(80)	S	User-defined unique identifier for this network location.

NLS DD Info

Purpose Records locale-dependent data dictionary information for an object type or attribute.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dm_nls_dd_info
 Object type tag: 4f

An nls dd info object holds locale-dependent data dictionary information for an object type or attribute. NLS DD info objects are created and managed by the server. Attributes in nls dd info objects are set when users add or change data dictionary information. Each nls dd info object contains the data dictionary information specific to one locale.

Not all nls dd info attributes apply to both object types and attributes. Some apply only to attributes.

[Table 2-109, page 320](#) lists the attributes that are applicable to both object types and attributes. [Table 2-110, page 321](#) lists the nls dd info attributes that are applicable only to attributes.

Table 2-109. Attributes applicable to object type and attributes

Attribute	Datatype	Single/ repeating	Description
comment_text	string(255)	S	User defined.
foreign_key_msgs	string(255)	R	The error message for foreign key constraint violations.
help_text	string(255)	S	User-defined help text for the attribute or type.
label_text	string(64)	S	User-defined label for the attribute or type.
map_data_string	string(128)	R	List of possible data values for the attribute.

Attribute	Datatype	Single/ repeating	Description
map_description	string(255)	R	Descriptions of the data values at the corresponding index levels in map_data_string.
map_display_string	string(128)	R	The character string to display for the data value at the corresponding index level in map_data_string.
primary_key_msg	string(255)	S	The error message for primary key constraint violations.
unique_key_msgs	string(255)	R	The error message for unique key constraint violations. The message at any one index level is displayed when the constraint at the corresponding index level is violated.
val_constraint_msg	string(255)	R	The error message for value constraint violations. The message at any one index level is displayed when the constraint at the corresponding index level is violated.

Table 2-110. Attributes applicable only to attributes

Attribute	Datatype	Single/ repeating	Description
category_name	string(64)	S	User defined.
def_value_length	integer	S	Length, in characters, of a typical value for the attribute. Zero indicates an unspecified (NULL) length.

Attribute	Datatype	Single/ repeating	Description
format_pattern	string(64)	S	For date attributes, the pattern to use for date interpretation.
format_pattern_tag	integer	S	Currently unused. (Currently, this is always 1.)
map_data_string	string(128)	R	List of possible data values for the attribute.
map_description	string(255)	R	Descriptions of the data values at the corresponding index levels in map_data_string.
map_display_string	string(128)	R	The character string to display for the data value at the corresponding index level in map_data_string.
not_null_msg	string(255)	S	The error message for NOT NULL constraint violations.
parent_id	ID	S	Object ID of the aggr domain object that references the nls dd info object.

Note

Purpose Represents an annotation for a document or other SysObject.

Implementation

Supertype: SysObject
Subtypes: None
Internal name: dm_note
Object type tag: 41

A note object represents an annotation for a document or other SysObject. The content of the annotation is stored in a content file associated with the note object. Note objects inherit all of their attributes from their supertype, SysObject. They have two methods defined for them: Addnote and Removenote.

Notepage

Purpose Represents richtext content that is authored and read online.

Implementation

Supertype: Document
Subtypes: None
Internal name: dmc_notepage
Object type tag: 09

A notepage object stores a content file in rich text format. You cannot check in or out, version, or create renditions of notepage objects. The content represented by notepage objects is created and used in the context of collaboration in Webtop. You must have installed Content Server with a Documentum Collaborative Services license to create a notepage object.

[Table 2-111, page 324](#), lists the attribute defined for the type.

Table 2-111. Attributes defined for the notepage type

Attribute	Datatype	Single/ repeating	Description
anchor_id	ID	R	Object IDs of any repository objects within the content file associated with the object.

Other File

Purpose Contains information about resource fork files for Macintosh.

Implementation

Supertype: Document
Subtypes: None
Internal name: dmi_otherfile
Object type tag: 23

An other file object contains information about resource fork files for Macintosh.

The type has two attributes, described in [Table 2-112, page 325](#).

Table 2-112. Attributes defined for the other file type

Attribute	Datatype	Single/ repeating	Description
data_ticket	integer	S	Value that tells the server how to access the file.
protocol	integer	S	Value indicating the Macintosh access protocol

Output Device

Purpose Contains a description of a printer that is accessible to users of the repository.

Implementation

Supertype: SysObject
Subtypes: None
Internal name: dm_outputdevice
Object type tag: 17

An output device object contains a description of a printer that is accessible to users of the repository.

[Table 2-113, page 326](#) lists the attributes defined for the type.

Table 2-113. Attributes defined for the output device type

Attribute	Datatype	Single/ repeating	Description
image_format	string(64)	S	Specifies the format that the printer uses to print images
os_name	string(32)	S	Name of the printer as defined in the <code>/etc/printcap</code> file.
output_command	string(128)	S	Contains the Unix print command sent to the printer
output_formats	string(128)	R	Contains the formats supported by the device.
output_mask	string(64)	S	Contains the job number of a queued print job, in a platform-dependent mask.
remove_command	string(128)	S	Contains the command that the server sends to the printer to cancel a print job
status_command	string(128)	S	Contains the command that the server sends when the user issues a Unix <code>lpq</code> command

Package

Purpose Serves as a container for one or more documents (or other SysObjects) moving through a workflow.

Implementation

Supertype: Persistent Object

Subtypes: None

Internal name: dmi_package

Object type tag: 49

A package object is used within workflows to deliver one or more documents (or other SysObjects) from the output port of a source activity to the input port of a destination activity. Packages are generated by the server from the attributes of the activity definition. Users cannot create or modify package objects.

[Table 2-114, page 327](#), lists the attributes defined for the type.

Table 2-114. Attributes defined for the package type

Attribute	Datatype	Single/ repeating	Description
i_acceptance_date	date	S	Records the date and time when a package arrived and was accepted by an input port.
i_package_oprtn	string(64)	S	Contains a value copied from the package definition upon creation of a package.
i_package_order	integer	S	Represents the order of where its associated package definition appears in the containing port definition.
r_act_seqno	integer	S	Contains the sequence number (r_act_seqno) of the activity in which the package is being handled.

Attribute	Datatype	Single/ repeating	Description
r_component _chron_id	ID	R	Chronicle ID of the object identified at the corresponding index position in r_component_id.
r_component_id	ID	R	Object IDs of the bound objects.
r_component_ name	string(80)	R	Object names of the bound objects. The name at a particular index position is the name of the object identified in r_component_id at the same index position.
			Whether this attribute is set is configurable. It is not set by default. Refer to Package control, page 225 , in <i>Content Server Fundamentals</i> for more information.
r_note_flag	integer	R	Identifies when a note was added to the package and how to handle the note in activity transitions. Valid values are: 0, Note added prior to activity; discard at next transition 1, Note added prior to activity; retain in next transition 2, Note added during this activity; send to next activity and then discard during transition from that activity 3, Note added during this activity; retain in subsequent transitions

Attribute	Datatype	Single/ repeating	Description
r_note_id	ID	R	Contains the ID of a dm_note object.
r_note_writer	string(32)	R	Contains the name of the user who made notes.
r_package_flag	integer	S	<p>Indicates whether the package is visible to the activity and whether it is allowed to be empty. The value is copied from the r_package_flag attribute in the activity object.</p> <p>Valid values are:</p> <p>0, meaning the package is invisible but cannot be empty</p> <p>1, meaning the package is visible and cannot be empty</p> <p>2, meaning the package is invisible but may be empty</p> <p>3, meaning the package is visible and may be empty</p>
r_package_label	string(32)	R	Contains a version label, if any.
r_package_name	string(16)	S	Contains the package name.
r_package_type	string(40)	S	Contains the real type name of the bound components.
r_port_name	string(16)	S	Contains the name (port_name) of the port of the activity run-time instance in which the package is being handled.
r_workflow_id	ID	S	Contains the object ID of the containing workflow.

Plugin

Purpose A plugin object represents a plugin library used by Content Server.

Implementation

Supertype: Document
 Subtypes: None
 Internal name: dm_plugin
 Object type tag: 67

Content Server uses plugin libraries to access external storage areas, including content-addressed storage, and for some forms of user authentication.

[Table 2-115, page 330](#) lists the attributes defined for the type.

Table 2-115. Attributes defined for the plugin type

Attribute Name	Datatype	Single/ repeating	Description
a_hw_platform	string(255)	R	Hardware architecture on which the plugin is running. The value is user defined. For example, a value might be Intel x86 or Sun SPARC Power PC
a_op_sys	string(255)	R	Identifies the operating system on which the plugin is running.
a_usage	string(255)	S	Description of the plugin's use.
a_plugin_type	string(255)	S	Indicates the type of plugin. The value is user-defined. Some example values are: batch file Bourne shell script C-shell script DLL docbasic executable java

Attribute Name	Datatype	Single/ repeating	Description
			java script sl so

Policy

Purpose A policy object contains the definition of a lifecycle.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_policy
 Object type tag: 46

Table 2-116, page 332 lists the attributes defined for the type.

Table 2-116. Attributes defined for the policy type

Attribute	Datatype	Single/ repeating	Description
action_object_id	ID	R	Object ID of the dm_procedure that implements state actions. All attributes specified in the procedure must be defined in the primary type.
alias_set_ids	ID	R	This is unused if java_methods is T. List of the object IDs of the alias set objects representing acceptable alias sets for the lifecycle.
allow_attach	Boolean	R	Indicates whether a document can be attached to this state with the Attach API method. The default value is false.
allow_demote	Boolean	R	Indicates whether the object can be demoted from this state. The default value is False.

Attribute	Datatype	Single/ repeating	Description
allow_schedule	Boolean	R	Indicates whether a state transition event (such as promote, demote, suspend, or resume event) can be scheduled when this state is the current state of the sysobject. The default value is true.
app_validation_id	ID	S	Identifies the procedure object that validates the lifecycle. If app_validation_ver is set, then app_validation_id contains the chronicle ID of the produce object. This is unused if java_methods is T.
app_validation_ver	string(32)	S	Version of the object identified in app_validation_id. This is unused if java_methods is T.
entry_criteria_id	ID	R	Object ID of the dm_func_expr that implements the entry criteria. All attributes specified in the expression must be defined in the primary type of this policy object. The source code for all expressions defined for this policy object are stored on page 0 of the policy's content, and all p-code is stored on page 1.

Attribute	Datatype	Single/ repeating	Description
exception_state	string(32)	R	Name of the exception state, if any. Optional for normal policy states. Ignored for exception states.
extension_type	string(32)	S	Name of the extension type used for the lifecycle.
i_state_no	integer	R	A 0-base internal integer number which uniquely identifies a state. Every time a new state is added, the highest number in the state chain plus one is used. This value, not the state name, is used in sysobject to record current state and resume state.
include_subtypes	Boolean	R	Indicates whether all subtypes of the type specified in the corresponding position of included_type are included. Possible values are: <ul style="list-style-type: none"> • false (all subtypes are NOT included) • true (all subtypes are included) <p>The attribute is always true when the object is created.</p>
included_type	string(27)	R	The acceptable object types. The first position stores the primary type associated with this policy object. The later positions are valid subtypes of the primary type. At least one value is required.

Attribute	Datatype	Single/ repeating	Description
java_methods	Boolean	S	Whether the entry criteria, actions, and validation methods are implemented in Java. T means they are written in Java. F means they are written in Docbasic.
r_definition_state	integer	S	<p>Status of the policy object. Possible values are:</p> <p>0, meaning draft 1, meaning validated 2, meaning installed</p> <p>The attribute is always 0 when the object is created.</p>
return_condition	integer	R	<p>Used when return_to_base is TRUE to identify which operations return an object to the policy's base state. Valid values are:</p> <ul style="list-style-type: none"> • 0, return to base state on Checkin or Save • 1, return to base state on checkin • 2, return to base state on Save • 4, return to base state on Saveasnew • 8, return to base state on Branch <p>The default value is 0.</p> <p>To identify multiple operations, sum the integers. For example, if you want to return to the base state on Checkin or Saveasnew, set the attribute to 5.</p>

Attribute	Datatype	Single/ repeating	Description
return_to_base	Boolean	R	<p>TRUE returns an object to the base state when the operations identified in the corresponding index position in return_condition are performed.</p> <p>The setting in each index position applies to the state in the corresponding index position in state_name.</p> <p>For a Checkin or Branch method, the affected object is the new version. For a Saveasnew, the affected object is the new copy. For a Save, the affected object is the saved object.</p> <p>The base state's entry criteria and actions are applied to the object when it is returned to base. If the entry criteria are not met, the method fails. If the actions fail, the object remains in the base state.</p> <p>FALSE means to leave the object in the current state.</p> <p>The default value is FALSE.</p>
state_class	integer	R	<p>Indicates whether the state is a normal state (with value 0), or an exception state (with value 1). The default value is 0.</p>
state_description	string(128)	R	<p>Purpose of the state.</p>

Attribute	Datatype	Single/ repeating	Description
state_name	string(32)	R	Name of the state. Unique within the policy definition to which the state belongs. A state name must start with a letter and cannot contain colons, periods, or commas.
state_type	string(32)	R	Name of the state type describing the state at the corresponding index position in state_name.
system_actions	ID	R	Object ID of tcf activity object describing a sequence of actions through modules implementing the IDfLifecycleAction interface
type_override_id	ID	R	Object ID of dm_aggr_domain which contains the type override information prepared by data dictionary. This attribute controls attributes of the document (whether attributes are read-only, or hidden, and so on) when the document is in this state. This is only set if java_methods is T.
user_action_id	ID	R	Object ID of the dm_procedure that implements user-defined actions, if the user_action_ver is unspecified; chronicle ID, otherwise. All attributes specified in the procedure must be defined in the primary type. This is unused if java_methods is T.

Attribute	Datatype	Single/ repeating	Description
user_action_service	string(128)	R	Names of the module objects implementing the user actions This is only set if java_methods is T.
user_action_ver	string(32)	R	The version label of user_action_id. If unspecified, the user_action_id is the id of the dm_procedure that contains user-defined actions. Otherwise, the user_action_id is the chronicle id. Together with user_action_id, this attribute allows users to associate a late bound user action object with a lifecycle. This is unused if java_methods is T.
user_criteria_id	ID	R	Object ID of the dm_procedure that implements the user-defined entry criteria for the state identified at the corresponding index position in state_name. All attributes specified in the procedure must be defined in the primary type. This is unused if java_methods is T.
user_criteria_service	string(128)	R	Names of the module objects implementing the entry criteria This is only set if java_methods is T.

Attribute	Datatype	Single/ repeating	Description
user_criteria_ver	string(32)	R	<p>The version label of user_criteria_id. If specified, the specified version of the procedure identified in user_criteria_id is used. If a version label is not specified, the version identified in user_criteria_id is used. Together with user_criteria_id, this attribute allows users to late bind an entry procedure with a lifecycle.</p> <p>This is unused if java_methods is T.</p>
user_postproc_id	ID	R	<p>Object ID of the dm_procedure that implements the user-defined procedure executed after entry into the state identified in the corresponding index position in state_name.</p> <p>This attribute may be used in conjunction with user_postproc_ver, to late-bind a particular version of the procedure to the lifecycle state. For information about late-binding, refer to Post-entry action definitions, page 287, in <i>Content Server Fundamentals</i>.</p> <p>This is unused if java_methods is T.</p>

Attribute	Datatype	Single/ repeating	Description
user_ postprocessing_ service	string(128)	R	Names of the module objects implementing the post-processing actions This is only set if java_methods is T.
user_postproc_ver	string(32)	R	Version labels identifying which versions of the procedures specified in user_postproc_id are to be executed. The label at a particular index position is applied to the procedure identified at the corresponding index position in user_postproc_id. Setting this attribute for a particular procedure allows you to late-bind a particular version of a procedure to the lifecycle state. For more information, refer to Post-entry action definitions, page 287 , in <i>Content Server Fundamentals</i> . This is unused if java_methods is T.
user_validation_ service	string(128)	R	Names of the module objects implementing user validation This is only set if java_methods is T.

Procedure

Purpose Procedure objects store Docbasic procedures that extend the behavior of the Documentum clients.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_procedure
 Object type tag: 08

Docbasic is an interpretive language that you can use to write applications to extend or customize the behavior of Documentum clients or Content Server. Depending on where they are stored in the repository, procedures can be executed automatically, when a user connects to a repository, or on demand, when users select a menu item. Users must have at least Read permission on the procedure object to execute it.

The type has one defined attribute, described in [Table 2-117, page 341](#).

Table 2-117. Attributes defined for the procedure type

Attribute	Datatype	Single/ repeating	Description
user_runnable	Boolean	S	Indicates whether the procedure is ready for use.

Process

Purpose A process object contains the definition of a workflow process.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_process
 Object type tag: 4b

A process object is created when a user creates a workflow definition in either Workflow Manager or Business Process Manager. There are three inherited attributes that are reserved for internal use for process objects: `a_special_app`, `a_status`, and `a_application_type`.

[Table 2-118, page 342](#) describes the attributes of the process type.

Table 2-118. Attributes defined for the process type

Attribute	Datatype	Single/ repeating	Description
<code>act_choose_by</code>	<code>string(32)</code>	R	Identifies by name the activity whose performer will choose the performer for the activity in the corresponding index position in <code>act_choose_for</code> .
<code>act_choose_for</code>	<code>string(32)</code>	R	Identifies by name the activity whose performer will be chosen by the performer of the activity named at the corresponding index position in <code>act_choose_by</code> .

Attribute	Datatype	Single/ repeating	Description
act_choose_name	string(32)	R	A user-defined name that links the performers chosen by the performer of the activity in the corresponding index position in act_choose_by to other activities.
is_private	Boolean	S	Indicates whether the definition is private or public. The default is private.
package_control	integer	S	<p>Controls whether Content Server sets the r_component_name attribute of a package object when the component name is identified in the Addpackage method.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> • 0, to allow the server to set the r_component_name attribute to the component name • 1, to require the server to set the attribute to a blank <p>The default value is 0.</p> <p>The value in package_control is only effective if the doctype attribute named wf_package_control_enabled is set to T.</p>
perf_alias_set_id	ID	S	Contains the object ID of the alias set object used to resolve performer aliases when a workflow based on this process definition is created.

Attribute	Datatype	Single/ repeating	Description
r_act_def_id	ID	R	Contains the object IDs of the activities (dm_activity objects) defined for this process.
r_act_name	string(32)	R	Contains the activity names. The name at each index position corresponds to the activity identified at the corresponding index position in r_act_def_id.
r_act_priority	integer	R	Indicates the importance of the activity compared to other activities in this process. The priority at each index position is assigned to the activity defined in r_act_def_id at the corresponding index position. The default value is 0.
r_act_type	integer	R	Indicates the type of the activity identified at the corresponding index position in r_act_def_id. Valid values are: 0, for step (default) 1, for begin 2, for end
r_definition_state	integer	S	Indicates the state of the process definition. Valid values are: 0, for draft 1, for validated 2, for installed
r_link_dest_act	string(32)	R	Identifies the destination activity for the link.

Attribute	Datatype	Single/ repeating	Description
r_link_dest_port	string(16)	R	Indicates the input port of the destination activity.
r_link_name	string(16)	R	Uniquely identifies the link.
r_link_src_act	string(32)	R	Identifies the source activity for the link.
r_link_src_port	string(16)	R	Indicates the output port of the source activity.

Public Key Certificate

Purpose Used to decrypt instructions encrypted using a private cryptographic key.

Implementation

Implementation

Supertype: SysObject

Subtypes: None

Internal name: dm_public_key_certificate

Object type tag: 08

A public key certificate object stores a public key certificate. The public key certificate is used by ACS and CACS servers to decrypt the instructions sent by Content Server in response to a request for content from an ACS or CACS server. There is only one public key certificate object in a repository. It is created automatically by Content Server. The ACL on the object give Delete permission to the object owner and the members of the dm_superuser group and Read permission to the world.



Caution: Public key certificate objects are for internal use only. Do not modify, remove, or add these objects.

Table 2-119, page 346, lists the attributes defined for the type.

Table 2-119. Attributes defined for the public key certificate type

Attribute	Datatype	Single/repeating	Description
certificate_value	string(1900)	S	Base 64-encoded DER certificate
key_identifier	string(40)	S	Base 64-encoded SHA1 digest of the DER public key

Attribute	Datatype	Single/repeating	Description
key_type	integer	S	Identifies what the key used for by Content Server. The only valid value is: 1, meaning used for ACS encryption
private_key_identifier	string(40)	S	Base 64-encoded SHA1 digest of the DER public key value in the dm_cryptographic_key.key_identifier attribute.

Qual Comp

Purpose A qual comp object represents a component routine.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_qual_comp
 Object type tag: 08

Component routines are written in a language compatible with the platform on which they will be executed. They are used by client applications. [Table 2–120, page 348](#) lists the attributes defined for the type.

Table 2-120. Attributes defined for the qual comp type

Attribute	Datatype	Single/ repeating	Description
available_offline	Boolean	S	Currently unused.
class_name	string(64)	S	Specifies the functionality provided by the component; for example, checkin or checkout.
client_capability	integer	S	Indicates who can use the component. Valid values are: 0, meaning no one can use 1, for consumer 2, for contributor 3, for consumer and contributor 4, for coordinator 5, for coordinator and consumer 6, for coordinator and contributor 7, for consumer, contributor, and coordinator 8, for administrator

Attribute	Datatype	Single/ repeating	Description
			<p>9, for administrator and consumer</p> <p>10, for administrator and contributor</p> <p>11, for administrator, contributor, and consumer</p> <p>12, for administrator and coordinator</p> <p>13, for administrator, coordinator, and consumer</p> <p>14, for contributor, coordinator, and administrator</p> <p>15, for all users</p>
component_obj_id	ID	S	<p>If there is no value specified for this attribute, then any user can use the component.</p> <p>Object ID of the component object associated with the qual comp object.</p>
is_global	Boolean	S	<p>TRUE indicates that the component is executable at all times. The user is not required to select an object before executing the component.</p> <p>FALSE indicates that there must be a currently selected object to execute the component.</p>

Attribute	Datatype	Single/ repeating	Description
nls_label	string(255)	S	Currently unused.
valid_groups	string(32)	R	Names of groups whose users can use this component. If there are no values, any user may use this component. If there are one or more groups named, users must belong to one of the groups to use the component.

Query

Purpose A query object stores a DQL query as content.

Implementation

Supertype: SysObject
Subtypes: None
Internal name: dm_query
Object type tag: 0a

Query objects are used by Desktop Client™. Their content files are saved DQL queries that a user can execute by selecting them from a menu. Users can create query objects in Desktop Client or through the API. To execute a saved query, a user must have at least Read permission on the query object.

The Query type has no defined attributes. It inherits all of its attributes from its supertype, SysObject.

Queue Item

Purpose A queue item object stores information about an object placed on a user's queue.

Implementation

Supertype: Persistent Object
Subtypes: None
Internal name: dmi_queue_item
Object type tag: 1b

Users and applications can query queue item objects directly or they can use the views provided by Content Server to obtain information about queued objects. Content Server views are dm_queue_view, dm_tasks_queued, and dm_tasks_dequeued. Refer to [Appendix A, The Documentum Views](#) for a description of these views.

Table 2-121, page 352 lists the attributes defined for the type.

Table 2-121. Attributes defined for the queue item type

Attribute	Datatype	Single/ repeating	Description
a_content_type	string(32)	S	File format to use when opening the object associated with the task from the inbox.
a_operations	string(16)	S	User-defined. The value is taken from the operations attribute on the router type.
actual_start_date	Date	S	Actual date that the task is started. This is set when the task is acquired.
content_type	string(32)	S	For router tasks: file format defined for the queued item, if any.
date_sent	Date	S	Not used for work items. The date the object was sent.

Attribute	Datatype	Single/ repeating	Description
delete_flag	Boolean	S	<p>Indicates whether the peer work item has been completed or if the event has been delivered or processed.</p> <p>0 (false) if not completed or delivered or 1 (true) if completed or processed.</p>
dependency_type	string(10)	S	<p>For queued router tasks, the value in the task's dependency_type attribute.</p> <p>For workflow tasks, this records the circumstances of the work item's creation. Valid values are:</p> <p>0, meaning Normal. Standard work item creation (default)</p> <p>1, meaning Revert. Item created by reverting a previous task</p> <p>2, meaning a resolution error occurred while trying to resolve an alias for a normal activity.</p> <p>3, meaning Revert and resolution error. Item was reverted and resolution error occurred for revert activity.</p> <p>4, meaning Delegated. The item was created when performer of previous task delegated the task or when Content Server automatically delegates the task to the user identified in the performer's delegated_user attribute.</p> <p>5, meaning Repeated. Item was created when activity</p>

Attribute	Datatype	Single/ repeating	Description
			performer chose to repeat the activity, with another performer.
			6, meaning Failed auto_delegation: Performer doesn't exist; performer is not available and has no indicated replacement; the indicated replacement does not exist; or the replacement availability turned off.
dequeued_by	string(32)	S	Contains the name of the user who completes the peer work item or the name of the user who dequeued the item, either explicitly or implicitly.
dequeued_date	Date	S	Contains the date and time that the item was dequeued or when an event was delivered and processed.
due_date	Date	S	Date by which the task or work item should be completed.
event	string(64)	S	A system- or user-defined event If dependency_type is 4 and the task was delegated by the performer, this is set to dm_delegatedworkitem. If the dependency_type is 4 and the task was automatically delegated by the server, the event is "dm_startedworkitem.
i_event_flags	integer	S	Used internally
instruction_page	integer	S	Not used for work items

Attribute	Datatype	Single/ repeating	Description
item_id	ID	S	<p>For a workflow task, the object ID of the peer work item.</p> <p>For events, the object ID of the object that generated the event. For checkin events, this is the object ID of the new version. For events queued to workflows, this is the workflow ID.</p> <p>If the queue item was generated by an Archive method, this is set to the object ID of the object selected by the method's predicate. If the predicate selects multiple objects, item_id is set to NULL.</p>
item_name	string(255)	S	<p>For a non-workflow event, the object name of the object that generated the event.</p> <p>For a workflow event, the object name of the process or workflow object that generated the event.</p> <p>Unused For workflow tasks or workflow-related events.</p>
item_type	string(32)	S	<p>For a workflow task, the object type of the peer work item.</p> <p>For an event, the object type of the object that generated the event.</p>
message	string(255)	S	<p>Message from the task's sender to the person receiving the queued item.</p>

Attribute	Datatype	Single/ repeating	Description
name	string(32)	S	Contains the name of the user to whom the queued item was sent.
next_tasks_type	string(10)	S	Obsolete. Not used for work items.
plan_start_date	Date	S	Obsolete. Not used for work items.
position	integer	S	If the queue item represents a work item for a work queue and the package associated with the work item has a defined skill level, this attribute is set to the integer value of that skill level.
priority	integer	S	User or application-interpreted value that defines the item's priority.
read_flag	Boolean	S	Not used for work items.
remote_pending	Boolean	S	Indicates whether the queue item must be distributed to other repositories. TRUE means that the object associated with the queue item must be sent to the repository defined in target_docbase.
router_id	ID	S	Object ID of the workflow or router that contains the queued task.
sent_by	string(32)	S	Name of the user that sent the queued item. This is the user's Content Server user name.
sign_off_date	Date	S	Not used for work items.

Attribute	Datatype	Single/ repeating	Description
sign_off_required	Boolean	S	Indicates if the task's user must sign off the task before forwarding the task.
sign_off_user	string(32)	S	Not used for work items.
source_docbase	string(120)	S	Name of the repository in which the queue item originated. This attribute has no value if the queue item is not a distributed item.
source_event	ID	S	Object ID of the queue item in the source repository.
source_stamp	integer	S	The i_vstamp value of the source queue item object.
stamp	ID	S	Object ID of the queue item object If the queue item object is a replica, this is the object ID of the replica.
supervisor_name	string(32)	S	Name of the router or workflow's supervisor.
target_docbase	string(120)	S	Name of the repository to which the queue item is sent. This attribute has no value if the queue item is not a distributed item.
task_name			Contains the string event, if this is an event, or contains the activity identifier if this is a work item. For a router task, this contains the task name.

Attribute	Datatype	Single/ repeating	Description
task_number	string(10)	S	<p>For work items, the sequence number within the workflow of the activity that generated the work item.</p> <p>For a router task, the number of the task.</p>
task_state	string(10)	S	<p>Indicates the status of the event, work item, or object queued for indexing.</p> <p>The status of the event or work item. One of: acquired, paused, or finished.</p> <p>If the queue item represents an object queued for indexing, the possible values are:</p> <ul style="list-style-type: none"> • blank, meaning the item has been queued but not yet picked up by the index agent • acquired, meaning the index agent has picked up the object but not yet completed processing • failed, meaning the item failed indexing and will not be available for searching • warning, meaning content indexing failed, but metadata was indexed • done, meaning indexing of content and metadata was successful. <p>For a router task, the valid values are: dormant,</p>

Attribute	Datatype	Single/ repeating	Description
task_subject	string(512)	S	<p>acquired, paused, ready, and finished.</p> <p>For workflow events, constructed from the value in dm_activity.task_subject.</p> <p>For dm_save events, this records the object IDs of the old and new storage areas if the object is saved with a change to a_storage_type.</p> <p>For dm_move_content events, this records the object IDs of the object and the old and new storage areas.</p> <p>For dm_destroy events, this records the object ID of the object's storage area.</p>
task_type	string(10)	S	Not used for work items.

Readcomment

Purpose Used to manage unread comments in a discussion.

Implementation

Supertype: Persistent Object
Subtypes: None
Internal name: dmc_readcomment
Object type tag: 00

A readcomment object contains information that allows the system to determine which comments in a particular discussion are unread by a particular viewer. These objects are created and managed internally. You cannot create these objects manually. Discussions, and consequently this object type, are supported only when you have installed Content Server with a Documentum Collaborative Services license.

[Table 2-122, page 360](#), lists the attributes defined for this type.

Table 2-122. Attributes defined for the readcomment type

Attribute	Datatype	Single/ repeating	Description
topic_id	ID	S	Object ID of topic object.
topic_viewed_ modtag	integer	S	Date and time at which the user in user_name last viewed the discussion represented by the topic. The value is copied from the topic object's last_update_modtag attribute.
user_name	string(32)	S	Name of user

Reference

Purpose A reference object contains the information that links an object in a remote repository to the local object that mirrors the remote object.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dm_reference
 Object type tag: 47

Users never work directly with reference objects. Content Server or client applications create them when users perform operations on remote objects. [Table 2–123, page 361](#) lists the attributes defined for the type.

Table 2-123. Attributes defined for the reference type

Attribute	Datatype	Single/ repeating	Description
binding_condition	string(32)	S	Identifies which version of the remote object to fetch when the object is accessed through its reference object. Valid values are: <ul style="list-style-type: none"> EXACT_ID, which directs the server to fetch the version that has the object ID specified in reference_by_id. VERSION_LABEL, which directs the server to fetch the version carrying the version label defined in binding_label. VERSION_LABEL is the default.

Attribute	Datatype	Single/ repeating	Description
binding_label	string(32)	S	Identifies the version to fetch for operations on the remote object when the value of binding_condition is VERSION_LABEL. The default version label is CURRENT.
i_external_rep	Boolean	S	Indicates whether the replication mode was external. TRUE indicates that external replication was used.
i_global_vstamp	integer	S	Version stamp of the remote object.
i_null_ref_flag	integer	S	Indicates the status of the remote object. Valid values are: 0, meaning the object exists 1, meaning the object does not exist 2, the object exists but no version matches the binding condition
i_ref_acl_domain	string(32)	S	Owner of the remote object's ACL.
i_ref_acl_name	string(32)	S	Name of the remote object's ACL.
i_ref_acl_vstamps	integer	S	Version stamp of the remote object's ACL.
i_replica_job_id	ID	S	Object ID of the replication job that created the replica. This only has a value if the reference object is the result of a replication job.
i_replica_master	ID	S	Object ID of the original (non-replica) object.

Attribute	Datatype	Single/ repeating	Description
i_replica_source	ID	S	Object ID of the replicated object. This value will be the same as that in i_replica_master unless the replicated object was itself a replica.
i_usage_count	integer	S	The number of system-created reference objects.
local_folder_link	string(200)	S	Identifies the local folder to which the mirror object is linked. If unspecified, this defaults to the user's default folder.
r_last_refresh	DATE	S	The time of the mirror object's last refresh.
r_mirror_object_id	ID	S	Object ID of the mirror object associated with the reference object.
r_next_refresh	DATE	S	Next scheduled refresh time for the mirror object.
r_ref_creator	string(32)	S	Name of the reference object's creator.
r_ref_object_id	ID	S	Object ID of the actual remote object to which the reference object points.
r_ref_version_id	ID	S	Object ID of the exact object that last satisfied the binding condition.

Attribute	Datatype	Single/ repeating	Description
r_reference_info	string(32)	S	Indicates how the reference is used. Valid values are: LINK_REFERENCE CHECKOUT_REFERENCE WORKFLOW_ REFERENCE ASSEMBLY_REFERENCE ANNOTATION_ REFERENCE REPLICA
r_refresh_status	string(255)	S	Status of the last mirror object refresh.
reference_db_name	string(120)	S	Name of the repository that contains the remote object. This attribute must be defined if reference_by_name is used to define the version tree containing the remote object. The attribute is optional if reference_by_id is used to define the object's version tree.
reference_by_id	ID	S	Object ID of the remote object. This is used to identify the version tree containing the remote object. If the binding_condition's value is EXACT_ID, then this identifies the exact object on the tree to which the reference points. If binding_condition is VERSION_LABEL, then this value only identifies the version tree containing the version specified in binding_label.

Attribute	Datatype	Single/ repeating	Description
reference_by_name	string(255)	S	<p>If reference_by_id is set, you cannot set reference_by_name.</p> <p>Full folder path of the remote object.</p> <p>If you set reference_by_name, you must also set reference_db_name.</p> <p>Reference_by_name and reference_by_id are mutually exclusive. You cannot set both.</p>
refresh_interval	integer	S	<p>Defines how often to poll the repository to determine if the mirror object needs to be refreshed. The value is specified in minutes.</p> <p>The default is one day.</p>

Registered

Purpose A registered object contains information about an underlying RDBMS table that has been registered with Content Server.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_registered
 Object type tag: 19

The information includes the names and datatypes of some or all of the columns in the table and a list of those columns belonging to indexes. To create a registered object, you use the DQL REGISTER statement. To destroy a registered object, use the UNREGISTER statement.

Note: Registered objects are not updated automatically if the definition of their corresponding RDBMS table changes.

Table 2-124, page 366 lists the attributes defined for the type.

Table 2-124. Attributes defined for the registered type

Attribute	Datatype	Single/ repeating	Description
column_count	integer	S	Number of columns in the table.
column_datatype	string(64)	R	List of the datatypes of the columns.
column_length	integer	R	Lengths of the columns that have a string data type.
column_name	string(64)	R	List of the names of the columns in the table. The names must consist of ASCII characters.
group_table_permit	integer	S	Defines the RDBMS table permit level assigned to the registered table's group.

Attribute	Datatype	Single/ repeating	Description
is_key	Boolean	R	Indicates if an index is built on the column.
owner_table _permit	integer	S	Defines the RDBMS table permit level assigned to the registered table's owner.
synonym_for	string(254)	S	Name of the table in the underlying RDBMS (can be an Oracle table synonym, or an MS SQL Server or Sybase table alias)
table_name	string(64)	S	Name of the table. The name must consist of ASCII characters.
table_owner	string(64)	S	Name of the owner of the RDBMS table (the person who created the RDBMS table).
world_table _permit	integer	S	Defines the RDBMS table permit level assigned to the world.

Registry

Purpose A registry object contains information about a registered event.

Implementation

Supertype: Persistent Object

Subtypes: None

Internal name: dmi_registry

Object type tag: 26

Users can register to receive notification of an event's occurrence. Users can also initiate auditing of events. Both actions generate registry objects. [Table 2-125, page 368](#) lists the attributes defined for the type.

Table 2-125. Attributes defined for the registry type

Attribute	Datatype	Single/ repeating	Description
a_authentication	integer	S	Defines whether an application should authenticate the user before the event. Valid values are: <ul style="list-style-type: none">• 0, meaning do not authenticate• 1, meaning authentication required This setting must be enforced by the application. Content Server ignores this attribute value.

Attribute	Datatype	Single/ repeating	Description
a_esignature_ required	integer	S	Whether the event requires an electronic signature created with Addesignature. The default value is 0, meaning FALSE, a signature is not required. A value of 1 means TRUE, a signature is required. This attribute value is not used by Content Server. It is for use by client applications.
audit_attr_names	string(40)	R	Names of the attributes to store in the attribute_list attribute of the audit trail entry.
audit_subtypes	Boolean	S	Whether to audit subtypes of the audited object type. The audit_subtypes setting is used only if the registered_id attribute identifies an object type.
controlling_app	string(32)	S	Identifies a particular application. If set, only objects controlled by that application are audited.
event	string(64)	S	Name of the event.
message	string(255)	S	For notifications, defines a message to be sent to the user with the notification. For events, defines a user-friendly event name.
oneshot	Boolean	S	Currently unused.

Attribute	Datatype	Single/ repeating	Description
policy_id	ID	S	Chronicle ID of a lifecycle. If set, only objects attached to the lifecycle are audited.
policy_state	string(32)	S	Name of a particular state in the lifecycle identified in policy_id. If set, only objects in that state are audited.
priority	integer	S	The event priority level. This is user-defined.
is_audittrail	Boolean	S	Defines whether this is a notification or audit trail event. Valid values are 0, for a notification event, and 1, for an audit event.
registered_id	ID	S	Identifies the object or object type being audited or for which a user has registered interest. For individual objects, this is the chronicle ID of the object. For object types, this is the object ID of the type's dm_type object.
sendmail	Boolean	S	Indicates whether to send email notification of the event.
sign_audit	Boolean	S	Whether Content Server must sign entries generated from this registration.
user_name	string(32)	S	Name of the user that registered for the event or initiated auditing.

Relation

Purpose Describes a relationship between two objects.

Implementation

Supertype: Persistent Object

Subtypes: Category Assign, Aspect Relation, State Extension, Validation Relation, DSM Sect Doc Attributes, DSM Doc Properties, DSM Study Report

Internal name: dm_relation

Object type tag: 37

A relation object describes a relationship between two objects. The attributes identify the objects and some behavioral characteristics of the relationship.

[Table 2-126, page 371](#) lists the attributes defined for the type.

Table 2-126. Attributes defined for the relation type

Attribute	Datatype	Single/ repeating	Description
child_id	ID	S	Identifies the object that is the child in the relationship.
child_label	string(32)	S	Version label of the specified child_id. This is optional. If set, the child_id must be the chronicle ID for the child.
description	string(255)	S	Not system defined. Provided for the user's convenience.
effective_date	date	S	Not system defined. Provided for the user's convenience.
expiration_date	date	S	Not system defined. Provided for the user's convenience.

Attribute	Datatype	Single/ repeating	Description
order_no	integer	S	Not system defined. This is provided for the user's convenience. For example, this could be used to order a set of relationships.
parent_id	ID	S	Identifies the object which is the parent in the relationship.
permanent_link	Boolean	S	Indicates if you want to maintain the relationship across versions of the parent object. If the relation is created by a 5.3 DFC, this defaults to the value of <code>dm_relation_type.permanent_link</code> . If the relation is created by the DMCL or a pre-5.3 DFC, the default is FALSE. Note: This attribute is deprecated in version 5.3. For more information, refer to The permanent_link and copy_child attributes , page 36.
relation_name	string(32)	S	Identifies a valid relation type object, which defines the type of relationship existing between the two objects.

Relation SSA Policy

Purpose Describes the relationship between a SysObject object and a storage policy.

Implementation

Supertype: Relation
 Subtypes: None
 Internal name: dm_relation_ssa_policy
 Object type tag: 37

A relation ssa policy object associates an ssa policy object, representing a content assignment policy, with an object type. The object type must be a SysObject or SysObject subtype. Relationships between object types and ssa policy objects are created through Documentum Administrator.

[Table 2-127, page 373](#), lists the attributes defined for the type.

Table 2-127. Attributes defined for the relation SSA policy type

Attribute	Datatype	Single/ repeating	Description
target_object_type	string(32)	S	Identifies the object type of the related object.
ssa_policy_id	ID	S	Object ID of the ssa policy object enforced for the related object.

Relation Type

Purpose Describes a relationship that can exist between two objects in the repository.

Implementation

Supertype: Persistent Object
Subtypes: dm_foreign_key
Internal name: dm_relation_type
Object type tag: 38

A relation type object describes a relationship that can exist between two objects in the repository.

When a user sets up a relationship between two objects, for example, by creating an annotation and attaching it to a document, the server creates an object of type dm_relation. The relation object links the annotation (child) to the document (parent) and describes the relationship between them by referring to the associated dm_relation_type object.

A user must have Sysadmin or Superuser privileges to create or destroy a relation type object.

For information about defining relationships, refer to [Relationships, page 34](#). For information about creating a relationship between two objects, refer to [User-defined relationships, page 157](#), in *Content Server Fundamentals*.

[Table 2–128, page 374](#), lists the attributes defined for the type.

Table 2-128. Attributes defined for the relation type object type

Attribute	Datatype	Single/ repeating	Description
child_parent_label	string(255)	R	User-defined label for a child-to-parent relationship.
child_type	string(32)	S	The object type of valid child objects in the relationship.

Attribute	Datatype	Single/ repeating	Description
copy_child	integer	S	<p>Specifies whether to copy the child in a relationship when permanent_link is T and the parent is copied. Valid values are:</p> <p>0, meaning do not copy the child</p> <p>1, meaning copy the child</p> <p>This attribute is used only by DFC when copying a parent object in a relationship and dm_relation_type. permanent_link is T. The DMCL and pre-5.3 versions of DFC do not use this attribute.</p>
description	string(250)	S	User-defined description of the relationship.
direction_kind	integer	S	<p>The direction of the relationship. Valid values are:</p> <ul style="list-style-type: none"> • 0, meaning from parent to child • 1, meaning from child to parent • 2, meaning bidirectional — objects are treated as siblings <p>The default is 0.</p>

Attribute	Datatype	Single/ repeating	Description
integrity_kind	integer	S	<p>Indicates how referential integrity is enforced. Valid values are:</p> <ul style="list-style-type: none"> • 0, meaning allow delete • 1, meaning restrict delete • 2, meaning cascade delete <p>The default is 0.</p>
parent_child_label	string(255)	R	User-defined label for a parent-to-child relationship.
parent_type	string(32)	S	Defines the object type of a valid parent object in the relationship.
permanent_link	Boolean	S	<p>Whether the relationship is maintained when the parent is copied or versioned. T means that the relationship is maintained with the new parent object. F means that the relationship is not maintained. The default value is F.</p> <p>This attribute is used only by DFC 5.3 when copying or versioning a parent object in a relationship. The DMCL and pre-5.3 versions of DFC do not use this attribute.</p> <p>Note: When an instance of the relationship is created using a 5.3 DFC-based application, this value is the default value assigned to <code>dm_relation.permanent_link</code> if that attribute is not set explicitly.</p>

Attribute	Datatype	Single/ repeating	Description
relation_name	string(32)	S	The name of the relationship. Names must be unique. The names of system-defined relationships begin with dm_.
security_type	string(10)	S	<p>Defines the security applied to objects of type dm_relation whose relation_name attribute value matches the relation_name attribute for the dm_relation_type object. Valid values are:</p> <p>SYSTEM</p> <p>Only users with Superuser or Sysadmin privileges can create, modify, or drop dm_relation objects having this relation_name</p> <p>PARENT</p> <p>Security is determined by the object type of the parent object participating in this kind of relationship.</p> <p>CHILD</p> <p>Security is determined by the object type of the child object participating in this relationship.</p> <p>NONE</p> <p>No security is applied to the dm_relation objects representing instances of this relationship.</p>

Replica Record

Purpose Contains information about replicated content in distributed storage areas.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dmi_replica_record
 Object type tag: 2d

A replica record object contains information about replicated content in distributed storage areas. The object is created and used by Content Server to manage distributed content storage areas.

[Table 2-129, page 378](#), lists the attributes defined for the type.

Table 2-129. Attributes defined for the replica record type

Attribute	Datatype	Single/ repeating	Description
component_id	ID	R	Object IDs of the components storage areas that contain replicas of the content file
data_ticket	integer	R	Information used internally to fetch and save the content.
epoch_number	integer	S	Used internally to manage distributed storage areas.
format_id	ID	S	Object ID of the format object representing the format of the content file associated with the replica record.
other_ticket	integer	R	Information used internally to fetch and save the content.

Attribute	Datatype	Single/ repeating	Description
owner_id	ID	S	Object ID of the distributed storage area that contains the content file.
r_storage_ticket	integer	S	Used internally to manage the content

Retainer

Purpose Describes a retention policy.

Implementation

Supertype: SysObject
 Subtypes: dmc_rps_retainer
 Internal name: dm_retainer
 Object type tag: 08

A retainer object records information about one retention policy. Retention policies are created through Retention Policy Services (accessed through Documentum Administrator) and require a Retention Policy Services license to create and use. Retainer objects are owned by a member of the dm_retention_managers group or by the group itself. Retainer objects must be uniquely named within the repository and cannot be versioned.

[Table 2-130, page 380](#), lists the attributes defined for a retainer object.

Table 2-130. Attributes defined for the retainer type

Attribute	Datatype	Single/ repeating	Description
disposition_rule	integer	S	Defines how RPS disposes of an object. Valid values are: 0, meaning DESTROY CHILDREN 1, meaning DESTROY_VDM 2, meaning DESTROY_ROOT

Attribute	Datatype	Single/ repeating	Description
enforcement_rule	integer	S	<p>Defines how objects under this policy are managed when the retention period expires. Valid values are:</p> <p>0, meaning objects are retained until the retention rule is removed using RPS.</p> <p>1, meaning Content Server disallows deletion if the objects are linked to an active retainer.</p> <p>2, meaning objects are retained until their associated structural retainer or retainers are removed</p> <p>Note: Disposition of objects associated with an active retainer is controlled by the retention policy's disposition rules defined within Retention Policy Services.</p>
immutability_rule	integer	S	<p>Defines whether to mark documents under the control of the policy immutable. Valid values are:</p> <p>0, meaning objects are not marked immutable</p> <p>1, meaning objects are marked immutable</p> <p>The default is 0.</p> <p>Notes on use:</p> <ul style="list-style-type: none"> • When the retainer is container-based (dm_retainer.retainer_strategy=1), the container is not affected by this setting. • When an object that is marked immutable due to its retention policy or policies is detached from all retainers, its

Attribute	Datatype	Single/ repeating	Description
			r_immutable_flag attribute is not reset to F unless the object is the CURRENT version and is not a cabinet or folder.
r_retention_status	integer	S	<p>Identifies the status of the retainer. Valid values are:</p> <p>0, meaning active—objects under the control of the retainer may not be deleted and new objects may be added to the retainer</p> <p>1, meaning locked—the retainer is active but no new objects may be placed under its control</p> <p>2, meaning inactive—objects under the control of the retainer may be deleted from the repository</p> <p>The default is 0.</p>
rendition_rule	integer	S	<p>Defines how multiple content files for one document are handled. Valid values are:</p> <p>0, meaning all content files associated with a document under the control of the policy are under retention</p> <p>1, meaning only the primary content files of a document under the control of the policy are under retention</p>

Attribute	Datatype	Single/ repeating	Description
retainer_root_id	ID	S	<p>Identifies the object to which this retainer is applied.</p> <p>If dm_retainer.retainer_strategy is set to 0 (object-based retention), retainer_root_id is the object ID of the individual object under retention.</p> <p>If retainer_strategy is set to 1 (container-based retention), retainer_root_id is the object ID of the folder containing the objects under retention.</p>
retainer_strategy	integer	S	<p>Defines whether the retainer controls a single object or multiple objects placed in a container. Valid values are:</p> <p>0, meaning the retainer controls a single object</p> <p>1, meaning the retainer controls multiple objects in a container</p> <p>The default is 0.</p>
retention_date	date	S	Retention value when dm_retainer.retention_rule_type is set to 1, meaning date.
retention_interval	integer	S	<p>Retention value when dm_retainer.retention_rule_type is set to 2, meaning interval.</p> <p>The value is interpreted in seconds.</p>

Attribute	Datatype	Single/ repeating	Description
retention_rule_type	integer	S	Specifies how the retention period is defined. Valid values are: 0, meaning there is no specified retention period 1, meaning the retention period is defined as a specific date 2, meaning the retention period is defined as an interval The default is 0.
retention_storage_class	string(40)	S	Reserved for future use
version_rule	integer	S	Defines how retention is applied to new versions of documents under the control of the retainer. The only valid value is 0, meaning that the retention period applies only to the specific version to which the retainer was attached.

Richtext

Purpose Represents rich text content associated with a SysObject

Implementation

Supertype: SysObject
 Subtypes: dmc_comment
 Internal name: dmc_richtext
 Object type tag: 08

A richtext object represents richtext content associated with a SysObject either directly, through a relationship, or indirectly as a comment in a topic thread. The actual content is stored in either the content_value attribute or as an actual content file associated with the richtext object. Richtext objects are supported only when you have installed Content Server with a Documentum Collaborative Services license.

[Table 2-131, page 385](#), lists the attributes defined for the type.

Table 2-131. Attributes defined for the richtext type

Attribute	Datatype	Single/ repeating	Description
anchor_id	ID	R	Object IDs of repository objects found in the content
content_value	string(2000)	S	Stores the first 2000 bytes of the content
has_content	Boolean	S	Whether the content is fully contained in the content_value attribute or whether there is also a content file attached to the object.

T means that there is a content file. F means that the content is fully contained in content_value.

Room

Purpose Provides an additional access management layer for SysObjects.

Implementation

Supertype: Folder
 Subtypes: None
 Internal name: dmc_room
 Object type tag: 0b

A room is a special folder that provides additional, optional functionality to control access to the objects in the folder when the objects are accessed through the room. Rooms are supported only when you have installed Content Server with the Documentum Collaborative Services license.

Table 2-132, page 386, lists the attributes defined for the type.

Table 2-132. Attributes defined for the room type

Attribute	Datatype	Single/ repeating	Description
builtin_groups	string(32)	R	Names of private groups in the room. The first four index positions are reserved for the system-defined groups in room: [0]=Members [1]=Owners [2]=Contributors [3]=Visitors
default_contrib_permit	integer	S	Default base object-level permission for the contributor's group applied to objects governed by this room

Attribute	Datatype	Single/ repeating	Description
default_contrib_ xpermit	string(32)	S	Default extended permission for the contributor's group applied to objects governed by this room
default_owner_ permit	integer	S	Default base object-level permission for the owner's group applied to objects governed by this room
default_owner_ xpermit	string(32)	S	Default extended permission for the owner's group applied to objects governed by this room
is_public	Boolean	S	<p>Whether the room is a public or private room.</p> <p>T means the room is a public room.</p> <p>F means the room is a private room and only members of the Members group (builtin_groups[0]) can access objects in the room.</p> <p>The only valid value is currently F.</p>
only_owners_ ungovern	Boolean	S	<p>Whether membership in the Owners groups is an additional requirement for users trying to remove an object from the room.</p> <p>T means that a user must be a member of the Owners group in addition to having Write and Change_permit permissions on an object to remove that object from the room.</p> <p>F means that any user in the room with Write and Change_permit permissions</p>

Attribute	Datatype	Single/ repeating	Description
status_text	string (<i>maximum</i>)	S	on the object can remove it from the room. Text description of the room's status. The length of this attribute is the maximum allowed by the underlying database.
status_value	integer	S	Application-defined status value of the room.

RouteCase Condition

Purpose Records a route case condition expression for an automatic transition of a workflow activity.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dmc_routeCase_condition
 Object type tag: 00

A routeCase condition object records a conditional expression in a route case condition. You cannot create these objects directly. They are created when an addConditionRouteCase method (defined for the IDfActivity interface) is executed from Business Process Manager 5.3 SP1 (or later) to save an activity's route case conditions when one or more of the route cases contains an XPath expression.

[Table 2-133, page 389](#), lists the attributes defined for the type.

Table 2-133. Attributes defined for the routeCase condition type

Attribute	Datatype	Single/ repeating	Description
a_attribute_name	string(32)	S	Name of the attribute referenced in the expression, if any
a_boolean_value	Boolean	S	Value to be used in the comparison if the datatype identified in a_value_type is Boolean. This is not set if unless the datatype is Boolean.
a_double_value	double	S	Value to be used in the comparison if the datatype identified in a_value_type is double. This is not set if unless the datatype is double.

Attribute	Datatype	Single/ repeating	Description
a_id_value	ID	S	Value to be used in the comparison if the datatype identified in r_value_type is ID. This is not set if unless the datatype is ID.
a_int_value	integer	S	Value to be used in the comparison if the datatype identified in a_value_type is integer. This is not set if unless the datatype is integer.
a_object_alias	string(32)	S	Name of the package, or manifest values referring to the workflow or work item
a_relational_op	integer	S	The relation operator in the condition. Valid values are: 0, meaning = 1, meaning \diamond 2, meaning < 3, meaning > 4, meaning <= 5, meaning >=
a_repeating_attr_flag	integer	S	Indicates whether the attribute named in a_attribute_name is a repeating attribute and if it is a repeating attribute, which values to examine when evaluating the condition. Valid values are: -1, meaning the attribute is not a repeating attribute 0, meaning ANY 1, meaning ALL 3, meaning LAST 2, meaning FIRST

Attribute	Datatype	Single/ repeating	Description
a_string_value	string(1024)	S	Value to be used in the comparison if the datatype identified in a_value_type is string. This is not set if unless the datatype is string.
a_time_value	date	S	Value to be used in the comparison if the datatype identified in a_value_type is Date. This is not set if unless the datatype is Date.
a_value_type	integer	S	Data type of the value in the relational expression. The data type is expressed as an IDfValue constant.
a_xpath_datatype	string(64)	S	The xschema's built-in datatype name. This is set only if the condition includes an XPath expression.
a_xpath_def_namespace_uri	string(255)	S	The schema URI for the default XPath namespace This is set only if the condition includes an XPath expression.
a_xpath_expression	string(1024)	S	An XPath expression. This is set only if the condition includes an XPath expression.

Attribute	Datatype	Single/ repeating	Description
a_xpath_ namespace_prefix	string(64)	R	<p>The prefixes that alias the namespace URIs within the XPath expression</p> <p>This is set only if the condition includes an XPath expression.</p>
a_xpath_ namespace_uri	string(255)	R	<p>Namespace URIs of the XPath expression components</p> <p>This is set only if the condition includes an XPath expression.</p>
a_xpath_value	string(1024)	S	<p>Literal value used in the XPath transition condition evaluation. The value is in the format in which it is found in the XML document.</p> <p>This is set only if the condition includes an XPath expression.</p>
r_aspect_name	string(64)	R	Used internally

Route Table

Purpose Records routing information from an email message.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dm_route_table
 Object type tag: 00

A route table object is created when an email message is archived. Route table objects store the routing information found in the To, From, bcc, and cc lists. Each object records one address in the message. Route table objects are primarily used by personal and compliance archiving applications.

[Table 2-134, page 393](#), lists the attributes defined for the object type.

Table 2-134. Attributes defined for the route table type

Attribute	Datatype	Single/ repeating	Description
addr_id	string(16)	S	Hash value of the email address
message_id	string(24)	S	Message identifier recorded in the associated dm_mail_message.message_id attribute
route_bit_flags	string(1)	S	Used internally
route_type	string(1)	S	Identifies the source of the email address. Valid values are: 1, meaning To 2, meaning From 3, meaning cc 4, meaning bcc 5, meaning Distributed list

6, meaning any other source

RPS Action

Purpose Identifies the Java class used to execute an action.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dmc_rps_action
 Object type tag: 08

An rps action object identifies the class and, optionally the interface within the class, that is executed to perform an action.



Caution: This object type is installed with the RPS DocApp, and instances of the type are created and maintained through the Retention Policy Services product. Neither the object type nor instances of the type can be changed or modified by users or client applications other than RPS.

[Table 2-135, page 395](#), lists the attributes defined for the type and those inherited attributes that are used for a specific purpose by the type.

Table 2-135. Attributes defined for the rps action type

Attribute	Datatype	Single/ repeating	Description
action_type	integer	S	Identifies what kind of action is represented by the rps action object. The only valid value is 0, meaning a notification.
class	string(128)	S	Path to the Java class that executes the action
interface	string(128)	S	Path to the Java class interface that executes the action
object_name	string(255)	S	Name of the action
subject	string(192)	S	Description of the action

RPS Action Rel

Purpose Relates an rps phase rel object to an rps action object.

Implementation

Supertype: Relation
 Subtypes: None
 Internal name: dmc_rps_action_rel
 Object type tag: 37

An rps action rel object associates a particular rps phase with an action and identifies the execution rule for the action.



Caution: This object type is installed with the RPS DocApp, and instances of the type are created and maintained through the Retention Policy Services product. Neither the object type nor instances of the type can be changed or modified by users or client applications other than RPS.

[Table 2-136, page 396](#), lists the attributes defined for the type and those inherited attributes that are used for a specific purpose by the type.

Table 2-136. Attributes defined for the rps action rel type

Attribute	Datatype	Single/ repeating	Description
arg_object_id	ID	R	Used internally. Value dependent on the type of action identified in the associated rps action object.
child_id	ID	S	Object ID of the rps action object
execution_rule_id	ID	S	Object ID of the execution rule that is the criterion for action
object_name	string(255)	S	Name of the object
parent_id	ID	S	Object ID of the rps phase rel object representing the phase associated with the action

RPS Authority

Purpose Identifies the persons or agencies that are authorized to determine whether an object has fulfilled its phase conditions.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dmc_rps_authority
 Object type tag: 08

An rps authority object records information about the authorities authorized to determine whether an object has fulfilled its phase conditions. The rps authority objects are related to objects of type dmc_rps_phase_rel objects (representing phases in a retention policy lifecycle). The relationship is named dmc_rps_phase_authority_rel_type.



Caution: This object type is installed with the RPS DocApp, and instances of the type are created and maintained through the Retention Policy Services product. Neither the object type nor instances of the type can be changed or modified by users or client applications other than RPS.

Table 2-137, page 397, lists the attributes defined for the type and those inherited attributes that are used for a specific purpose by the type.

Table 2-137. Attributes defined for the RPS authority type

Attribute	Datatype	Single/ repeating	Description
authorizers_id	ID	R	Object IDs of the rps contact objects representing users or agencies.
is_authority_valid	Boolean	S	Whether this authority is valid. T means that this authority can determine phase fulfillment; F means that it cannot be used to determine phase fulfillment.

Attribute	Datatype	Single/ repeating	Description
object_name	string(255)	S	Name of the authorizing body or person. This attribute is required.
title	string(400)	S	User-defined description of the authority.

RPS Base Date

Purpose Records which attribute in an object type is used to populate the `retention_base_date` attribute in the `dmc_rps_retainer` objects for instances of the type.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: `dmc_rps_base_date`
 Object type tag: 08

An rps base date object records the attribute in an object type whose value is used to populate the `retention_base_date` attribute in the `dmc_rps_retainer` objects with which instances of object type are associated.



Caution: This object type is installed with the RPS DocApp, and instances of the type are created and maintained through the Retention Policy Services product. Neither the object type nor instances of the type can be changed or modified by users or client applications other than RPS.

Table 2–138, page 399, lists the attributes defined for the type.

Table 2-138. Attributes defined for the RPS base date type

Attribute	Datatype	Single/ repeating	Description
<code>object_type</code>	<code>string(32)</code>	S	Name of the object type. The object type must be <code>dm_sysobject</code> or one of its subtypes. Use the type's internal name. This is a required attribute.

Attribute	Datatype	Single/ repeating	Description
base_date_name	string(32)	S	Name of an attribute defined for or inherited by the object type identified in object_type. The attribute must be of date/time datatype.
title	string(32)	S	This is a required attribute. User-defined description of the attribute mapping.

RPS Child Strategy

Purpose Records the name of the class called to apply an rps retainer object to the objects associated with a dm_retainer object.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dmc_rps_child_strategy
 Object type tag: 08

An rps child strategy object defines the Java class called to attach an rps retainer to objects that are stored in a container that is associated with an rps retainer object.



Caution: This object type is installed with the RPS DocApp, and instances of the type are created and maintained through the Retention Policy Services product. Neither the object type nor instances of the type can be changed or modified by users or client applications other than RPS.

[Table 2-139, page 401](#), lists the attributes defined for the type and those inherited attributes that have a specific use in the type.

Table 2-139. Attributes defined for the RPS child strategy type

Attribute	Datatype	Single/ repeating	Description
class	string(128)	S	Path to the Java class that executes based on the selected child strategy. This is a required attribute.
interface	string(128)	S	Path the interface for the class specified in class.
is_container_aging	Boolean	S	Whether the rps retainer ages. T means the container ages; F means it does not.

Attribute	Datatype	Single/ repeating	Description
object_name	string(255)	S	Name of the child strategy. Valid values are: ONE_FOR_ALL ONE_FOR_EACH RECORDS_STRATEGY EMAIL_STRATEGY This inherited attribute is required.
retainer_strategy	integer	S	Inherited from rps retainer, this indicates whether all objects are to use the same retainer or each object must have its own retainer. Valid values are: 0, meaning that each object must have its own retainer 1, meaning that all objects are to use the same retainer

RPS Condition

Purpose Represents a template from which rps event objects are generated.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dmc_rps_condition
 Object type tag: 08

An rps condition object defines a condition associated with a particular phase of retention policy lifecycle. RPS condition objects are related to phases (dmc_rps_phase_rel objects) through a relationship named dmc_rps_phase_condition_rel_type.



Caution: This object type is installed with the RPS DocApp, and instances of the type are created and maintained through the Retention Policy Services product. Neither the object type nor instances of the type can be changed or modified by users or client applications other than RPS.

[Table 2-140, page 403](#), lists the attributes defined for the type and the inherited attributes used for a specific purpose by the type.

Table 2-140. Attributes defined for the RPS condition type

Attribute	Datatype	Single/ repeating	Description
category	string(32)	S	Identifies the category to which the condition is assigned. This is a required attribute.
object_name	string(255)	S	The name of the condition. For example: Project Closed or Employee Retired This inherited attribute is a required attribute.
title	string(400)	S	User-defined description of the condition.

RPS Contact

Purpose Records information about a person defined in Retention Policy Manager.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dmc_rps_contact
 Object type tag: 08

RPS contact objects contain information about persons defined in Retention Policy Manager. The objects are referenced by rps hold objects, rps authority objects, and rps event objects.

Note: The persons defined by rps contact objects are not necessarily repository users.



Caution: This object type is installed with the Retention Policy Services DocApp, and instances of the type are created and maintained through the Retention Policy Services product. Neither the object type nor instances of the type can be changed or modified by users or client applications other than RPS.

[Table 2-141, page 404](#), lists the attributes defined for the type and those inherited attributes used for a specific purpose by the type.

Table 2-141. Attributes defined for the RPS contact type

Attribute	Datatype	Single/ repeating	Description
email	string(32)	R	Email addresses for the contact
phone	string(64)	R	Telephone numbers for the contact
object_name	string(255)	S	Name of the contact
title	string(400)	S	User-defined description of the contact
user_id	ID	S	Object ID of the contact's user object in the repository if the contact is a repository user.

RPS Disposition Method

Purpose Records the name of the Java class called to dispose of a retained object.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dmc_rps_disposition_method
 Object type tag: 08

An rps disposition method object identifies the Java class called to dispose of an object at the end of retention. Which class is called is dependent on the selected disposition option.



Caution: This object type is installed with the Retention Policy Services DocApp, and instances of the type are created and maintained through the Retention Policy Services product. Neither the object type nor instances of the type can be changed or modified by users or client applications other than RPS.

[Table 2-142, page 405](#), lists the attributes defined for the type and the inherited attribute used for a specific purpose by the type.

Table 2-142. Attributes defined for the RPS disposition method type

Attribute	Datatype	Single/ repeating	Description
class	string(128)	S	Path to the Java class invoked to dispose of an object.
is_container_ destroyed	Boolean	S	Whether the container was destroyed on disposition. T means the container is destroyed; F means that it was not.

Attribute	Datatype	Single/ repeating	Description
object_name	string(255)	S	Name of the disposition method. Valid values are: UNKNOWN REVIEW TRANSFER/EXPORT DESTROY
priority_no	integer	S	Defines the action on a retained object when two of its retainers qualify for disposition during the same qualification period. Valid values are: 10, undefined action 30, review 50, transfer or export 70, destroy

RPS Event

Purpose Represents one instance of a dmc_rps_condition object.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dmc_rps_event
 Object type tag: 08

An rps event object is an instance of a condition defined in an rps condition object. It contains data specific to that instance of the condition. There is one rps event object generated for each rps condition on each rps phase rel object. RPS event objects are associated with the generating rps condition object through a relation object named dmc_rps_condition_event_rel_type.



Caution: This object type is installed with the Retention Policy Services DocApp, and instances of the type are created and maintained through the Retention Policy Services product. Neither the object type nor instances of the type can be changed or modified by users or client applications other than RPS.

Table 2-143, page 407, lists the attributes defined for the type and those inherited attributes used for a specific purpose by the type.

Table 2-143. Attributes defined for the RPS event type

Attribute	Datatype	Single/ repeating	Description
a_last_review_date	Date	S	The most recent date and time at which the condition was fulfilled.
condition_id	ID	S	Object ID of the rps condition object from which this event was generated This is a required attribute.

Attribute	Datatype	Single/ repeating	Description
fulfilled_by	ID	R	Object IDs of the rps contact objects representing contacts who have indicated that the event has been fulfilled.
fulfillment_message	string(255)	S	User-defined message to be displayed when this event is fulfilled.
object_name	string(255)	S	Name for the event. This attribute is required.
title	string(400)	S	User-defined description of the event.

RPS Execution Rule

Purpose Defines a criterion for execution of an action.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dmc_rps_execution_rule
 Object type tag: 08

An rps execution rule object defines when a particular action is to be executed against an object controlled by a retention policy. The definition is recorded in the object name. For example, if `object_name` is "Phase Entry", the rule is executed only when the target object experiences phase entry.

The object type has no defined attributes.



Caution: This object type is installed with the Retention Policy Services DocApp, and instances of the type are created and maintained through the Retention Policy Services product. Neither the object type nor instances of the type can be changed or modified by users or client applications other than RPS.

[Table 2-144, page 409](#), lists the inherited attributes that it uses for a specific purpose.

Table 2-144. Attributes defined for the rps execution rule type

Attribute	Datatype	Single/ repeating	Description
<code>object_name</code>	string(255)	S	Identifies the criterion for execution of the associated action
<code>subject</code>	string(192)	S	Description of the execution rule

RPS Hold

Purpose Records information about a hold defined in Retention Policy Services.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dmc_rps_hold
 Object type tag: 08

An rps hold object represents a hold defined in RPS. Holds prevent deletion of an object when the object is available for disposition at the end of its retention. A hold is associated with the object through a relationship named dmc_rps_hold_object_rel_type.



Caution: This object type is installed with the Retention Policy Services DocApp, and instances of the type are created and maintained through the Retention Policy Services product. Neither the object type nor instances of the type can be changed or modified by users or client applications other than RPS.

[Table 2-145, page 410](#), lists the attributes defined for the type and those inherited attributes that are used for a specific purpose by the type.

Table 2-145. Attributes defined for the RPS hold type

Attribute	Datatype	Single/ repeating	Description
a_last_review_date	Date	S	Date on which the status of the hold must be reviewed
a_retention_date	Date	S	Date when the hold was applied to the object
approvers_id	ID	R	Object IDs of the rps contact objects representing users who have approved the hold
object_name	string(255)	S	Name of the hold This is a required attribute.

Attribute	Datatype	Single/ repeating	Description
requestors_id	ID	R	Object IDs of the rps contact objects representing the users who requested the hold
title	string(400)	S	This is a required attribute. User-defined description of the hold

RPS Notification

Purpose Identifies the recipients of notifications

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dmc_rps_notification
 Object type tag: 08

An rps notification object records the contacts to be sent notifications for a particular action and those contacts who acknowledge the notification.



Caution: This object type is installed with the Retention Policy Services DocApp, and instances of the type are created and maintained through the Retention Policy Services product. Neither the object type nor instances of the type can be changed or modified by users or client applications other than RPS.

[Table 2-146, page 412](#), lists the attributes defined for the type and those inherited attributes that are used for a specific purpose by the type.

Table 2-146. Attributes defined for the rps notification type

Attribute	Datatype	Single/ repeating	Description
a_last_review_date	Date	S	Date and time at which the notification was sent
acknowledged_ by_id	ID	R	Object IDs of the rps contact objects representing contacts who have acknowledged the notification
action_rel_id	ID	S	Object ID of the rps action rel object that identifies the action for a given phase
contact_id	ID	R	Objects IDs of the rps contact objects representing contacts to be notified

Attribute	Datatype	Single/ repeating	Description
number_sent	integer	S	Number of times the notification has been sent
object_name	string(255)	S	Application-defined identifier for the notification
subject	string(192)	S	Description of the notification

RPS Phase Rel

Purpose Represents one phase of an RPS retention policy.

Implementation

Supertype: State Extension
 Subtypes: None
 Internal name: dmc_rps_phase_rel
 Object type tag: 37

An rps phase rel object represents one phase in an rps retention policy. The object associates a lifecycle definition (dm_policy object) with the dmc_rps_retention_policy. The parent_id attribute identifies the dm_policy object on which the retention policy is based and the child_id attribute identifies the rps retention policy. The attributes defined for the rps phase rel object define the retention phase.

The relationship represented by the rps phase rel object is named dmc_rps_phase_rel_type.



Caution: This object type is installed with the Retention Policy Services DocApp, and instances of the type are created and maintained through the Retention Policy Services product. Neither the object type nor instances of the type can be changed or modified by users or client applications other than RPS.

Table 2-147, page 414, lists the attributes defined for the type.

Table 2-147. Attributes defined for the RPS phase rel type

Attribute	Datatype	Single/ repeating	Description
cutoff_day	integer	S	The day in a month up to which cutoff will be applied. The day is specified by its calendar number.
cutoff_month	integer	S	The month in the year in which cutoff will be applied. The month is specified by its numeric position in the year.

Attribute	Datatype	Single/ repeating	Description
cutoff_period	string(32)	S	Defines an interval used to round-up cutoff values during processing. Values are: MONTHLY QUARTERLY SEMI_ANNUALLY ANNUALLY DISABLED
duration_days	integer	S	Number of days in the duration of this phase.
duration_months	integer	S	Number of months in the duration of this phase
duration_years	integer	S	Number of years in the duration of this phase
is_final_phase	Boolean	S	Whether this phase is the final phase in the retention lifecycle. T means that this is the final phase; F means that it is not.
phase_name	string(32)	S	Name of the phase. This is a derived value, not user-defined.
storage_id	ID	S	Object ID of the storage area to which to move objects when they complete this phase.

RPS Retainer

Purpose Records information about the state of an object's retention.

Implementation

Supertype: Retainer
 Subtypes: None
 Internal name: dmc_rps_retainer
 Object type tag: 08

An rps retainer object mirrors an object's association with a dm_retainer object and includes additional information about the object's status in the retention lifecycle.



Caution: This object type is installed with the Retention Policy Services DocApp, and instances of the type are created and maintained through the Retention Policy Services product. Neither the object type nor instances of the type can be changed or modified by users or client applications other than RPS.

[Table 2-148, page 416](#), lists the attributes defined for the type and those inherited attributes used for a specific purpose by the type.

Table 2-148. Attributes defined for the RPS retainer Type

Attribute	Datatype	Single/ repeating	Description
current_phase_id	ID	S	Object ID of the rps phase rel object representing the phase with which this rps retainer is associated
entry_date	Date	S	Date on which the current phase was entered
event_date	Date	S	The fulfillment date furthest in the future for the current phase's conditions.

Attribute	Datatype	Single/ repeating	Description
immutability_rule	integer	S	Whether objects under control of this retention policy are immutable. Valid values are: 0, the object is changeable 1, the object is immutable This value is copied from the dm_retainer object.
object_name	string(255)	S	The object ID of the retainer. (An rps retainer's name is its object ID.)
parent_ancestor_id	ID	S	Reserved for future use
phase_name	string(32)	S	Name of the phase identified in current_phase_id
qualification_date	Date	S	Date on which objects in the current phase may be qualified for promotion
r_policy_id	ID	S	Object ID of the lifecycle applied to this rps retainer.
retainer_root_id	ID	S	Object ID of the object attached to this rps retainer object. This is blank if the child strategy for the retainer is ONE_FOR_ALL. This is an inherited attribute.
retention_base_date	Date	S	Date from which all qualifications, promotions, and dispositions are calculated This is as required attribute.

Attribute	Datatype	Single/ repeating	Description
retention_date	Date	S	Earliest time at which the object attached to this retainer ceases to be retained. The date is calculated by adding sum of all phase durations to the value in retention_base_date.
retention_policy_id	ID	S	Object ID of the retainer object on which this rps retainer is based.
retention_rule_type	integer	S	Inherited from dm_retainer. This value must be 1, meaning retention is specified as a date.
ultimate_ancestor_id	ID	S	Object ID of the retainer from which all other rps retainer objects were generated. This value is applicable when the child strategy is ONE_FOR_EACH.
vdm_retention_rule	integer	S	Defines how virtual documents and their elements are retained. Valid values are: 0, meaning the policy may not be applied to virtual documents 1, meaning apply the policy only to the root document 2, meaning apply the policy to all virtual document components The value is copied from the retention policy when the retainer is created.

RPS Retainer Event Rel

Purpose Associates rps event objects with rps retainer objects.

Implementation

Supertype: State Extension
 Subtypes: None
 Internal name: dmc_rps_retainer_event_rel
 Object type tag: 37

An rps retainer event rel object associates rps event objects with a rps retainer object. The name of the relationship defined by rps retainer event rel objects is dmc_rps_retainer_event_rel_type.



Caution: This object type is installed with the Retention Policy Services DocApp, and instances of the type are created and maintained through the Retention Policy Services product. Neither the object type nor instances of the type can be changed or modified by users or client applications other than RPS.

[Table 2-149, page 419](#), lists the attributes defined for the type.

Table 2-149. Attributes defined for the RPS retainer event rel type

Attribute	Datatype	Single/ repeating	Description
phase_id	ID	S	Object ID of the rps phase rel object to which this relationship applies
state_no	integer	S	Identifies the phase to which the event applies

RPS Retention Policy

Purpose Defines a rule set from which rps retainer objects are generated.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dmc_rps_retention_policy
 Object type tag: 08

An rps retention policy object defines the rules by which rps retainer objects are generated from the dm_retainer objects. The rules represent the influence of the retention lifecycle definition on the retention policy.



Caution: This object type is installed with the Retention Policy Services DocApp, and instances of the type are created and maintained through the Retention Policy Services product. Neither the object type nor instances of the type can be changed or modified by users or client applications other than RPS.

Table 2–150, page 420, lists the attributes defined for the type.

Table 2-150. Attributes defined for the RPS retention policy type

Attribute	Datatype	Single/ repeating	Description
aging_method	string(32)	S	<p>Determines whether the promotion out of phases is based on conditions or dates. Valid values are:</p> <p>CONDITIONAL, meaning that fulfillment of conditions is used to advance through the phases</p> <p>CHRONOLOGICAL, meaning that dates are used to advance through the phases.</p> <p>This must be set to CONDITIONAL to define conditions for phases.</p>

Attribute	Datatype	Single/ repeating	Description
child_strategy_id	ID	S	Object ID of the rps child strategy object that defines the Java class invoked to apply retention.
child_strategy_class	string(128)	S	Java class that executes based on the selected child strategy. The value is a fully specified class name.
disposition_date	Date	S	Date on which objects associated with retention policy are eligible for disposition
disposition_method_class	string(128)	S	Java class that executes based on the selected disposition method. The value is a fully specified class name.
disposition_method_id	ID	S	Object ID of the rps disposition method object that defines how to dispose of objects controlled by this retention policy.
disposition_rule	integer	S	Defines how RPS disposes of an object. Valid values are: 0, meaning DESTROY CHILDREN 1, meaning DESTROY_VDM 2, meaning DESTROY_ROOT
immutability_rule	integer	S	Whether objects controlled by rps retainers based on this policy are set to immutable. Valid values are: 0, do not set the objects to immutable 1, set the objects immutable

Attribute	Datatype	Single/ repeating	Description
is_enabled	Boolean	S	Whether the policy may be applied to objects.
object_name	string(255)	S	Name of the retention policy. This value is derived from the dm_policy on which the retention policy is based.
rendition_rule	integer	S	Whether the retention policy applies to all content (primary and renditions) of the object or only to primary content. Valid values are: 0, applies to all content 1, applies only to primary content
retainer_ lifecycle_id	ID	S	Object ID of the lifecycle definition that is applied to all rps retainers generated from this retention policy object.
title	string(400)	S	User-defined description of the retention policy.
vdm_retention_ rule	integer	S	Defines how virtual documents and their elements are retained. Valid values are: 0, meaning the retention policy may not be applied to a virtual document 1, meaning the policy is applied only to the root document 2, meaning the policy is applied to all the virtual document components

Scope Config

Purpose Defines the context for the referenced display config objects.

Implementation

Supertype: None
 Subtypes: None
 Internal name: dm_scope_config
 Object type tag: 6c

A scope config object defines the context for the referenced display config objects. Each display config object defines a set of attributes and display hint for the attributes. Each scope config object identifies one or more display config objects and one or more contexts in which the display config is used. Both scope config and display config objects are used by client applications. Content Server does not use these objects.

[Table 2-151, page 423](#), lists the attributes defined for the type.

Table 2-151. Attributes defined for the scope config type

Attribute	Datatype	Single/ repeating	Description
display_config	ID	R	Object IDs of the display config objects used in the scopes defined in the scope config object.
scope_class	string(32)	R	The category of the scope. The only valid value is application.
scope_value	string(32)	R	The specific scope within the scope class. For example, if scope_class is application, scope_value might be Webtop or a DocApp name.
			The scope at a particular index position is the specific instance of the class at the corresponding index position in scope_class.

Script

Purpose Stores a script as content.

Implementation

Supertype: SysObject
Subtypes: None
Internal name: dm_script
Object type tag: 08

A script object stores a script as content. Script objects are used by Documentum clients. Their content files contain scripts of API methods. Depending on where the script object is stored, the script is executed automatically when users connect to a repository or on demand, when users select the script from a menu. Users must have at least Read permission for the script object to execute its associated script.

The script type has no defined attributes. It inherits all of its attributes from its supertype, SysObject.

Sequence

Purpose Used by Content Server to generate object IDs.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dmi_sequence
 Object type tag: 20

A sequence object is used by Content Server to generate object IDs. There is one sequence object for each object type in a repository, up to a maximum of 255.

[Table 2-152, page 425](#), lists the attributes defined for the type.

Table 2-152. Attributes defined for the sequence type

Attribute	Datatype	Single/ repeating	Description
i_current_mask	integer	S	Used internally to manage object IDs
i_high_mask	integer	S	Used internally to manage object IDs
i_high_water _mark	integer	S	The highest number that can be assigned
i_in_use	Boolean	S	Indicates if any of the numbers in the allowed range have been assigned.
i_last_no	integer	S	Contains the last number assigned
i_low_mask	integer	S	Used internally to manage object IDs
i_low_water _mark	integer	S	The first or lowest number that can be assigned

Server Config

Purpose Describes a Content Server.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_server_config
 Object type tag: 3d

A server config object contains information that the server uses to define its operation and operating environment, such as the number of allowed concurrent sessions, maximum cache sizes, and the storage area locations, and the locations of executables that the server calls.

Table 2-153, page 426, lists the attributes defined for the type.

Table 2-153. Attributes defined for the server config type

Attribute	Datatype	Single/ repeating	Description
a_silent_login	Boolean	S	Used internally
agent_launcher	string(32)	S	The name of the method object that launches the agent exec process. The default is agent_exec_method.
alias_set_id	ID	S	The object ID of the alias set object representing the system-level default alias set.
app_server_name	string(32)	R	Name of a Java servlet. The name do_method identifies the servlet to which DO_METHOD functions are directed.

Attribute	Datatype	Single/ repeating	Description
app_server_uri	string(255)	R	<p>URI for the servlet at the corresponding index position in app_server_name. Contains the host, port, and servlet_path in the following format:</p> <p><i>http://host:port/servlet_path</i></p> <p><i>host</i> is the IP address or name of the machine hosting the application server.</p> <p><i>port</i> is the port number on which the application server is listening.</p> <p><i>servlet_path</i> is the path to the servlet to which an HTTP_POST request is directed.</p> <p>The URI can contain only ASCII characters.</p>
application_access_control	Boolean	S	<p>T (TRUE) means users without Superuser privileges must use an application access token to connect to the server.</p> <p>F (FALSE) means any user can connect without an access token.</p> <p>The default is F.</p>
assume_user_location	string(32)	S	<p>Name of the location object for the directory containing the assume user program. The default is assume_user.</p>
auth_plugin_location	string(32)	S	<p>Name of the location object that identifies the directory location of authentication plugins. This is set to auth_plugin by default.</p>
cached_types	string(32)	R	<p>Names of the user-defined object types that you want to cache on server startup. The default is a single blank.</p>

Attribute	Datatype	Single/ repeating	Description
certdb_location	string(32)	S	Name of the location object pointing to the location of the certificate database. The default is ldapcertdb_loc.
change_password_location	string(32)	S	Name of the location object for the directory containing the change password program. The default is change_password.
checkpoint_interval	integer	S	Defines how often the server broadcasts its service information. The default is 300 (seconds).
client_cache_size	integer	S	Maximum allowed size of the client cache, expressed as the number of objects. The default is 50. The number of objects may exceed the specified maximum size if an object must be added to the cache and no candidate for deletion is found. Also, persistently cached objects are not counted towards the maximum. The value in this attribute is the default for all clients that do not have a client cache size explicitly set in the dmcl.ini file.
common_location	string(32)	S	Name of the location object for the common directory. The default is common.
compound_integrity	Boolean	S	Indicates if the server is enforcing referential integrity for virtual documents. The default is TRUE.
concurrent_sessions	integer	S	Number of concurrent sessions. The default is 100.
dba_location	string(32)	S	Name of the location object that identifies the location of the dba directory in the Documentum installation. The default is dm_dba.

Attribute	Datatype	Single/ repeating	Description
default_acl	integer	S	<p>Specifies which ACL the server will use as the default ACL when creating a new object if an ACL is not explicitly associated with the object. Valid values are:</p> <ul style="list-style-type: none"> • 1, to specify the ACL associated with the object's primary folder • 2, to specify the ACL associated with the object's type • 3, to specify the ACL associated with the user who created the object <p>The default is 3.</p>
default_client_codepage	string(64)	S	<p>Identifies the default code page for clients. Legal values are:</p> <p>US-ASCII UTF-8 ISO_8859-1 Shift_JIS EUC-JP EUC-KR ISO-10646-UCS-2</p> <p>The value is determined programmatically and set during Content Server installation.</p>
events_location	string(32)	S	<p>Name of the location object that identifies the events directory. The default is events.</p>
extra_directory_config_id	ID	R	<p>Object IDs of the ldap config objects representing LDAP directories from which users and groups are synchronized.</p>
far_stores	string(32)	R	<p>Names of the storage areas that are considered far for the server. The default is a single blank.</p>

Attribute	Datatype	Single/ repeating	Description
fulltext_location	string(64)	S	Name of the location object that identifies the location of the fulltext configuration file (dmfulltext.ini).
ignore_pre_processing	Boolean	S	Not currently used
keep_entry_interval	integer	S	Defines how long a connection broker keeps a server entry in the absence of a checkpoint message from the server. The default is 1440 minutes (24 hours).
ldap_config_id	ID	S	Object ID of the ldap config object representing the default LDAP directory used by Content Server.
locale_name	string(32)	S	Content Server's locale. Legal values are: en, for English es, for Spanish fr, for French de, for German it, for Italian ja, for Japanese ko, for Korean The value is determined programmatically and set during Content Server installation.
log_location	string(32)	S	Name of the location object for the logs directory. The default is log.
login_ticket_timeout	integer	S	Defines the length of time for which a login ticket is valid. The value is interpreted in minutes. The minimum accepted value is 1 minute. The default is 5 minutes.

Attribute	Datatype	Single/ repeating	Description
mail_method	string(32)	S	<p>Name of the method that Content Server uses to send email messages.</p> <p>Valid values are dm_event_sender, an empty string, or dm_event_template_sender. The default value is dm_event_sender.</p> <p>“dm_event_sender_template” is a valid value only if you have Business Process Designer installed and are using the workflow email template feature.</p> <p>Note: Setting this to an empty string does not disable server email. If you want to disable server email, you must set the mail_notification key in the server.ini file.</p>
max_login_ticket_timeout	integer	S	<p>Maximum length of time, in minutes, that a login ticket can remain valid.</p> <p>The minimum value is 1 minute. The default value is 43200 minutes (30 days).</p>
nfs_enabled	Boolean	S	<p>Indicates if the server is using NFS for file sharing. The default is FALSE.</p>
nls_location	string(32)	S	<p>Name of the location object for the nls directory. The default is a single blank.</p>
object_name	string(255)	S	<p>Identifies the server config object. The default is the name of the repository for which the server was started.</p> <p>Server config names for a Content Server at a primary site is <=32 characters in length.</p> <p>Server config names for content-file servers (remote Content Servers) are 25 characters or less.</p> <p>Server config names are created automatically when the server is installed.</p>

Attribute	Datatype	Single/ repeating	Description
operator_name	string(32)	S	Identifies the operator for the repository (for archiving and restore operations). The default is the value in the server config object's owner_name attribute.
projection_enable	Boolean	R	Indicates whether projection to the connection broker specified at the corresponding index position in projection_targets is enabled.
projection_netloc_enable	Boolean	R	Indicates whether projection to the network location specified at the corresponding index position in projection_netloc_id is enabled.
projection_netloc_id	string(80)	R	User-defined identifiers of network locations.
projection_notes	string(80)	R	User-defined
projection_ports	integer	R	Identifies the port on which the connection broker is listening. The value at each index level is matched to the connection broker specified at the corresponding level in projection_targets. The default is 1489.
projection_proxval	integer	R	Proximity values projected to the connection brokers, the network locations, or both. The value at each index position is projected to the connection broker specified at the corresponding position in projection_targets if projection_enable is TRUE in the corresponding index position. Similarly, the value at each index position is projected to the network location specified at the corresponding index position in projection_netloc_id if projection_netloc_enable is TRUE in the corresponding index position.

Attribute	Datatype	Single/ repeating	Description
projection_targets	string(80)	R	Names of the host machines on which the connection brokers reside.
r_castore_enabled	Boolean	S	Whether the server was installed with a Content Services for EMC Centera license. Valid values are: 1, meaning the server was installed with a Content Services for EMC Centera license 2, meaning the server was not installed with a Centera license.
r_host_name	string(64)	S	Name of the host machine on which the server resides.
r_install_domain	string(16)	S	Name of the Windows domain
r_install_owner	string(32)	S	Name of the installation owner
r_process_id	integer	S	Contains the process id of the main server thread (parent server).
r_server_version	string(32)	S	Version number of the server.
r_trusted_mode	integer	S	Whether the server is a trusted server. 0 means that the server is not a trusted server. 1 means the server is a trusted server. This attribute is set during server installation, depending on whether the user provides a trusted server license key.
rend_backing_store	string(32)	S	Name of the file store object representing the file store storage area where the server will store renditions generated by full-text indexing operations.

Attribute	Datatype	Single/ repeating	Description
restrict_su_ticket_login	Boolean	S	<p>T (TRUE) means a user with Superuser privileges cannot connect to the server using a global login ticket.</p> <p>F (FALSE) allows users with Superuser privileges to connect to the server using a global login ticket.</p> <p>The default is F.</p>
rightsite_image	string(255)	S	<p>Used to construct the URL when a user is sending a document to a mail recipient. Valid values are:</p> <ul style="list-style-type: none"> • webtop • a RightSite imageRightSite images are found in the Windows registry. An example of a Rightsite image is /rs-bin/RightSite.dll <p>The default is webtop.</p>
secure_connect_mode	string(16)	S	<p>The type of port on which the server is listening. Valid values are:</p> <p>native, meaning an unsecure port</p> <p>secure, meaning a secure port (using SSL)</p> <p>dual, meaning both a native and a secure port</p> <p>Non-trusted servers do not use this attribute. They accept only native connections.</p>
secure_writer_location	string(32)	S	<p>Name of the location object for the directory containing the secure writer program. The default is secure_common_area_writer.</p>
server_cache_size	integer	S	<p>Maximum allowed size, expressed as the number of objects. The default is 200.</p>

Attribute	Datatype	Single/ repeating	Description
server_os_ codepage	string(64)	S	<p>The code page used by the operating system of the Content Server host. Legal values vary by operating system. Valid options are:</p> <p>US-ASCII UTF-8 ISO_8859-1 Shift_JIS EUC-JP EUC-KR ISO-10646-UCS-2</p> <p>The value is determined programmatically and set during Content Server installation.</p>
sibling_ checkpoint _interval	integer	S	Defines how often the server broadcasts sibling information (Not currently used)
sibling_export _enabled	Boolean	S	Indicates if the server can broadcast information about its sibling servers. (Not currently used)
signature_chk_ loc	string(32)	S	Name of the location object for the directory containing the signature validation program. The default is validate_signature.
smtp_server	string(64)	S	<p>Name of the SMTP server host.</p> <p>This is used only on Windows platforms, by the email notification system.</p>
stage_destroyer _location	string(32)	S	Not currently used
system_ converter _location	string(32)	S	Name of the location object for the directory containing the convert.tbl file and the system-supplied transformation scripts. The default is convert.

Attribute	Datatype	Single/ repeating	Description
temp_location	string(32)	S	Name of the location object for the temp directory. The default is temp.
turbo_backing_store	string(32)	S	Contains the name of the file store object for the file store storage area where the server puts renditions generated by indexing blob and turbo content. The default is filestore_01.
user_converter_location	string(32)	S	Name of the location object that identifies the directory containing the user-defined transformation scripts. The default is a single blank.
user_validation_location	string(32)	S	Name of the location object for the directory containing the user validation program. The default is validate_user.
verity_location	string(32)	S	Obsolete
web_server_loc	string(255)	S	Name of the Web server's host machine, and optionally, its domain and protocol. Examples: <pre>cinderella cinderella. mycompany.com http:// cinderella.mycompany.com</pre> <p>The default protocol is http.</p>
web_server_port	integer	S	Identifies the port which the Web server is using. The default value is 80.
wf_agent_worker_threads	integer	S	Number of workflow agent worker sessions. The maximum value is 1000. The default value is 3. Setting this to 0 disables the workflow agent.
wf_sleep_interval	integer	S	Length of time, in seconds, that the master workflow agent master session waits before querying the repository for activities awaiting execution in the absence of a notification of an activity waiting for execution. The default is 5 seconds. There is no maximum value.

Server Locator

Purpose Contains information about the servers known to a connection broker.

Implementation

The server locator type is a non-persistent type

A server locator object contains information about the servers known to a connection broker. A server locator object is constructed and returned by the connection broker in response to a `Getservermap` method call.

Table 2-154, page 437, lists the attributes defined for the type.

Table 2-154. Attributes defined for the server locator type

Attribute	Datatype	Single/ repeating	Description
<code>a_silent_login</code>	Boolean	S	Used internally by the Documentum clients.
<code>i_connection_protocol</code>	string(12)	R	Used internally by the dmcl to establish a network connection.
<code>i_docbase_id</code>	string	S	Decimal value of the DocBase identifier assigned during installation.
<code>i_docbroker_version</code>	string(32)	S	Version number of the responding connection broker.
<code>i_host_addr</code>	string(32)	S	IP address of the host machine on which the responding connection broker resides.
<code>i_host_name</code>	string(128)	S	Name of the host machine on which the responding connection broker resides.

Attribute	Datatype	Single/ repeating	Description
i_port_number	integer	S	Port number of the port on the host machine that the connection broker is using for communications.
i_server _connection _address	string(40)	R	Used internally by the dmcl to establish a network connection.
i_logon_support	string(64)	S	Type of logon (Will be NT_Unified_logon for servers running under Windows)
r_client_proximity	integer	R	Indicates how far the server is from the client. The value rises directly in relation to the distance.
r_host_name	string(128)	R	Name of the host machine on which the server resides.
r_keep_entry _interval	integer	R	How long the connection broker will keep the server's entry in the absence of a checkpoint message from the server.
r_last_checkpoint	TIME	R	The time that the server last reported to the connection broker.
r_last_status	string(24)	R	Status of the server process. The value is one of: starting, open, stopped, or presumed down.
r_next_checkpoint	TIME	R	The time that the server is next expected to report to the connection broker.
r_object_id	string	S	Object ID of the server locator object.
r_process_id	integer	R	Process ID of the server.

Attribute	Datatype	Single/ repeating	Description
r_server_name	string(128)	R	Name of the server.
r_server_version	string(32)	R	Version number of the server.

Session

Purpose Not currently used.

Implementation

Supertype: Persistent Object

Subtypes: None

Internal name: dmi_session

Object type tag: 01

The session object type is constructed by the server during server start-up. It is not currently used by Content Server, and there are no objects of that type stored in the repository. However, you can execute a describe command through IDQL to view its attributes, and it is visible as a dm_type object.

Session Config

Purpose Contains information about an open repository session.

Implementation

The session config object type is a non-persistent type.

A session config object contains information about an open repository session. Each repository session has one associated session config object. Access to this object is through the Get and Set methods, using the object's alias, sessionconfig.

[Table 2-155, page 441](#), lists the attributes defined for the type.

Table 2-155. Attributes defined for the session config type

Attribute	Datatype	Single/ repeating	Description
alias_set	string(32)	S	The session-level default alias set.
api_exec_count	integer	S	Number of dmAPIExec function calls issued during the session.
api_get_count	integer	S	Number of dmAPIGet function calls issued during the session.
api_set_count	integer	S	Number of dmAPISet function calls issued during the session.
application_code	string(32)	R	Identifies, by application, the application-controlled objects that this session can modify. If this is NULL, the session cannot modify any application-controlled objects. Application codes can contain only alphanumeric characters and the underscore character.

Attribute	Datatype	Single/ repeating	Description
			They cannot contain spaces nor can they start with the characters dm_. Codes beginning with dm_ are reserved for use by Documentum.
batch_hint_size	integer	S	Defines the number of rows returned to Content Server by the RDBMS in a single call to the RDBMS. The default is 20.
client_cache_size	integer	S	Defines the size of the client cache for the session. The value is derived from the setting in the connection config object for the first repository connection of the session.
connect_callback_enabled	Boolean	S	T or F, indicating whether the server calls the content callback functions. The default is T.
connect_failure_callback	integer	R	Contains the memory addresses of user-defined applications to capture and handle login failures.
connect_failure_data	integer	R	Contains the memory addresses of data to be passed to user-defined applications that handle login failures.
connect_success_callback	integer	R	Contains the memory addresses of user-defined applications that clear status messages resulting from execution of applications identified in new_connection_callback.

Attribute	Datatype	Single/ repeating	Description
connect_success_data	integer	R	Contains the memory addresses of data to be passed to applications identified in connect_success_callback.
content_callback_data	integer	R	Contains a memory address that points to the argument values for the content callback function.
content_callback_function	integer	R	Contains the memory address of a content callback function.
docbase_scope	string(120)	S	Defines the default repository for operations executed during the session. Valid values are a repository name, an object ID, or an indirect reference (<i>@object_id</i>). All operations not specifically directed to a different repository are executed against the default repository.
dynamic_groups	string(32)	R	List of the dynamic groups to which the session belongs.
ignore_pre_processing	Boolean	S	Used for customized workflows. It directs the server not to execute external applications defined as pre-processing. The default is FALSE.

Attribute	Datatype	Single/ repeating	Description
ignore_post_processing	Boolean	S	Used for customized workflows. It directs the server not to execute external applications defined as post-processing. The default is FALSE.
local_clean_on_init	Boolean	S	Indicates whether the server should automatically purge the client local area whenever the client is started. The default is T.
local_diskfull_check	integer	S	Defines how often the server checks the free space on the disk containing the client's local area. The default is 1, meaning that the space is checked with each Getfile. Setting this to a number greater than 1, n, means that the space is checked every n Getfiles. Setting it to 0 means the space is never checked.
local_diskfull_limit	integer	S	Specifies how full you want to allow the disk containing the client local area to become. Valid values are from 1 and 100. The value represents a percentage of the total possible space on the disk.
local_diskfull_warn	integer	S	Specifies a percentage value of the total possible space on the disk. If copying a file into the client local area fills the disk beyond the specified percentage, a warning is sent to the client. Valid values are from 1 to 100.

Attribute	Datatype	Single/ repeating	Description
local_purge_on _diskfull	Boolean	S	Indicates whether the server should automatically purge the client local area when the diskfull limit is reached. The default value is T.
local_path	string(255)	S	Specifies the location of the local client area. Use a full path specification.
network_callback_ data	integer	R	Contains the memory address of the argument values for the network callback function.
network_callback_ _function	integer	R	Contains the memory address of the network callback function.
network_enabled	Boolean	S	T or F, indicating whether the server calls the network callback functions. The default is T.
network_requests	integer	S	Number of RPC calls made during the session. This is a cumulative number, incrementing by one each time an RPC call is made in the session.
new_connection_ _callback	integer	R	Contains the memory address of user-defined applications that display status messages for new repository connections prior to establishment of the connection.
new_connection_ _data	integer	R	Contains the memory address of data used by applications identified in new_connection_callback.

Attribute	Datatype	Single/ repeating	Description
nfs_enabled	Boolean	S	Indicates whether the session will use NFS as its file-sharing protocol. The value is derived from the setting in the connection config object for the first repository connection of the session.
r_cache_query	Boolean	S	Indicates whether the session is enabled for query caching. The default value is FALSE.
r_date_format	string(40)	S	The date format the server will use to return dates to the client. The value is derived from the setting in the connectionconfig object for the first repository connection of the session.
r_events_location	string(32)	S	Identifies the location object in the repository representing the events directory. The value is derived from the setting in the connection config object for the first repository connection of the session.
r_host_name	string(128)	S	Contains the host name of the machine on which the current session was initiated.
r_mac_protocol	string(32)	S	Specifies the file-sharing protocol if this is a Macintosh client session. The value is derived from the setting in the connection config object for the first repository connection of the session.

Attribute	Datatype	Single/ repeating	Description
r_security_mode	string(32)	S	Contains the security level enforced for the repository you are using. The value is derived from the setting in the connection config object for the first repository connection of the session.
r_user_name	string(32)	S	Contains the user name of the currently logged-in user.
r_working_directory	string(255)	S	Contains the path of the user's working directory.
ref_binding_label	string(32)	S	Defines which document version to fetch when accessing a remote document. If this attribute is defined, its value overrides the binding specified in the dm_reference object that points to the remote object.
session_alias	string(12)	S	Contains the session identifier, as a string literal.
session_codepage	string(64)	S	The code page in use for the repository session. By default, the value is obtained from the client_codepage attribute of the api config object. If client_codepage is unset, the value is determined programmatically based on the value in session_locale. For all session locales except ja (Japanese) and ko (Korean), the default setting is ISO-8859-1. For Japanese, the default setting is Shift_JIS.

Attribute	Datatype	Single/ repeating	Description
session_id	ID	S	For Korean, the default setting is EUC-KR. Contains the ID of the session.
session_locale	string(32)	S	Locale of the repository session. By default, the value is obtained from the client_locale attribute of the api config object. If client_locale is unset, the setting is determined programmatically based on the locale of the client's host machine. Valid values are: en, for English de, for German fr, for French ja, for Japanese ko, for Korean
use_local_always	Boolean	S	Indicates whether the session will use a local common area.
use_local_on_copy	Boolean	S	Used in conjunction with the use_local_always and nfs_enabled (api config) attributes to determine whether the server copies requested files to the client local area. The default is F.

Smart List

Purpose Represents the set of objects determined by the SELECT query defined for the Smart List.

Implementation

Supertype: SysObject
Subtypes: None
Internal name: dm_smart_list
Object type tag: 08

A Smart List object represents the set of objects determined by the SELECT query defined for the Smart List. A Smart List object can only be created using Desktop Client or Webtop. When a user opens a Smart List, the query is executed and the retrieved items are displayed to the user in a window that allows the users to manipulate the objects.

If the Smart List is created from Desktop Client, the query is saved as DQL. If the Smart List is created from Webtop, the query is saved as an XML file. Smart Lists created in Desktop Client can be read by Webtop. However, Smart Lists created in Webtop cannot be read by Desktop Client.

The Smart List type has no attributes defined for it. It inherits all its attributes from its supertype.

State Extension

Purpose Identifies the lifecycle state with which a state extension object is associated.

Implementation

Supertype: Relation

Subtypes: RPS Phase Rel, RPS Retainer Event Rel

Internal name: dm_state_extension

Object type tag: 37

A state extension object identifies the lifecycle state with which the state extension object is associated. The state extension object type is the base object type for state extension object subtypes. Typically, the state extension object type is subtyped to add attributes to store the application-specific information needed for particular lifecycle states.

[Table 2-156, page 450](#), lists the attribute defined for the type.

Table 2-156. Attributes defined for the state extension type

Attribute	Datatype	Single/ repeating	Description
state_no	integer	S	Number of the state with which the state extension object is associated. The number is the value in the dm_policy object's state_no attribute.

State Type

Purpose Identifies a particular lifecycle state type name and application pair.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dm_state_type
 Object type tag: 00

A state type object identifies a particular lifecycle state type name and application pair. State type objects can be used to associate a state type name with a particular application. Some Documentum client applications require specific state type names be assigned to lifecycle states. User applications may also make use of state types. (For more information, refer to *Content Server Fundamentals*.)

[Table 2-157, page 451](#), lists the attributes defined for the type.

Table 2-157. Attributes defined for the state type object type

Attribute	Datatype	Single/ repeating	Description
application_code	string(32)	S	Name of the application that recognizes the state type.
state_type_name	string(32)	S	Name of a lifecycle state type.

SSA Policy

Purpose Defines a content assignment policy for new content files.

Implementation

Supertype: SysObject
Subtypes: None
Internal name: dm_ssa_policy
Object type tag: 08

An ssa policy object defines storage rules for content files. The rules are stored as content of the object. A policy is associated with one or more object types through a relationship, recorded in an relation ssa policy object.

SSA policy objects are created using Documentum Administrator. Existing policies can be copied or modified, but not versioned. You must have installed Content Server with a Content Storage Services license to create and use assignment policies.

[Table 2-158, page 452](#), lists the attributes defined for the type.

Table 2-158. Attributes defined for the SSA policy type

Attribute	Datatype	Single/ repeating	Description
is_activated	Boolean	S	Whether the policy is active (being enforced by the policy engine). T (TRUE) means the policy is enforced. F (FALSE) means the policy is inactive. The default is T.

Store

Purpose Represents a content storage area.

Implementation

Supertype: Persistent Object

Subtypes: Distributed Store, File Store, Linked Store, Blob Store, External Store, CA Store

Internal name: dm_store

Object type tag: 0e

A store object represents a content storage area. The attributes in store objects may be changed only by users with Sysadmin or Superuser user privileges.

[Table 2-159, page 453](#), lists the attributes defined for the type.

Table 2-159. Attributes defined for the store type

Attribute	Datatype	Single/ repeating	Description
base_url	string(255)	S	The basic URL used to retrieve contents directly from a storage area.
capacity	integer	S	Not currently used
component	string(64)	R	The name of the storage objects representing component storage areas. (Used by linked store and distributed stores.)
compression_mode	integer	S	Applicable only to file store and ca store storage areas and if the server is installed with a Content Storage Services license, this attribute indicates whether content written to the storage area is compressed. Valid values are 0, meaning the content in the storage area is not compressed, and

Attribute	Datatype	Single/ repeating	Description
			1, meaning the content is compressed.
			If not explicitly set when the storage area is created, the default value is 0.
			The setting may not be changed after the storage area is created.
content_dupl_pref	integer	S	Used only if the server is installed with a Content Storage Services license, this attribute indicates the content duplication preference for the storage area. Valid values are: 0, meaning Content Server does not check the storage area for duplicate content files when saving content files to the storage area 1, meaning Content Server checks the storage area for duplicate content files when saving content files to the storage area The default is 0. Refer to Content duplication checking and prevention, page 224 in the <i>Content Server Administrator's Guide</i> for information about how this attribute is used.

Attribute	Datatype	Single/ repeating	Description
content_hash_mode	integer	S	<p>Used only if the server is installed with a Content Storage Services license, this attribute configures hash generation for content files in the storage area. This attribute setting is applicable only to file store storage areas. Valid values are:</p> <p>0, meaning do not generate hashes for the content files in the storage area.</p> <p>1, meaning generate a SHA1 hash for each content file in the storage area</p> <p>If content_dupl_pref is set to a non-zero value, the value of content_hash_mode must be 1. Content Server sets the value to 1 automatically under that condition.</p> <p>If content_dupl_pref is 0, the default value for content_hash_mode is 0.</p> <p>Refer to Content duplication checking and prevention, page 224 in the <i>Content Server Administrator's Guide</i> for information about how this attribute is used.</p>

Attribute	Datatype	Single/ repeating	Description
crypto_key	string(256)	S	<p>Contains the symmetric key used to encrypt the content files stored in the storage area.</p> <p>You cannot create indexes on this attribute.</p>
crypto_mode	integer	S	<p>Whether the contents of the storage area are encrypted. 0 means the contents are not encrypted. 1 means the contents are encrypted.</p> <p>This value is set when the storage area is created and cannot be changed afterwards. You can only set this value for file store storage areas.</p> <p>The default value is 0.</p>
current_use	integer	S	For internal use
digital_shredding	Boolean	S	<p>Whether digital shredding is enabled for the storage area. Valid values are:</p> <p>0, meaning shredding is disabled</p> <p>1, meaning shredding is enabled</p> <p>The default is 0.</p> <p>This attribute only affects file store storage areas and may only be set to 1 if the server is installed with a Trusted Content Services license.</p>

Attribute	Datatype	Single/ repeating	Description
get_method	string(32)	S	Contains the name of the dm_method object representing the user-defined program that retrieves a specified content.
i_use_mask_after	integer	S	Used internally to manage storage ticket numbers for content files.
media_type	integer	S	Identifies the media type of content stored in this storage area. Valid values are: <ul style="list-style-type: none"> • 0, for any type other than thumbnail or streaming • 1, for thumbnail • 2, for streaming <p>The attribute cannot be changed after the storage area is created.</p>
name	string(64)	S	Contains the name of the store object. This must be a unique name within the repository.
offline_get_method	Boolean	S	Indicates whether the application should regard the retrieved content as immediately available or awaiting restoration (from archiving).
r_component	ID	R	The object IDs of any component storage areas.
r_component_count	integer	S	The total number of component storage areas.

Attribute	Datatype	Single/ repeating	Description
r_status	integer	S	Records the state of the storage area. Valid values are: 0, for on-line 1, for off-line 2, for read-only
require_ticket	Boolean	S	Indicates whether Content Server generates a ticket when returning a URL to a content file. TRUE means a ticket is returned. FALSE means a ticket is not returned. The default is FALSE.
store_type	integer	S	Value indicating the subtype of the storage area. This can be one of the following values: <ul style="list-style-type: none">• 1, for file store• 3, for linked store• 4, for distributed store• 5, for blob store• 7, for external store• 8, for external file store• 9, for external URL store• 10, for external free store• 11, for CA store Note: 2 and 6 are currently unused values.

Subcontent

Purpose Stores a content file in turbo storage or the content addresses of content stored in content-addressed storage systems.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dmi_subcontent
 Object type tag: 64

A subcontent object stores a content file in turbo storage or the content addresses of content stored in content-addressed storage systems. When a content file in turbo storage is too big to store in a content object, the content is stored in subcontent object. If a content file stored in a content-addressed system is modified or if its metadata attributes in the storage system are modified, the storage system creates a new address for the content file. This address is stored in a subcontent object.

Subcontent objects are internal objects that users and applications cannot access directly. However, users and applications can query subcontent objects using DQL.

[Table 2-160, page 459](#), lists the attributes defined for the subcontent object type.

Table 2-160. Attributes defined for the subcontent type

Attribute	Datatype	Single/ repeating	Description
parent_id	string(16)	S	Object ID of the dmr_content object that represents the content file or content address stored in this object.
i_contents	For all databases except Sybase: string(2000) For Sybase: string(255)	R	For objects in turbo storage, this contains the object's content file if the content is too big for i_contents in the content object. This attribute cannot be selected using either the API or DQL when it used as a storage area for turbo content. For objects in content-addressed storage, this contains additional content addresses generated by operations on the object or its

Attribute	Datatype	Single/ repeating	Description
i_contents_size	integer	R	<p>storage metadata values. All addresses point to the same content, however the metadata and retention period information associated with a particular address typically differs.</p> <p>Actual size of the content, including trailing spaces, stored in the corresponding index position of i_contents.</p> <p>Note: Subcontent objects created by server versions prior to 5.0 will have no value in this attribute.</p> <p>This attribute is not used if the subcontent object records a content address.</p>

SysObject

Purpose Serves as the parent type of the most commonly used objects in the Documentum system.

Implementation

Supertype: Persistent Object

Subtypes: Document, Folder, Output Device, Category Class, and so forth

Internal name: dm_sysobject

Object type tag: 08

The SysObject type is the parent type of the most commonly used objects in the Documentum system. The SysObject type has three important properties, represented by attributes, that it passes on to all its subtypes. These properties are:

- A SysObject accepts security permissions. Attributes defined for the SysObject allow you to set permissions on the object.
- A SysObject, unless it is a cabinet, can belong to a folder.
- A SysObject can own one or more content objects.

Tables [Table 2–161, page 462](#) through [Table 2–169, page 474](#) list the attributes defined for the type. The tables group the SysObject attributes by use or function:

- General attributes, described in [Table 2–161, page 462](#)
- Folder-related attributes, described in [Table 2–162, page 464](#)
- Virtual document-related attributes, described in [Table 2–163, page 465](#)
- Content-related attributes, described in [Table 2–164, page 467](#)
- Version-control attributes, described in [Table 2–167, page 472](#)
- Event-related attribute, described in [Table 2–168, page 474](#)
- Lifecycle-related attributes, described in [Table 2–169, page 474](#)

General attributes

The general attributes provide a general description of the object.

Table 2-161. General attributes of SysObjects

Attribute	Datatype	Single/ repeating	Description
a_application_type	string(32)	S	Currently unused.
a_archive	Boolean	S	Used internally.
a_extended _properties	string(32)	R	Individual values in this attribute are either: <ul style="list-style-type: none"> • The value of the relation_name attribute in a dm_relation object or subtype of a dm_relation. This is used in conjunction with the r_object_id of the sysobject to identify the relation object that contains the SysObject's extended properties. • Room ID
a_is_hidden	Boolean	S	Indicates if this object is visible to end users
a_is_template	Boolean	S	Indicates whether the object is a template
a_last_review_date	Date	S	Date of the last review of the object. This value is set automatically by some Documentum client products.
a_retention_date	date	S	Used internally.
a_special_app	string(32)	S	Reserved for use by Documentum products.
a_status	string(16)	S	Not currently used.
authors	string(48)	R	List of the authors for the object. This is user-defined.

Attribute	Datatype	Single/ repeating	Description
i_is_deleted	Boolean	S	If the object is the root version of a version tree, deleting the object sets this attribute to TRUE.
i_is_reference	Boolean	S	Indicates whether the object is a mirror object for a reference link to a remote object. TRUE means that the object is a mirror object.
i_retainer_id	ID	R	Object IDs of the retainer representing the retention policy that controls the object.
keywords	string(48)	R	List of user-defined keywords for the object
language_code	string(5)	S	Five-character code, in the format <i>xx_yy</i> , indicating the language in which the document is written and the country of origin. The first two characters (<i>xx</i>) contain the language code and the final two (<i>yy</i>), the country code. Appendix B, Language and Country Codes contains recommended language and country codes.
object_name	string(255)	S	Name of the object. This is user-defined. Note: If you are using a Documentum client, this is the name that appears under the object's icon.
r_access_date	date	S	Contains the date and time when this object was last accessed by a Getfile or Print method

Attribute	Datatype	Single/ repeating	Description
r_alias_set_id	ID	S	Object ID of the alias set associated with the sysobject.
r_aspect_name	string(64)	R	Reserved for internal use Manipulation of this attribute by users or external applications is not supported.
r_creation_date	date	S	Date the object was created. This is set by the server.
r_modifier	string(32)	S	Name of the user who made the last modification.
r_modify_date	time	S	Date the object was last modified. This is set by the server.
r_object_type	string(32)	S	The object's type, for example, dm_sysobject or dm_document, set when the object is created.
subject	string(192)	S	Subject of the object. This is user-defined.
title	string(400)	S	Title of the object. This is user-defined.

Folder-related attributes

The folder attributes provide information for managing folders.

Table 2-162. SysObject folder-related attributes

Attribute	Datatype	Single/ repeating	Description
governing_room_id	ID	S	Object ID of the room that governs this object.
i_folder_id	ID	R	Object IDs of all folders linked to the object.

Attribute	Datatype	Single/ repeating	Description
i_has_folder	Boolean	S	Flag indicating whether this object is the CURRENT object in the version tree. T means the object has the CURRENT version label. F means it does not.
r_link_cnt	integer	S	Number of objects linked to the folder. Note: This attribute has two uses in the SysObject object type. One, described here, for folders and one, described in Table 2-163, page 465 , for virtual documents.
r_order_no	integer	R	No longer used. Preserved for backwards compatibility.
i_reference_cnt	integer	S	Number of folder references made to this object.
r_component_label	string(32)	R	No longer used. Preserved for backwards compatibility.
r_composite_id	ID	R	No longer used. Preserved for backwards compatibility.
r_composite_label	string(32)	R	No longer used. Preserved for backwards compatibility.

Virtual document-related attributes

These attributes provide information for the management of virtual documents.

Table 2-163. Virtual document-related attributes

Attribute	Datatype	Single/ repeating	Description
a_compound_architecture	string(16)	S	Used by Virtual Document Manager to determine if the object can be structurally changed through VDM. An empty string indicates that the object can be changed.
a_link_resolved	Boolean	S	Used internally.

Attribute	Datatype	Single/ repeating	Description
resolution_label	string(24)	S	Contains the default version label used to resolve late-bound nodes of a virtual document.
r_assembled_from_id	ID	S	Object ID of the virtual document that was the source of the assembly associated with a document.
r_frzn_assembly_cnt	integer	S	Contains a count of the number of frozen assemblies that contain this object.
r_has_frzn_assembly	Boolean	S	Indicates that the document's assembly is frozen.
r_is_virtual_doc	integer	S	Indicates whether the SysObject is a virtual document. If the value is 1, the object is a virtual document. If the value is 0, the object is not a virtual document unless the r_link_cnt is greater than 0.
r_link_cnt	integer	S	Number of components in the virtual document.
r_link_high_cnt	integer	S	Records the current maximum order number assigned to a component.

Content-related attributes

The content-related attributes provide information about the content of sysobjects.

Table 2-164. SysObject content-related attributes

Attribute	Datatype	Single/ repeating	Description
a_content_type	string(32)	S	File format of the object's content. Note: For some system-created renditions, Content Server sets this value to dm_internal.
a_full_text	Boolean	S	Indicates whether the document is marked for full-text indexing.
a_is_signed	Boolean	S	Indicates whether the object has been digitally signed. The attribute is set to T by the client application after the user signs the object.
a_storage_type	string(32)	S	Identifies the storage area for content files associated with the object. This is the name of the storage object representing the storage area.
i_contents_id	ID	S	Object ID of the content object for an object that has only one content. This is not used if an object has multiple content. In such cases, the Content Facility manages the relationships between the object and its multiple content.

Attribute	Datatype	Single/ repeating	Description
r_content_size	integer	S	Size, in bytes, of the first content file associated with the document. This attribute cannot record content sizes greater than 2GB. Examine r_full_content_size to obtain the size of content larger than 2GB.
i_retain_until	date	S	Most restrictive retention date among all content associated with this object. This attribute is only set if the content has a retention period. Retention periods are set by a retention policy or through storage in a content-addressed storage area with a retention period.
r_full_content_size	double	S	Size, in bytes, of the first content file added to the SysObject.
r_page_cnt	integer	S	Number of content files associated with the object.

Web-content attributes

The Web-content attributes are used by the Documentum applications that manage SysObjects that are published on Web sites. These attributes are not intended for use by user applications.

Table 2-165. Web-content attributes

Attribute	Datatype	Single/ repeating	Description
a_category	string(64)	S	Used internally to manage the application.

Attribute	Datatype	Single/ repeating	Description
a_effective_date	date	R	The date on which the document can be published to the Web site. The value at a particular index position applies to the Web site named at the corresponding index position in a_effective_label.
a_effective_flag	Boolean	R	Indicates whether a pending expiration notice was sent to the document's owner. The value at a particular index position applies to the Web site named at the corresponding index position in a_effective_label.
a_effective_label	string(32)	R	User-defined, typically a symbolic label. The WebCache export operation examines the values in the a_effective_date and a_expiration_date attributes of all documents whose a_effective_label attribute value matches the webc config effective_label attribute. If unspecified, effective labels are not enforced.

Attribute	Datatype	Single/ repeating	Description
a_expiration_date	date	R	The date at which the document is to be removed from the Web site. The value at a particular index position applies to the Web site named at the corresponding index position in a_effective_label.
a_publish_formats	string(32)	R	Lists the object's renditions that are to be exported to the WebCache repository. The system will export these renditions plus those defined in the source_formats attribute of the webc config object. This is an optional attribute. If unspecified, only renditions defined in the webc config object are published.

Security-related attributes

The security-related attributes provide information that allows the server to enforce security on the object. Changes to security-related attributes may create a new ACL for the object.

Table 2-166. SysObject security-related attributes

Attribute	Datatype	Single repeating	Description
a_controlling_app	string(32)	S	Identifies the application or applications that can modify this object. If NULL, then any application can modify the object.

Attribute	Datatype	Single repeating	Description
acl_domain	string(32)	S	Identifies the domain of the ACL associated with the object. The value will be either the user who created the ACL or, for system-level ACLs, the name of the repository owner.
acl_name	string(32)	S	The object name of the associated ACL.
group_name	string(32)	S	Group to which this object belongs.
group_permit	integer	S	Object-level permission assigned to the object's group for this object.
i_cabinet_id	ID	S	Object ID of the cabinet that is the object's primary storage location.
owner_name	string(32)	S	Name of the object's owner. This can be a user or group name.
owner_permit	integer	S	Object-level permission assigned to the owner for this object.
r_creator_name	string(32)	S	Name of the object's creator.
r_is_public	Boolean	S	TRUE means the object is public. FALSE indicates that the object is not public. If this is TRUE, the object's attributes and content are open to the public—there is no security checking on the object.

Attribute	Datatype	Single repeating	Description
r_lock_machine	string(32)	S	Name of the client machine on which a user is working when he or she locks an object (by a checkout or branch operation). If the server is unable to resolve the name, r_lock_attribute is set to the client machine's IP address instead.
world_permit	integer	S	Object-level permission assigned to the world (all users except owner and group members) for this object.

Version-related attributes

The version-related attributes provide information that lets the server manage and track the versions of an object.

Table 2-167. SysObject version-related attributes

Attribute	Datatype	Single/repeating	Description
i_antecedent_id	ID	S	Object ID of the object's parent version.
i_branch_cnt	integer	S	Number of branches on the version tree that contains the object.
i_chronicle_id	ID	S	Object ID of the root object of the version tree that contains this object.
i_direct_dsc	Boolean	S	Indicates whether the object has any direct descendants (versions derived directly from this object). The default is FALSE.

Attribute	Datatype	Single/ repeating	Description
i_latest_flag	Boolean	S	Indicates whether this version is the most recent version of the object on a particular branch in the version tree.
log_entry	string(120)	S	Comments specified by the user.
r_frozen_flag	Boolean	S	Indicates whether the object is unchangeable because it was specifically frozen. The default is FALSE.
r_immutable_flag	Boolean	S	Indicates whether the object can be changed. The default is FALSE.
r_lock_date	date	S	Date that this object was locked.
r_lock_owner	string(32)	S	Name of the user who locked the object.
r_version_label	string(32)	R	List of the version labels associated with the object. The first position in this attribute holds the object's implicit version label. The remaining rows contain the object's symbolic version labels.

Event-related attributes

The SysObject type has the following event-related attribute.

Table 2-168. SysObject event-related attribute

Attribute	Datatype	Single/ repeating	Description
r_has_events	Boolean	S	Indicates whether any users have registered to receive events for this object. The default is FALSE.

Lifecycle-related attributes

The lifecycle-related attributes provide information about associated lifecycles (policy objects).

Table 2-169. SysObject lifecycle-related attributes

Attribute	Datatype	Single/ repeating	Description
r_policy_id	ID	S	Object ID of the associated policy object
r_current_state	integer	S	State number of the current state of the object in the lifecycle.
r_resume_state	integer	S	The state number to which the object is resumed if r_current_state identifies an exception state.

Taxonomy

Purpose Records the default values for the attributes in the categories of a taxonomy structure in the repository.

Implementation

Supertype: Category
 Subtypes: None
 Internal name: dm_taxonomy
 Object type tag: 0b

A taxonomy object records the default values for the attributes in the categories of a taxonomy structure in the repository.

[Table 2-170, page 475](#), lists the attribute defined for a taxonomy.

Table 2-170. Attributes defined for the taxonomy type

Attribute	Datatype	Single/ repeating	Description
taxonomy_ version	string(32)	S	Current version of the installed taxonomy.

TCF Activity

Purpose Records a sequence of actions to be executed on entry for a lifecycle state.

Implementation

Supertype: Document
Subtypes: None
Internal name: dmc_tcf_activity
Object type tag: 09

A tcf activity object represents a sequence of actions to execute when an object enters a particular state. TCF activity objects are created when a user identifies one or more actions for a state when defining a lifecycle state in Lifecycle Editor. You cannot create these objects manually. The actions and their parameters are recorded in an XML file that is stored as the content of the tcf activity object.

[Table 2-171, page 476](#), lists the attributes defined for the type.

Table 2-171. Attributes defined for the TCF activity type

Attribute	Datatype	Single/ repeating	Description
act_identifier	string(256)	S	Character string identifier that identifies this tcf activity object.
act_template_ identifier	string(256)	S	Used internally

TCF Activity Template

Purpose Represents one lifecycle state action.

Implementation

Supertype: Document
 Subtypes: None
 Internal name: dmc_tcf_activity_template
 Object type tag: 09

A tcf activity template object represents one action that may be executed when an object enters a lifecycle state. A suite of TCF activity template objects is installed with the Lifecycle Editor. The content files associated with the objects are XML files that contain the template's string identifier and the parameters that must be supplied to execute the actions. You cannot create these objects manually.

[Table 2-172, page 477](#), lists the attributes defined for the type.

Table 2-172. Attributes defined for the TCF activity template type

Attribute	Datatype	Single/ repeating	Description
act_template_ identifier	string(256)	S	Character string identifier that identifies this tcf activity template object.
template_ groups	string(32)	S	Reserved for future use

Topic

Purpose Used to manage a single discussion.

Implementation

Supertype: Folder
 Subtypes: None
 Internal name: dmc_topic
 Object type tag: 0b

A topic object is used to manage a discussion thread—the set of comments about a single topic. Discussions are supported in Webtop if you have installed Content Server with a Documentum Collaborative Services license. You cannot create or manage topic objects manually. They are created and managed through Webtop when a discussion is started.

Table 2-173, page 478, lists the attributes defined for the type.

Table 2-173. Attributes defined for the topic type

Attribute	Datatype	Single/ repeating	Description
is_disabled	Boolean	S	Whether the topic is disabled. T means the topic is disabled. F means the topic is not disabled. The default is F. Note: Users cannot create, view or edit comments in disabled topics.
next_comment_id	integer	S	Value to be used as the object ID of the next new comment.
last_update_modtag	integer	S	Value used internally to track each time a topic is modified

Transition Condition

Purpose Records a route case condition expression for an automatic transition of a workflow activity.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dmc_transition_condition
 Object type tag: 00

A transition condition object records a conditional expression in a route case condition. You cannot create these objects directly. They are created when an addConditionRouteCase method (defined for the IDfActivity interface) is executed by a DFC at version level 5.3 FCS. This method is executed by Business Process Manager to save an activity's route case conditions when any one or more of the route cases contains an XPath expression.

Note: When the addConditionRouteCase method is called by a DFC 5.3 SP1 or later, the method creates routecase condition objects, instead of transition condition objects, to record the expression.

[Table 2-174, page 479](#), lists the attributes defined for the type.

Table 2-174. Attributes defined for the transition condition type

Attribute	Datatype	Single/repeating	Description
r_aspect_name	string(64)	R	Used internally
r_attribute_name	string(32)	S	Name of the attribute referenced in the expression, if any
r_boolean_value	Boolean	S	Value to be used in the comparison if the datatype identified in r_value_type is Boolean. This is not set if unless the datatype is Boolean.

Attribute	Datatype	Single/repeating	Description
r_double_value	Double	S	Value to be used in the comparison if the datatype identified in r_value_type is double. This is not set if unless the datatype is double.
r_id_value	ID	S	Value to be used in the comparison if the datatype identified in r_value_type is ID. This is not set if unless the datatype is ID.
r_int_value	integer	S	Value to be used in the comparison if the datatype identified in r_value_type is integer. This is not set if unless the datatype is integer.
r_object_alias	string(32)	S	Name of the package, or manifest values referring to the workflow or work item
r_relational_op	integer	S	The relation operator in the condition. Valid values are: 0, meaning = 1, meaning <> 2, meaning < 3, meaning > 4, meaning <= 5, meaning >=

Attribute	Datatype	Single/repeating	Description
r_repeating_attr_flag	integer	S	<p>Indicates whether the attribute named in r_attribute_name is a repeating attribute and if it is a repeating attribute, which values to examine when evaluating the condition. Valid values are:</p> <p>-1, meaning the attribute is not a repeating attribute</p> <p>0, meaning ANY</p> <p>1, meaning ALL</p> <p>2, meaning FIRST</p> <p>3, meaning LAST</p>
r_string_value	string(1024)	S	<p>Value to be used in the comparison if the datatype identified in r_value_type is string.</p> <p>This is not set if unless the datatype is string.</p>
r_time_value	Date	S	<p>Value to be used in the comparison if the datatype identified in r_value_type is Date.</p> <p>This is not set if unless the datatype is Date.</p>
r_value_type	integer	S	<p>Data type of the value in the relational expression. The data type is expressed as an IDfValue constant.</p>
r_xpath_datatype	string(64)	S	<p>The xschema's built-in datatype name.</p> <p>This is set only if the condition includes an XPath expression.</p>

Attribute	Datatype	Single/repeating	Description
r_xpath_expression	string(1024)	S	An XPath expression. This is set only if the condition includes an XPath expression.
r_xpath_value	string(1024)	S	Literal value used in the XPath transition condition evaluation. The value is in the format in which it is found in the XML document. This is set only if the condition includes an XPath expression.

Type

Purpose Stores structural information about an object type in the repository.

Implementation

Supertype: Persistent Object

Subtypes: None

Internal name: dm_type

Object type tag: 03

A type object stores structural information about an object type in the repository. The object types in Content Server are themselves represented as types (that is, each type is an object of type dm_type). Content Server does not allow you to create new objects of type dm_type directly. Instead, when you create a new type or subtype, Content Server automatically creates an object of type dm_type to describe your new type. (To create a new type, use the DQL CREATE TYPE statement.)

Table 2-175, page 483, lists the attributes defined for the type.

Table 2-175. Attributes defined for the type object type

Attribute	Datatype	Single/ repeating	Description
attr_count	integer	S	Number of attributes in the type (includes defined and inherited)
attr_length	integer	R	Indicates the length of those attributes that are string-valued.
attr_name	string(40)	R	Names of the type's attributes
attr_repeating	Boolean	R	Indicates if the attributes are repeating
attr_type	integer	R	Contains integer values representing the data type of the attributes.
info	ID	S	Object ID of the type's associated type info object

Attribute	Datatype	Single/ repeating	Description
name	string(27)	S	Name of the type
owner	string(40)	S	Name of the type's owner (creator)
r_object_id	ID	S	Object ID of the type
r_index_attr	ID	S	Object ID of the index object describing any indexes built on the type's repeating attributes.
start_pos	integer	S	Position of the first non-inherited attribute
super_name	string(40)	S	Name of the type's supertype
s_index_attr	ID	S	Object ID of the index object describing any indexes built on the type's single-valued attributes.
views_valid	Boolean	S	Indicates if the views for the type are valid. This is used internally during recovery operations.

Type Info

Purpose Sstores non-structural information about an object type.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dmi_type_info
 Object type tag: 2e

A type info object stores non-structural information about an object type. Storing non-structural information separately from the type's structure definition (the dm_type object) enhances the performance when a type is altered.

Table 2-176, page 485, lists the attributes defined for the type.

Table 2-176. Attributes defined for the type info type

Attribute	Datatype	Single/ repeating	Description
acl_domain	string(32)	S	The domain (owner) of the ACL associated with the type definition. The value is either the name of the user who created the ACL or, for system ACLs, the name of the repository owner.
acl_name	string(32)	S	The object name of the associated ACL.
default_group	string(27)	S	Default group defined for the type.
default_group _permit	integer	S	Default object-level permission defined at the group level for the type.
default_owner _permit	integer	S	Default object-level permission defined at the owner level for the type.

Attribute	Datatype	Single/ repeating	Description
default_storage	ID	S	Object ID of a storage object of type dm_store. This identifies the default storage type for the contents associated with any object of this type.
default_world_permit	integer	S	Default object-level permission defined at the world level for the type.
ftindex_attrs	string(27)	R	List of the indexable attributes for the type.
locally_managed	Boolean	R	Indicates whether a given attribute is locally or globally managed if the repository is participating in a federation.
r_object_id	ID	S	The value at each index level corresponds to the attribute named in attr_names in the type's dm_type object. Object ID of the type info object
r_orig_declaration	Boolean	R	Indicates whether a given attribute is defined for the type or inherited. TRUE means it is defined for the type.
r_supertype	string(27)	R	The value at each index level corresponds to the attribute named in attr_names in the type's dm_type object. List of all supertypes of the type.

Attribute	Datatype	Single/ repeating	Description
r_type_id	ID	S	Object ID of the type object that contains the definition of the type.
r_type_name	string(27)	S	Name of the type represented by the type object identified in r_type_id.
type_override	ID	S	Object ID of the dm_aggr_domain object, if any, for the object type.

User

Purpose Contains information about a user in the repository.

Implementation

Supertype: Persistent Object

Subtypes: None

Internal name: dm_user

Object type tag: 11

A user object contains information about a user in the repository. The information includes the user's default permissions, electronic mail address, default folder, and inbox object identifier. You must have Sysadmin or Superuser user privileges to create or drop a user or to activate or deactivate a user.

[Table 2-177, page 488](#), lists the attributes defined for the type.

Table 2-177. Attributes defined for the user type

Attribute	Datatype	Single/ repeating	Description
acl_domain	string(32)	S	Identifies the domain (owner) of the ACL associated with the user. The value is either the user who created the ACL or, for system ACLs, the name of the repository owner.
acl_name	string(32)	S	The object name of the associated ACL.
alias_set_id	ID	S	Object ID of the alias set object representing the user-level default alias set.

Attribute	Datatype	Single/ repeating	Description
client_capability	integer	S	<p>Indicates what level of use is expected of the user. Valid values are:</p> <ul style="list-style-type: none"> • 0 or 1, for consumer • 2, for contributor • 4, for coordinator • 8, for system administrator <p>The default value is 0.</p>
deactivated_ip_addr	string(64)	S	Reserved for future use
deactivated_utc_time	Date	S	Reserved for future use
default_folder	string(200)	S	<p>Identifies the user's default folder.</p> <p>The default is Temp.</p>
description	string(255)	S	User-defined description of the user.
failed_auth_attempt	integer	S	<p>Number of unsuccessful authentication attempts on behalf of the user.</p> <p>With the exception of the installation owner, this is set to 0 when a user is created. The default value for the installation owner is -1.</p> <p>Setting this to -1 for any user disables the feature for that user.</p> <p>If the feature is enabled for a user, the value is reset to 0 whenever the user is authenticated successfully.</p>

Attribute	Datatype	Single/ repeating	Description
first_failed_auth_ utc_time	Date	S	Reserved for future use
globally_managed	Boolean	S	Indicates whether the user object is managed globally or locally. The default is FALSE, meaning that it is locally managed.
group_def_permit	integer	S	Requires at least Sysadmin privileges to change. Default group permit. This is used to assign object-level permission at the group level to any object the user creates if no permission at that level is explicitly assigned.
home_docbase	string(120)	S	The user's home repository.
last_login_utc_ time	Date	S	Reserved for future use
owner_def_permit	integer	S	Default owner permit. This is used to assign object-level permission at the owner level to any object the user creates if no permission at that level is explicitly assigned.
r_has_events	Boolean	S	Whether someone has registered the user for auditing.
r_is_group	Boolean	S	Indicates if the user represents a group or an individual user.
r_modify_date	date	S	Time and date of the last change to the user object.
r_object_id	ID	S	Object ID of the user.

Attribute	Datatype	Single/ repeating	Description
restricted_folder_ids	ID	R	Object IDs of the cabinets or folders which the user can access. The user can access these objects and their subfolders. If set, the user may access only these folders. This is a local attribute, not a global attribute.
user_address	string(80)	S	User's electronic mail address. This is a required attribute.
user_admin	string(32)	S	Reserved for future use
user_db_name	string(32)	S	User's user name in the underlying RDBMS. This must consist of ASCII characters.
user_delegation	string(32)	S	The name of a user to whom to delegate work items.
user_global_unique_id	string(255)	S	Reserved for future use
user_group_name	string(32)	S	The default group. This is used when the user creates an object as part of the determination of the object's default group.
user_initials	string(16)	S	Reserved for future use
user_ldap_dn	string(255)	S	User's Distinguished Name in LDAP.

Attribute	Datatype	Single/ repeating	Description
user_login_domain	string(255)	S	<p>Name of the Windows domain against which this user is authenticated.</p> <p>If the user is an LDAP user, the attribute stores the object name of the LDAP config object representing the LDAP directory against which the user is authenticated.</p> <p>This may be blank.</p>
user_login_name	string(80)	S	<p>Name used to authenticate the user. This attribute must be set.</p> <p>The combination of user_login_name and user_login_domain must be unique.</p>
user_name	string(32)	S	<p>Content Server user name. This can be an individual user or a group name. It must be unique among the user and group names in the repository.</p> <p>The name must consist of characters compatible with the server_os_codepage of the Content Server.</p> <p>This is a required attribute.</p>
user_os_domain	string(15)	S	<p>Windows domain name associated with user.</p>

Attribute	Datatype	Single/ repeating	Description
user_os_name	string(32)	S	<p>The name of the user's operating system account, if any.</p> <p>This must consist of ASCII characters if specified.</p>
user_password	string(256)	S	<p>Encrypted password for the user. This is set only if the user's user_source is set to "inline_password".</p> <p>When displayed, it displays as 16 asterisks (*).</p>
user_privileges	integer	S	<p>Identifies the user's user privileges. This value is the sum of the values corresponding to the user privileges defined for the user. For example, if a user has Create Type and Create Cabinet user privileges, the value is 3.</p> <p>The privileges and their values are:</p> <ul style="list-style-type: none"> None (0) Create Type (1) Create Cabinet (2) Create Group (4) Sysadmin (8) Superuser (16) <p>For more information about these levels, refer to Chapter 9, Users and Groups in the <i>Content Server Administrator's Guide</i>.</p>

Attribute	Datatype	Single/ repeating	Description
user_source	string(16)	S	<p>Indicates the source of the user's authentication. Valid values are:</p> <p>LDAP, meaning the user is authenticated through the LDAP directory server.</p> <p>unix only, meaning the user is authenticated by standard UNIX mechanism; the domain isn't used.</p> <p>domain only, meaning the user is authenticated against the Windows domain; the UNIX password file isn't used.</p> <p>unix first, meaning the user is authenticated first by the standard UNIX mechanism. If that fails, the user is authenticated against the Windows domain.</p> <p>domain first, meaning the user is authenticated first against the Windows domain. If that fails, the user is authenticated using the standard UNIX mechanism.</p> <p>inline password, meaning that the user is authenticated against the password stored in the user_password attribute.</p> <p><i>plugin_identifier</i>, meaning the user is authenticated using the plug-in identified by the identifier. The identifier for the plug-in provided with Content Server is dm_netegrity.</p>

Attribute	Datatype	Single/ repeating	Description
			<p>Note: The UNIX- and domain-related values are effective only if an auth config object exists in the repository.</p>
user_state	integer	S	<p>Indicates the user's activation state. Valid values are:</p> <p>0, indicating a user who can log in</p> <p>1, indicating a user who cannot log in</p> <p>2, meaning a user who is locked</p> <p>3, meaning a user who is locked and inactive</p>
user_web_page	string(255)	S	Reserved for future use
user_xprivileges	integer	S	<p>The user's extended user privileges. The value is the sum of the values corresponding to the extended user privileges defined for the user. For example, if a user has Config Audit and Purge Audit privileges, the value is 24.</p> <p>The privileges and their values are:</p> <p>8, Config Audit</p> <p>16, Purge Audit</p> <p>32, View Audit</p>

Attribute	Datatype	Single/ repeating	Description
workflow_ disabled	Boolean	S	Indicates the user's availability for work item assignment. Valid values are: 0, meaning available 1, meaning not available The default is 0 (available).
world_def_permit	integer	S	Default world permit. This is used to assign object-level permission at the world level to any object the user creates if no permission at that level is explicitly assigned.

Userdata Table

Purpose Records application-specific information about a user referenced by an archived message.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dm_userdata_table
 Object type tag: 00

A userdata table object records application-specific information about a user referenced in an archived message. Userdata table objects are created when an email message is archived. Userdata table objects are primarily used by personal and compliance archiving applications.

[Table 2-178, page 497](#), lists the attributes defined for the type.

Table 2-178. Attributes defined for the userdata table type

Attribute	Datatype	Single/ repeating	Description
message_id	string(24)	S	Message identifier recorded in the associated dm_mail_message.message_id attribute
user_data	string(2000)	S	Application-specific information about the user identified in user_tag.
user_tag	string(64)	S	Tag name assigned to the user by the application

Validation Module

Purpose Represents a business module that corresponds to a Docbasic expression defined in a func expr object.

Implementation

Supertype: Module
 Subtypes: None
 Internal name: dmc_validation_module
 Object type tag: 0b

A validation module represents the Java equivalent of the Docbasic expressions identified in func expr objects. Each validation module stores the Java equivalents for all Docbasic expressions defined for check constraints or for conditional value assistance for a particular object type. A validation module object is associated with the equivalent func expr objects by a relationship whose name is dmc_expr_to_module. The validation module is related to the dmc_jar object that contains the compiled code through a relationship named dmc_module_to_jar.

[Table 2-179, page 498](#), lists the attributes defined for the type.

Table 2-179. Attributes defined for the validation module type

Attribute	Datatype	Single/ repeating	Description
expr_code_ object_id	ID	S	Object ID of the dmi_expr_code object that contains the original Docbasic expressions file and P-code file for the expressions. (The expressions are compiled into one file and the P-code file generated from that file. Both the source file and the P-code file are stored as content of the expr code object.)

Attribute	Datatype	Single/ repeating	Description
implementation_ enabled	Boolean	S	<p>Whether Java evaluation is enabled for the expressions implemented by this module.</p> <p>T means that Java evaluation is enabled; F means that Java evaluation is not enabled.</p> <p>The value in this attribute takes precedence over the <code>expr_enabled</code> setting in the associated <code>dmc_validation_relation</code> object.</p>
referenced_type_ name	string(27)	R	<p>Name of the object type to which the expression is bound. Currently, only one value in this attribute is supported.</p>

Validation Relation

Purpose Relates a func expr object to a validation module object.

Implementation

Supertype: Relation
 Subtypes: None
 Internal name: dmc_validation_relation
 Object type tag: 37

A validation relation object relates a func expr object to the validation module that represents the Java equivalent of the Docbasic expression in the func expr object. The relation_name value for all validation relation objects is dmc_expr_to_module.

Table 2-180, page 500, lists the attributes defined for the type.

Table 2-180. Attributes defined for the validation relation type

Attribute	Datatype	Single/ repeating	Description
expr_enabled	Boolean	S	Whether Java evaluation is enabled for the expression represented by the related validation module. T means Java evaluation is enabled; F means that Java evaluation is not enabled. The setting in this attribute can be overridden by the setting of the implementation_enabled attribute in the dmc_validation_module object.
has_implementation	Boolean	S	Indicates whether actual Java code exists for the expression. This is set to F if the Docbasic expression was not successfully migrated to

Attribute	Datatype	Single/ repeating	Description
			Java code for the validation module.
			If this is F, then expr_enabled is F and cannot be reset to T.

Value Assist

Purpose Describes the value assistance provided through the data dictionary for an attribute.

Implementation

Supertype: Persistent Object
Subtypes: Value List, Value Query, Value Func
Internal name: dm_value_assist
Object type tag: 5a

A value assist object describes the value assistance provided through the data dictionary for an attribute. Value assist objects are created and managed by Content Server and cannot be created by users.

[Table 2-181, page 502](#), lists the attributes defined for the type.

Table 2-181. Attributes defined for the value assist type

Attribute	Datatype	Single/ repeating	Description
complete_list	Boolean	S	Indicates whether the values returned by value assistance are the only acceptable values for the attribute. TRUE means that they are the only acceptable values. FALSE means that other values are acceptable.
domain_type	integer	S	Indicates the datatype for the values returned by value assistance. Valid values are: 0, meaning Boolean 1, meaning Integer 2, meaning String 3, meaning ID 4, meaning Time/date 5, meaning Double
object_name	string(32)	S	Currently unused.

Attribute	Datatype	Single/ repeating	Description
parent_id	ID	S	Object ID of the aggr domain object to which the attribute belongs.
use_as_constraint	Boolean	S	Currently unused.
value_estimate	integer	S	The number of values expected to be returned by value assistance.

Value Func

Purpose Describes the procedural form of value assistance.

Implementation

Supertype: Value Assist
Subtypes: None
Internal name: dm_value_func
Object type tag: 5d

A value func object describes the procedural form of value assistance. Value func objects are created and managed by Content Server and cannot be created by users.

[Table 2-182, page 504](#) lists the attributes defined for the type.

Table 2-182. Attributes defined for the value func type

Attribute	Datatype	Single/ repeating	Description
func_expression	ID	S	Object ID of the func expr object representing the user function to call.
value_separator	string(1)	S	Defines the character used as a separator in the value list returned by the user function. The default is a comma (,).

Value List

Purpose Contains the valid values for the list form of value assistance.

Implementation

Supertype: Value Assist
 Subtypes: None
 Internal name: dm_value_list
 Object type tag: 5b

A value list object contains the valid values for the list form of value assistance. Value list objects are created and managed by Content Server and cannot be created by users.

[Table 2-183, page 505](#), lists the attribute defined for the type.

Table 2-183. Attributes defined for the value list type

Attribute	Datatype	Single/ repeating	Description
valid_values	string(255)	R	<p>List of string values representing literals of the appropriate data type. Each literal must be unique within the list.</p> <p>The list values are displayed to users in the order in which they appear in the attribute.</p>

Value Query

Purpose Contains information for the query form of value assistance.

Implementation

Supertype: Value Assist
 Subtypes: None
 Internal name: dm_value_query
 Object type tag: 5c

A value query object contains information for the query form of value assistance. Value query objects are created and managed by Content Server and cannot be created by users.

[Table 2-184, page 506](#) lists the attributes defined for the type.

Table 2-184. Attributes defined for the value query type

Attribute	Datatype	Single/ repeating	Description
allow_caching	Boolean	S	Indicates whether the queries defined for value assistance can be cached. TRUE means that the queries can be cached. FALSE means that the query should be rerun each time it is needed.
query_attribute	string(32)	S	The attribute from the query's selected values list that will provide the data to be displayed. The default is the first selected value.
query_string	string(255)	R	The DQL SELECT statement that returns the value assistance values. If it is longer than 255 characters, the first 255 characters appear in query_string[0], the second 255 characters in

Attribute	Datatype	Single/ repeating	Description
			query_string[1], and so forth. The continuation character is an underscore (_).

Vstamp

Purpose Used internally by the system, at start-up, to validate the consistency of the server and repository.

Implementation

Supertype: Persistent Object

Subtypes: None

Internal name: dmi_vstamp

Object type tag: 1e

A vstamp object is used internally by the system, at start-up, to validate the consistency of the server and repository.

[Table 2-185, page 508](#), lists the attributes for the object type.

Table 2-185. Attributes defined for the vstamp type

Attribute	Datatype	Single/ repeating	Description
i_application	string(64)	S	Name of the client program
i_stamp	integer	S	Value used internally to validate repository consistency

Webc Config

Purpose Describes the documents and attributes that are to be exported to a Site Caching Services repository.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_webc_config
 Object type tag: 08

A webc config object describes the documents and attributes that are to be exported to a Site Caching Services repository.

[Table 2-186, page 509](#), lists the attributes defined for the type.

Table 2-186. Attributes defined for the webc config type

Attribute	Datatype	Single/ repeating	Description
a_event_number	integer	S	Unique number generated internally for each publish operation
a_increment_cnt	integer	S	Number of successful incremental refreshes since the initial publication or the last full refresh
a_initial_publish	date	S	Date of the initial publication
a_last_increment	date	S	Date of the last successful incremental refresh
a_publish_status	string(128)	S	Status of the last operation
a_refresh_date	date	S	Date of the last successful full refresh

Attribute	Datatype	Single/ repeating	Description
content_meta_tags	Boolean	S	If TRUE , the system inserts the document attributes as metatags into the HTML files of the ContentDB. The default is FALSE.
effective_label	string(32)	S	Used to determine which documents are valid for publishing. The export operation examines the values in the a_effective_date and a_expiration_date attributes of all documents whose a_effective_label attribute value matches the webc config effective_label attribute. If unspecified, effective labels are not enforced.
export_directory	string(255)	S	Local directory where exported files are placed for pickup by the transfer agent. This is a required attribute.
export_properties	Boolean	S	If TRUE, the system creates the PropertiesDB on the target site. If FALSE, the PropertiesDB is not created. The default is FALSE.
is_active	Boolean	S	Indicates whether the system will publish documents described by this webc config object. TRUE (the default) directs the system to publish the documents; FALSE suspends publication.

Attribute	Datatype	Single/ repeating	Description
method_trace_level	integer	S	Controls the trace level for operations. Valid values are the same as those for the Trace API method. The default is 0.
notification_user	string(32)	S	Reserved for future use
object_name	string(255)	S	This attribute is inherited from dm_sysobject. However, because it is required for webc config objects, it is included in this table. For webc config objects, object_name is the logical name of the WebCache repository.
owner_name	string(32)	S	This attribute is inherited from dm_sysobject. For the webc config subtype, the user defined as the owner must have Superuser user privileges.
publish_arguments	string(255)	S	Specifies arguments to pass to the publishing method. Currently, there is one valid argument: -system_validate This is a Boolean argument that determines, in conjunction with the target_validate_required flag in agent.ini, how authentication is conducted. -system_validate is set to T (TRUE) by default. Refer to the WebCache documentation for details.

Attribute	Datatype	Single/ repeating	Description
send_notification	Boolean	S	Reserved for future use
source_attr_isrep	Boolean	R	Indicates whether the attribute named in the corresponding index position in source_attrs is a repeating attribute. TRUE means the attribute is repeating. FALSE means the attribute is single-valued.
source_attr_length	integer	R	Length of the attribute named in the corresponding index position in source_attrs.
source_attr_type	integer	R	Identifies the datatype of the attribute named in the corresponding index position in source_attrs. Valid values are: 0, for Boolean 1, for Integer 2, for String 3, for ID 4, for Date 5, for Double
source_attrs	string(80)	R	Identifies additional attributes to export to the PropertyDB. The specification syntax is: <i>type_name.attribute_name</i> If this attribute has no values, then only the default attributes are exported.

Attribute	Datatype	Single/ repeating	Description
source_folder_id	ID	S	Object ID of the local folder that is the root of the repository structure that matches the WebCache repository. This is a required attribute.
source_formats	string(64)	R	Document formats to publish. If formats are specified in this attribute, only documents in the specified formats are published. If a document has multiple formats (a primary format and renditions), only those that are included in the list are published. If this attribute is unspecified, all existing renditions, including the one in the primary format, are published by default.
source_version	string(32)	S	Version of the WebCache software running on the source. (Informational only)
target_id	ID	R	Object ID of the webc target object that references this webc config object. This is a required attribute. Note: This is a repeating attribute to allow for future enhancements.

Attribute	Datatype	Single/ repeating	Description
transfer_method	string(32)	S	Reserved for future use.
version_labels	string(32)	R	<p>Identifies the repository version you want to publish. Only documents carrying the specified version are published. Specify only one version label.</p> <p>If you set this to ANY VERSION, the system publishes whichever version is found in the source folder (the source folder is identified in source_folder_id). It is assumed that only one version of any particular document will be in the source folder at a given time.</p> <p>The default is CURRENT.</p> <p>Note: This is a repeating attribute to allow for future enhancements.</p>

Webc Target

Purpose Describes a Site Caching Services repository on a Site Caching Services target host.

Implementation

Supertype: SysObject
 Subtypes: None
 Internal name: dm_webc_target
 Object type tag: 08

A webc target object describes a Site Caching Services repository on a Site Caching Services target host.

[Table 2-187, page 515](#), lists the attributes defined for the type.

Table 2-187. Attributes defined for the webc target type

Attribute	Datatype	Single/ repeating	Description
app_server	string(32)	S	Name of the Web site delivery engine. (Informational only)
app_server_version	string(32)	S	Version level of the Web site delivery engine. (Informational only)
object_name	string(255)	S	This attribute is inherited from dm_sysobject. However, because it is required for webc target objects, it is included in this table. For webc target objects, object_name is the logical name of the WebCache repository.

Attribute	Datatype	Single/ repeating	Description
online_sync	Boolean	S	Indicates whether the data in the WebCache repository is accessible to users during the synchronization operation. If set to TRUE, the synchronization agent minimizes the possibility that users will be interrupted by repository updates. The default is FALSE.
online_sync_dir	string(255)	S	Remote directory used for the backup copy of the WebCache repository during online updates. This is required if online_sync is TRUE.
owner_name	string(32)	S	This attribute is inherited from dm_sysobject. For the webc target subtype, the user defined as the owner must have Superuser user privileges.
post_sync_script	string(48)	S	Name of a script to be executed after synchronization. The script must be stored in the WebCache bin directory on the target machine. Refer to the WebCache documentation for details.
pre_sync_script	string(48)	S	Name of a script to be executed prior to synchronization. The script must be stored in the WebCache bin directory on the target machine. Refer to the WebCache documentation for details.

Attribute	Datatype	Single/ repeating	Description
probdb_flags	string(64)	R	Reserved for future use.
propdb_tablename	string(64)	S	<p>Name to use when creating the PropertyDB tables. Three tables are created using the name in this attribute:</p> <p><i>propdb_tablename_s</i> <i>propdb_tablename_r</i> <i>probdb_tablename_m</i></p> <p>You must specify a name if the export_properties attribute in the webc config object is set to TRUE.</p>
propdb_dbversion	string(32)	S	Version level of the target RDBMS. (Informational only)
secure_connection	string(16)	S	<p>Identifies which agent port to use for the transfer. Valid values are:</p> <ul style="list-style-type: none"> • ssl, meaning use the target_ssl_port • raw, meaning use the target_raw_port • both, meaning use the target_ssl_port if available; if not, use the target_raw_port <p>Note: Documentum Administrator will not allow you to specify the both option; it must be set through the API. If you publish using the both option, you will receive warning messages in the log file.</p>
sync_arguments	string(128)	R	Reserved for future use

Attribute	Datatype	Single/ repeating	Description
target_host	string(128)	S	Host name or IP address of the WebCache host machine.
target_is_active	Boolean	S	Reserved for future use
target_raw_port	integer	S	Port number at which the target Agent process resides. This port is used for unencrypted transmissions. Either this attribute or target_ssl_port must be set.
target_root_directory	string(255)	S	The physical directory on the Web server where the files will be placed. This is a required attribute.
target_ssl_port	integer	S	Port number at which the target Agent process resides. This port is used for encrypted transmissions. Either this attribute or target_raw_port must be set.
target_version	string(32)	S	Version level of the WebCache software running on the host server. (Informational only)
target_virtual_dir	string(255)	S	The URL for the WebCache installation. This is a required attribute if you are using WebPublisher™.
transfer_arguments	string(128)	R	Reserved for future use

Attribute	Datatype	Single/ repeating	Description
transfer_directory	string(255)	S	Remote directory where exported files are placed for pickup by the Synchronization agent. This attribute is required if the transfer_method attribute in the webc config object is set.
transfer_domain	string(16)	S	For Windows NT, the domain of the user identified in transfer_user. For other platforms, this is unused.
transfer_protocol	string(16)	S	Reserved for future use
transfer_user	string(32)	S	User name to use to access the remote host in the transfer operation. The default is the local repository owner.

WF Attachment

Purpose Describes an attachment to a workflow.

Implementation

Supertype: Persistent Object

Subtypes: None

Internal name: dmi_wf_attachment

Object type tag: 00

A wf attachment object describes an object added at runtime to a workflow or workitem as an attachment.

[Table 2-188, page 520](#) lists the attributes defined for the type.

Table 2-188. Attributes defined for the WF attachment type

Attribute	Datatype	Single/ repeating	Description
r_component_id	ID	S	Object Id of the object attached to the workflow
r_component_name	string(80)	S	Name of the object attached to the workflow
r_component_type	string(40)	S	Object type (actual or supertype) of the object attached to the workflow
r_creation_date	Date	S	Date and time the attachment was attached to the workflow
r_creator_name	string(32)	S	Name of the user who attached the object to the workflow
r_workflow_id	ID	S	Object ID of the workflow to which the object is attached.

WF Package Schema

Purpose Stores the URI of a workflow package schema.

Implementation

Supertype: Relation
 Subtypes: None
 Internal name: dmc_wf_package_schema
 Object type tag: 37

A wf package schema object stores the URI of a schema associated with a package whose component is an XML file. It is used to validate any XPath expression referencing that file in a transition condition for an activity. WF package schema objects are created when the package is defined using Business Process Manager. The object type is installed by a script when Content Server is installed.

[Table 2-189, page 521](#), lists the attribute defined for the type.

Table 2-189. Attribute defined for the WF package schema type

Attribute	Datatype	Single/ repeating	Description
schema_uri	string(255)	S	URI of a schema

WF Package Skill

Purpose Associates a skill level required for a workflow package with a workflow.

Implementation

Supertype: Relation
 Subtypes: None
 Internal name: dmc_wf_package_skill
 Object type tag: 37

A wf package skill object identifies the skill level a user must have to acquire or be assigned the task associated with a particular package. This feature is effective only when applied to tasks on workqueues. WF package skill objects are created at runtime and associated with the workflow.



Caution: Do not modify this object type nor instances of the object. These objects are created, maintained, and destroyed internally, as needed.

[Table 2-190, page 522](#), lists the attributes defined for the type and the inherited attributes whose settings are particular to instances of this type.

Table 2-190. Attribute defined for the wf package skill type

Attribute	Datatype	Single/ repeating	Description
child_id	ID	S	This is not set.
child_label	string(32)	S	There is no label set for instances of this type.
description	string(250)	S	There is no description provided for instances of this type.
package_name	string(16)	S	Name of the package that requires the skill level identified in skill_level.
parent_id	ID	S	Object ID of the workflow with which the package is associated
permanent_link	Boolean	S	Set to T (TRUE).

Attribute	Datatype	Single/ repeating	Description
relation_name	string(32)	S	Set to dmc_wf_package_skill
skill_level	integer	S	An integer value identifying the required skill level. Refer to the BPM documentation for information about the skill levels.

WF Timer

Purpose Records the configuration of a timer for a workflow activity.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dmi_wf_timer
 Object type tag: 00

A wf timer object describes a timer for a workflow activity. The wf timer objects are created automatically when the timer is instantiated. The object type is installed by a script when Content Server is installed.

Table 2-191, page 524, lists the attributes defined for the type.

Table 2-191. Attributes defined for the WF timer type

Attribute	Datatype	Single/ repeating	Description
r_act_id	ID	S	Object ID of the activity for which this timer is defined
r_act_name	string(32)	S	Name of the activity for which this timer is defined
r_act_seqno	integer	S	Sequence number of the activity instance
r_action_id	ID	S	Object ID of the module config object associated with the timer
r_action_index	integer	S	The index to the r_pre_timer_action or r_post_timer_action attribute of the activity
r_timer	Date	S	Absolute date and time at which to trigger the timer

Attribute	Datatype	Single/ repeating	Description
r_timer_type	integer	S	Identifies what kind of timer this is. Valid values are: 0, meaning a pre-timer 1, meaning a post-timer 2, meaning a suspend-timer
r_workflow_id	ID	S	Object ID of the workflow that contains the activity instance.

Workflow

Purpose Contains the runtime information about a workflow.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dm_workflow
 Object type tag: 4d

A workflow object contains the runtime information about a workflow.

[Table 2-192, page 526](#), describes the attributes defined for the workflow type.

Table 2-192. Attributes defined for the workflow type

Attribute	Datatype	Single/ repeating	Description
i_next_seqno	integer	S	Indicates the next sequence number of an activity instance.
i_performer_flag	integer	R	Indicates special run-time conditions such as extension.
instructions	string(255)	R	User-defined string displayed to users who are using the sendToDistributed workflow. Applications can reference this attribute. It must be set before the first Save operation on the workflow object.
object_name	string(32)	S	Contains the workflow name.

Attribute	Datatype	Single/ repeating	Description
process_id	ID	S	Contains the object ID of process definition (the dm_process object) on which this workflow is based.
r_act_def_id	ID	R	Contains the object ID of the activity definitions (dm_activity objects) included in the process definition identified in process_id.
r_act_errorno	integer	R	Records the failing operation, if any. Used for error recovery.
r_act_name	string(32)	R	Contains the activity identifier as defined in the dm_process object.
r_act_seqno	integer	R	Contains the unique sequence number of an activity instance.
r_act_state	integer	R	Records the activity's current state. Valid values and their corresponding states are: 0, meaning dormant 1, meaning active 2, meaning finished 3, meaning halted 4, meaning failed
r_alias_set_id	ID	S	Records the object ID of the alias set used to resolve performer aliases when the workflow is created. This is a runtime copy of the alias set identified in perf_alias_set_id of the dm_process object.

Attribute	Datatype	Single/ repeating	Description
r_complete_witem	integer	R	Records the number of completed work items.
r_creator_name	string(32)	S	Indicates the creator. Automatically set by the server.
r_last_performer	string(32)	R	Contains the activity performer who last marks his or her own work item as complete. Before any work item of an activity instance completes, this may contain the last performer of the previous activities.
r_last_witem_id	ID	R	Records the ID of the work item that was most recently marked as complete.
r_perf_act_name	string(32)	R	Contains the names of activities whose performers were chosen at workflow initiation or upon completion of another activity. An activity name appears once for each performer of the activity. The activity's performer appears at the corresponding index position in r_performers.
r_performers	string(32)	R	Contains the name of a user or group chosen at workflow initiation or upon completion of another activity as a performer for the activity at the corresponding index position in r_perf_act_name. If there are multiple performers for an activity, each performer appears at a

Attribute	Datatype	Single/ repeating	Description
			different index position, and r_perf_act_name records the name of the activity in the corresponding index positions.
r_post_timer	Date	R	This attribute is obsolete in version 5.3. If the workflow and activity were started prior to upgrading to 5.3, this value is the absolute date and time when the activity should finish.
r_pre_timer	Date	R	This attribute is obsolete in version 5.3. If the workflow was started prior to upgrading to 5.3, this value is the absolute date and time when the activity should start.
r_repeate_invoke	Boolean	R	Indicates whether this activity can be triggered multiple times.
r_runtime_state	integer	S	Indicates the current state of the workflow. Values are: 0, meaning dormant 1, meaning running 2, meaning finished 3, meaning halted 4, meaning terminated
r_start_date	datetime	S	Represents the start time of the instance, set by the server when the workflow enters the running state.
r_total_witem	integer	R	Records the total number of generated work items.
r_trigger_revert	integer	R	Records the number of revert ports triggered (either 0 or 1).

Attribute	Datatype	Single/ repeating	Description
r_trigger_input	integer	R	Records the number of input ports triggered.
r_trigger_thresh	integer	R	Indicates the triggering threshold.
supervisor_name	string(32)	S	Indicates the workflow supervisor. The default is the creator.

Work Item

Purpose Stores information about a task for a human or automatic performer.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dmi_workitem
 Object type tag: 4a

Work items are generated by Content Server from an activity object. Users cannot create work items. Users can modify only the following attributes: a_held_by, a_wq_doc_profile, a_wq_flag, a_wq_name, and a_wq_policy_id.

[Table 2-193, page 531](#) lists the attributes for the work item type.

Table 2-193. Attributes defined for the work item type

Attribute	Datatype	Single/ repeating	Description
a_held_by	string(32)	S	Name of the user who has acquired the task. This is only set if the workflow tasks are managed using the work queue feature available through BPM.
a_wq_doc_profile	string(64)	S	Name of the work queue doc profile associated with this task. Note: This is only set if the workflow tasks are managed using the work queue feature available through BPM.

Attribute	Datatype	Single/ repeating	Description
a_wq_flag	integer	S	<p>Indicates whether the task is pushed to a queue member or pulled by a queue member. Valid values are:</p> <p>0, meaning the task is pulled by a queue member</p> <p>1, meaning the task is pushed to a queue member</p> <p>The default is 1.</p> <p>Note: This is only set if the workflow tasks are managed using the work queue feature available through BPM.</p>
a_wq_name	string(32)	S	<p>Name of the work queue to which this work item is assigned.</p> <p>Note: This is only set if the workflow tasks are managed using the work queue feature available through BPM.</p>
a_wq_policy_id	ID	S	<p>Object ID of the work queue policy that controls how the work item is handled on the queue.</p> <p>Note: This is only set if the workflow tasks are managed using the work queue feature available through BPM.</p>
r_act_def_id	ID	S	<p>Refers to the activity definition object.</p>

Attribute	Datatype	Single/ repeating	Description
r_act_seqno	integer	S	Records the sequence number of the activity in which the package is being handled.
r_auto_method_id	ID	S	Contains the object ID of an application to be invoked (a dm_method instance).
r_creation_date	datetime	S	Records date and time of when a work item is generated.
r_due_date	datetime	S	Indicates the date and time when a work item is expected to complete.
r_exec_launch	Boolean	S	Indicates whether the work item is currently executing. TRUE means the work item is executing; FALSE means it is not executing.
r_exec_os_error	string	S	Contains the operating system error string, if any.
r_exec_result_id	ID	S	Contains the document ID of the saved results of the application execution.
r_exec_timed_out	Boolean	S	Indicates if an execution times out.
r_ext_performer	string(32)	R	Lists new performers to repeat the same activity. If a group name, only one work item is generated and any group member can acquire the work item.

Attribute	Datatype	Single/ repeating	Description
r_launch_timeout	Date	S	Records the date and time at which a work item execution times out. This is set when the work item begins execution if the work item is generated by an automatic activity. Otherwise, its value is NULLDATE.
r_output_port	string(16)	R	Allows a performer to specify a set of output ports by their unique names.
r_performer_name	string(32)	S	Contains the name of an activity performer, or the owner name of the method if this is an automatic work item.
r_priority	integer	S	Represents the priority assigned by a performer or an application. The default value comes from the activity_priority attribute of dm_process of the activity by which the work item is generated.
r_queue_item_id	ID	S	Refers to a peer dmi_queue_item object.
r_runtime_state	integer	S	Contains the current state of the work item. Valid values are: 0, meaning dormant 1, meaning acquired 2, meaning finished 3, meaning paused 4, meaning Dpaused (a work item in the dormant state is paused) 5, meaning Apaused (a work item in the acquired state is paused)

Attribute	Datatype	Single/ repeating	Description
			6, meaning Ppaused (a work item in the paused state is paused)
			Values 4-6 occur when a workflow containing the work item is halted.
r_workflow_id	ID	S	Contains the object ID of the workflow that generated this item.
return_value	integer	S	Contains the returned value set by a performer or an application.
user_cost	double	S	Actual user cost spent to complete the work item.
			The default value is 0.
user_time	integer	S	Actual amount of time spent by the user to complete the work item.
			The default is 0.

Work Queue

Purpose Represents a work queue for work items generated from a workflow.

Implementation

Supertype: Persistent Object
Subtypes: None
Internal name: dmc_workqueue
Object type tag: 00

A work queue object represents a work queue for workflow tasks. The workqueue object type is installed by a script when Content Server is installed. However, work queues (instances of the type) are created and managed using Webtop.

[Table 2-194, page 536](#), lists the attributes defined for the type.

Table 2-194. Attributes defined for the work queue type

Attribute	Datatype	Single/ repeating	Description
wq_category_id	ID	S	Object ID of the workqueue category object representing the category to which this work queue belongs.
wq_name	string(32)	S	Name of the work queue.
wq_policy_id	ID	S	Object ID of the workqueue policy used to handle work items in this queue.

Work Queue Category

Purpose Defines a category for a work queue.

Implementation

Supertype: Folder
Subtypes: None
Internal name: dmc_workqueue_category
Object type tag: 0b

A work queue category object represents a category of work queues. The object type has no attributes defined for it. It inherits all its attributes from its supertype, dm_folder. It uses only the object_name, acl_name, and acl_domain attributes. The object_name attribute stores the work queue category name. The acl_name and acl_domain attributes are set to specify the ACL named "Work Queue User Default ACL" for all work queue category objects.

The object type is installed by a script when Content Server is installed. However, instances of the type are created and managed through Webtop when work queues are set up and managed.

Work Queue Doc Profile

Purpose Stores information about a particular kind of document.

Implementation

Supertype: Persistent Object

Subtypes: None

Internal name: dmc_workqueue_doc_profile

Object type tag: 00

A work queue doc profile contains information that describes a particular kind of document. The objects are used to manage work items placed on work queues. Work queues, and consequently, work queue doc profiles, are created using Webtop. The object type is installed using a script at the time Content Server is installed.

[Table 2-195, page 538](#), lists the attributes defined for the type.

Table 2-195. Attributes defined for the work queue doc profile type

Attribute	Datatype	Single/ repeating	Description
doc_profile_name	string(64)	S	Name of the doc profile.
owner_name	string(32)	S	Name of the user who owns the work queue doc profile object.
wq_name	string(32)	R	Names of the work queues to which documents that use this profile may be assigned.
wq_policy_id	ID	R	Object IDs of work queue policy objects. The work queue policy object identified at a particular index position is associated with the work queue specified at the corresponding index position in wq_name.

Work Queue Policy

Purpose Defines configuration information for a work queue.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dmc_workqueue_policy
 Object type tag: 00

A work queue policy object defines configuration parameters for handling a task in a workqueue. The parameters control how the items are handled. Each work queue has one associated work queue policy. If a document associated with a task has a defined work queue policy, that policy overrides the work queue's policy.

The work queue policy object type is installed by a script when Content Server is installed. You must have installed Business Process Manager to use work queues and their associated work queue policies.

[Table 2-196, page 539](#), lists the attributes defined for the type.

Table 2-196. Attributes defined for the work queue policy type

Attribute	Datatype	Single/ repeating	Description
increment_priority	integer	S	Value by which the priority of a task in the queue is incremented.
initial_priority	integer	S	Initial priority value of tasks in the work queue.
max_priority	integer	S	Maximum allowed priority value for an unfinished task on the queue.
max_threshold	ID	S	Maximum number of unfinished tasks allowed in the queue.
owner_name	string(32)	S	Name of the user who owns this work queue policy object.

Attribute	Datatype	Single/ repeating	Description
percent_quality_ check	integer	S	Value, interpreted as percentage, that determines whether the quality assurance is performed on the task when it is completed.
policy_name	string(255)	S	Name of the work queue policy.
policy_type	integer	S	Identifies whether the policy defines the policy of a work queue or a document. Valid values are: 0, meaning it is a work queue policy 1, meaning it is a document policy

Work Queue User Profile

Purpose Records information about a work queue user.

Implementation

Supertype: Persistent Object
 Subtypes: None
 Internal name: dmc_workqueue_user_profile
 Object type tag: 00

A work queue user profile describes a user who performs tasks taken from a work queue. The information includes the user's skill level. The user profiles are created when the work queue is created.

The object type is installed by a script when Content Server is installed. You must have installed Business Process Manager to use work queues and the associated user profiles.

[Table 2-197, page 541](#), lists the attributes defined for the type.

Table 2-197. Attributes defined for the work queue user profile type

Attribute	Datatype	Single/ repeating	Description
doc_profile_name	string(64)	R	Names of doc profile objects
owner_name	string(32)	S	Name of the owner of this workqueue user profile object
skill_level	integer	R	Defines skill levels for the user. The level identified at a particular index position applies to the workqueue and doc profile identified in the corresponding index positions in workqueue_name and doc_profile_name.

Attribute	Datatype	Single/ repeating	Description
user_name	string(32)	S	Name of the user
workqueue_name	string(32)	R	Names of the work queues

XFM Form

Purpose Contains information about a form template.

Implementation

Supertype: Document
 Subtypes: None
 Internal name: dm_xfm_form
 Object type tag: 09

An xfm form object stores information about a form template. The object type is installed when the DocApp provided with EMC Documentum Forms Builder is installed. Instances of the type are created internally when EMC Documentum Forms Builder is used to create a form template.

Table 2–198, page 543, lists the attributes defined for the type.

Table 2-198. Attributes defined for the xfm form type

Attribute	Datatype	Single/repeating	Description
definition_state	integer	S	Indicates the status of the form. Valid values are: 0, meaning draft 1, meaning validated 2, meaning installed 3, meaning obsolete
description	string(300)	S	User-defined description of the form

Attribute	Datatype	Single/repeating	Description
display_mode	integer	S	Indicates the display mode of the form. Valid values are: 0, meaning dialog 1, meaning publish
relationship_to_base	integer	S	Identifies the relationship between the current form and the base form. Valid values are: 0, meaning the form is not related to the base form 1, meaning the form is related to the base form

XFM Instance

Purpose Represents an instance of a form.

Implementation

Supertype: Document
 Subtypes: None
 Internal name: dm_xfm_instance
 Object type tag: 09

A xfm instance object contains information about one form. The object type is installed when the DocApp provided with EMC Documentum Forms Builder is installed. Instances of the type are created internally when EMC Documentum Forms Builder is used to create a form template.

[Table 2-199, page 545](#), lists the attribute defined for the type.

Table 2-199. Attributes defined for the xfm instance type

Attribute	Datatype	Single/repeating	Description
is_default	Boolean	S	Indicates the form is a form default instance. T (TRUE) means the form is a default instance. F (FALSE) means the form is not a default instance.

XFM Schema

Purpose Stores information about an XML schema for a form.

Implementation

Supertype: dm_document
 Subtypes: None
 Internal name: dm_xfm_schema
 Object type tag: 09

An xfm schema object represents an XML schema used with a form. The object type is installed when the DocApp provided with EMC Documentum Forms Builder is installed. Instances of the type are created internally when EMC Documentum Forms Builder is used to create a form.

Table 2–200, page 546, lists the attributes defined for the type.

Table 2-200. Attributes defined for the xfm schema type

Attribute	Datatype	Single/repeating	Description
data_model	string(128)	S	Identifies the root element of the schema.
storage_option	integer	S	Specifies the storage mode for the schema. Valid values are: 0, meaning store as content 1, meaning store as content and use an XML Application to populate object attributes with the content.
storage_type	string(32)	S	Name of the object type of the object with which the content of the form instance is associated

XML Application

Purpose Stores the information that defines a particular kind of XML document.

Implementation

Supertype: Folder
 Subtypes: None
 Internal name: dm_xml_application
 Object type tag: 0b

The information in an xml application object includes the document type definition (DTD) used by the document, the name space identified in the document's header, and the document's root element.

[Table 2-201, page 547](#) lists the attributes defined for the type .

Table 2-201. Attributes defined for the xml application type

Attribute	Datatype	Single/ repeating	Description
dtd_public_id	string(255)	S	The public ID of the document type definition identified in dtd_system_id.
dtd_system_id	string(255)	S	The system ID of a document type definition.
namespace	string(80)	S	The name space identified in the XML document's header.
root_elements	string(32)	R	Identifies the type of XML document.
root_object_types	string(32)	R	Names of the object types used by the XML application. These are the formal names of the object types; for example, dm_document. Custom object types may be included.

Attribute	Datatype	Single/ repeating	Description
			EMC Documentum Desktop uses this attribute to determine which object types to display in user interface fields for operations on objects handled by an XML application.

XML Config

Purpose Stores a document type definition file (DTD) as content.

Implementation

Supertype: Document
Subtypes: None
Internal name: dm_xml_config
Object type tag: 09

An xml config object stores, as content, a document type definition file (DTD), the XML file that describes the rules for chunking an XML document in an XML application.

[Table 2-202, page 549](#), lists the attribute defined for the XML config object type.

Table 2-202. Attributes defined for the xml config type

Attribute	Datatype	Single/ repeating	Description
config_locator	ID	S	Used internally to manage XML documents.

XML Custom Code

Purpose Stores the Java code that implements a custom Java interface as content.

Implementation

Supertype: Document
 Subtypes: None
 Internal name: dm_xml_custom_code
 Object type tag: 09

An xml custom code object stores, as content, the Java code that implements a custom Java interface.

[Table 2-203, page 550](#) lists the attributes defined for the type.

Table 2-203. Attributes defined for the xml custom code type

Attribute	Datatype	Single/ repeating	Description
class_path	string(255)	S	Java class path of the code.
code_type	integer	S	Identifies the Java interface implemented by the code. Valid values are: 1, meaning code is a DfDTDHandler subclass 2, meaning code is a DfDTDDocumentHandler subclass 3, meaning Code implements IDfLinkDetector
com_class_id	string(80)	S	Currently unused

XML Style Sheet

Purpose Stores an XSL file as content.

Implementation

Supertype: Document
Subtypes: None
Internal name: dm_xml_style_sheet
Object type tag: 09

An xml style sheet object stores an XSL file as content. An XSL file is a style sheet that defines how to format an XML document on output.

[Table 2-204, page 551](#) lists the attributes defined for the type.

Table 2-204. Attributes defined for the xml style sheet type

Attribute	Datatype	Single/ repeating	Description
transformed _format	string(80)	S	Defines the output file format of the XML document.

XML Zone

Purpose Contains the information used to populate the Zone tab on a Find dialog.

Implementation

Supertype: Document
Subtypes: None
Internal name: dm_xml_zone
Object type tag: 09

An xml zone object contains the information used to populate the Zone tab on a Find dialog.

[Table 2–205, page 552](#) lists the attributes defined for the type.

Table 2-205. Attributes defined for the xml zone type

Attribute	Datatype	Single/ repeating	Description
description	string(32)	S	User defined.
zone_name	string(32)	R	Identifies one or more zones that can be searched in a fulltext index search. A zone is the name of an XML tag.

The Documentum Views

Documentum provides you with three views to make it easier to obtain information about the items in your inbox and about routers and router tasks. These views are registered tables in the system so that you can join them to a type in a DQL query if you wish. The views are:

- [The dm_queue view, page 553](#)
- [The dm_tasks_queued view, page 556](#)
- [The dm_tasks_dequeued view, page 559](#)

This appendix contains tables that describe each view.

The dm_queue view

The dm_queue view is a view on the dmi_queue_item object type. This view shows only those items that are currently in an inbox. The server manages all items in inboxes as items of type dmi_queue_item.

Table A-1. The dm_queue view

Attribute Name	Datatype	Description
stamp	ID	Object ID of the dmi_queue_item that represents the queued item.
stamp_i	integer	Integer equivalent of the final 8 characters in the stamp value.
name	string(32)	Name of the user to whom this event was sent.
sent_by	string(32)	Name of the user that sent this event.
date_sent	date	Date that the event was sent.
event	string(10)	The system- or user-defined event.

Attribute Name	Datatype	Description
item_name	string(255)	For a non-workflow event, the object name of the object that generated the event. For a workflow event, the object name of the process or workflow object that generated the event.
item_id	ID	Object ID of the queued item.
item_id_i	integer	Integer equivalent of the final 8 characters of the item_id.
item_type	string(32)	Object type of the object identified by the item_id.
content_type	string(32)	File format, if defined, of the object's content.
due_date	date	User-defined date that indicates when the task is due for completion.
message	string(255)	Message sent to the user receiving the event from the user who initiated the event.
router_id	ID	Object ID of the queued router.
router_id_i	integer	Integer equivalent of the final 8 characters of the router_id value.
supervisor_name	string(32)	Name of the router's supervisor.
task_number	string(5)	Number of the task if the queued item is a router task.
task_name	string(32)	Name of the task if the queued item is a router task. Otherwise, this attribute contains the word events.
task_type	string(10)	Type of the task if the queued item is a router task. This is one of: begin, step, or end.
task_state	string(10)	State of the task if the queued item is a router task. One of: dormant, ready, acquired, paused, or finished.

Attribute Name	Datatype	Description
dependency_type	string(10)	For queued router tasks, the value in the task's dependency_type attribute. One of: <ul style="list-style-type: none"> • or and none For workflow tasks, this records the circumstances of the work item's creation.
next_tasks_type	string(10)	Obsolete. No longer used.
instruction_page	integer	Page number of the router's content file that contains instructions for this task (if the item is a router task).
plan_start_date	date	Obsolete. No longer used.
actual_start_date	date	Obsolete. No longer used.
read_flag	smallint	0 (FALSE) if the event has not been read, or 1 (TRUE) if the event has been read.
priority	integer	Integer value representing an application-interpreted priority value for the task.
sign_off_required	BOOLEAN	T or F, indicating whether the task's user is required to sign off before forwarding the router.
sign_off_user	char(32)	Name of the user who signed off the task.
sign_off_date	date	Date that the task was signed off.
a_content_type	char(32)	File format in which to display the contents of the object attached to the task. (Note that this may be different than the content type specified for the object.)
a_operations	char(16)	Contains the value of the operations attribute from the task's router.
delete_flag	BOOLEAN	Indicates whether the task has been dequeued. This is 0 (FALSE) if the task is still queued or 1 (TRUE) if the task has been dequeued.
position	integer	Used internally

Attribute Name	Datatype	Description
dequeued_by	char(32)	Documentum user name of the user who dequeued the task, either explicitly or implicitly.
dequeued_date	date	Date and time when the task was dequeued.

The dm_tasks_queued view

The dm_tasks_queued view is a view on the dmi_queue_item type.. There is a row in this view for every queued task in the system.

Table A-2. The dm_tasks_queued view

Attribute	Datatype	Description
stamp	ID	The object ID of the dmi_queue_item representing the queued object.
stamp_i	integer	The integer equivalent of the final 8 characters of the object ID found in the stamp attribute.
name	string(32)	The name of the user to whom the queued item was sent. This is the user's Documentum user name.
sent_by	string(32)	The name of the user that sent the queued item. This is the user's Documentum user name.
date_sent	date	The date that the item was sent.
event	string(10)	A system- or user-defined event.
item_name	string(32)	For a workflow task, the object name of the peer work item. For an event, the object name of the object that generated the event. For a router task, the object name of the queued item associated with the task.

Attribute	Datatype	Description
item_type	string(10)	For a workflow task, the object type of the peer work item. For an event, the object type of the object that generated the event. For a router task, the object type of the queued item associated with the task.
content_type	string(10)	The file format defined for the object, if any.
message	string(255)	Message from the task's sender to the person receiving the queued item.
task_number	string(5)	For router tasks, the number of the task.
task_name	string(10)	For router tasks, the name of the task.
read_flag	smallint	0 (FALSE) if the task has not been read or 1 (TRUE) if it has been read.
delete_flag	smallint	0 (FALSE) if the task not been dequeued or 1 (TRUE) if it has been dequeued.
priority	integer	User- or application-interpreted value that defines the item's priority.
task_owner	string(32)	The name of the user (individual or group) to whom this task is assigned.
router_id	ID	The object ID of the router that contains this task.
router_id_i	integer	The integer equivalent of the final 8 characters of the value in the router_id attribute.
task_state	string(10)	The status of the task. One of: dormant, ready, acquired, paused, or finished.
task_type	string(5)	The type of the task. One of: begin, step, or end.
next_tasks_type	string(10)	Indicates which of the tasks appearing in the next_tasks_list are queued when the current task is completed and the router forwarded. One of: all, none, or decision (user must specify at run time).

Attribute	Datatype	Description
next_tasks_list	string(64)	Contains a list of all tasks to which the router is forwarded when the current task is completed.
plan_start_date	date	Planned starting date of the task. This is set by the router's supervisor.
actual_start_date	date	The actual starting date of the task. This is set when the task is acquired.
late_action	string(32)	not currently used
instruction_page	integer	The page number of the content file associated with the router that holds instructions for the task.
task_package_id	ID	The object ID of the object associated with this task.
task_package_id_i	integer	The integer value of the final 8 characters of the value in the task_package_id attribute.
task_package_label	string(32)	Contains a symbolic label defined for one of the versions in the version tree that contains the object identified in the task_package_id.
supervisor_name	string(32)	The name of the router's supervisor.
router_state	string(10)	The status of the router. One of: dormant, active, halted, or finished.
package_id	ID	The object ID of the object associated with the router.
package_id_i	integer	The integer equivalent of the final 8 characters of the value in package_id.
package_label	string(32)	Contains a symbolic label defined for one of the versions in the version tree that contains the object identified in the package_id.
sign_off_user	char(32)	Name of the user who signed off the task.
sign_off_date	date	Date that the task was signed off.
a_content_type	char(32)	File format in which to display the contents of the object attached to the task. (Note that this may be different than the content type specified for the object.)
a_operations	char(16)	Contains the value of the operations attribute from the task's router.

The dm_tasks_dequeued view

The dm_tasks_dequeued view is view on the queue item type.

Table A-3. The dm_tasks_dequeued view

Attribute	Datatype	Description
stamp	ID	Object ID of the queue item object for the dequeued item.
stamp_i	integer	Integer equivalent of the final 8 characters in the stamp value.
user_name	char(32)	Name of the user who owns the task.
sent_by	char(32)	Name of the user who queued the item.
date_sent	date	Date that the item was sent.
event	char(10)	The system- or user-defined event.
item_name	char(255)	Name of the dequeued item.
message	char(255)	Message that was sent to the task's owner by the sender of the task.
task_number	integer	For router tasks, the number of the task within the router.
task_name	char(32)	For router tasks, the name of the task.
item_type	char(32)	Object type of the dequeued item.
content_type	char(32)	File format defined for the dequeued item's content.
due_date	date	Date that the task was to be completed.
read_flag	BOOLEAN	Whether the item has been looked at by the owner.
delete_flag	BOOLEAN	Whether the item has been removed from the queue.
priority	integer	User-defined priority for the item.
task_owner	char(32)	Name of the user who owns the item.
router_id	ID	Object ID of the dequeued router.
router_id_i	integer	Integer equivalent of the last 8 characters of the router_id value.
task_state	char(10)	State of the task.

Attribute	Datatype	Description
task_type	char(10)	Type of the task. One of begin, step, or end.
next_tasks_type	char(10)	Identifies which tasks in the next_tasks_list are to be queued after the current task is completed. One of: all, none, or decision.
next_tasks_list	char(64)	Lists, by task number, those tasks that are to be queued after the current task completes.
plan_start_date	date	Planned starting date for the task.
actual_start_date	date	Actual starting date for the task.
late_action	char(32)	Not currently used.
actual_end_date	date	Actual date that task was completed.
instruction_page	integer	Page number of the content file associated with the router that contains instructions for the task.
task_package_id	ID	Identifies the object that is associated with the task.
task_package_id_i	integer	The integer equivalent of the final 8 characters in the task_package_id value.
task_package_label	char(32)	A symbolic label defined for one of the versions in the version tree that contains the object identified by the task_package_id value.
supervisor_name	char(32)	Name of the router's supervisor.
router_state	char(10)	State of the router that contains the task.
router_name	char(32)	Name of the router that contains the task.
package_id	ID	Object ID of the object associated with the router.
package_id_i	integer	Integer equivalent of the final 8 characters in the package_id value.
package_label	char(32)	A symbolic version label defined for one of the versions in the version tree that contains the object specified by the package_id attribute.
dequeued_by	char(32)	Name of the user who dequeued the item.
dequeued_date	date	Date that the item was dequeued.
sign_off_user	char(32)	Name of the user who signed off the task.

Attribute	Datatype	Description
sign_off_date	date	Date that the task was signed off.
a_content_type	char(32)	File format in which to display the contents of the object attached to the task. (Note that this may be different than the content type specified for the object.)
a_operations	char(16)	Contains the value of the operations attribute from the task's router.

Language and Country Codes

This appendix lists the recommended language codes and some of the recommended country codes for use in the `language_code` attribute of `SysObjects`. The language codes are taken from ISO 639. The country codes are taken from the larger set found in ISO 3116.

Table B-1. Recommended language codes

Language	Code	Language	Code
Albanian	sq	Arabic	ar
Armenian	hy	Azerbaijani	az
Basque	eu	Bulgarian	bg
Byelorussian	be	Cambodian	km
Catalan	ca	Chinese	zh
Croatian	hr	Czech	cs
Danish	da	Dutch	nl
English	en	Estonian	et
Finnish	fi	French	fr
Georgian	ka	German	de
Greek	el	Hebrew	he
Hungarian	hu	Icelandic	is
Indonesian	id	Italian	it
Japanese	ja	Kazakh	kk
Korean	ko	Latvian, Lettish	lv
Lithuanian	lt	Macedonian	mk
Malay	ms	Moldavian	mo
Norwegian	no	Persian	fa

Language	Code	Language	Code
Polish	pl	Portuguese	pt
Romanian	ro	Russian	ru
Serbian	sr	Serbo-Croatian	sh
Slovak	sk	Slovenian	sl
Spanish	es	Swedish	sw
Thai	th	Turkish	tr
Ukrainian	uk	Uzbek	uz
Vietnamese	vi		

Table B-2. Recommended country codes

Country	Code	Country	Code
Argentina	AR	Austria	AT
Australia	AU	Azerbaijan	AZ
Belarus	BY	Belgium	BE
Belize	BZ	Bolivia	BO
Brazil	BR	Bulgaria	BG
Cambodia	KH	Cameroon	CM
Canada	DA	Chile	CL
China	CN	Colombia	CO
Costa Rica	CR	Croatia	HR
Czech Republic	CZ	Denmark	DK
Ecuador	EC	Egypt	EG
Estonia	EE	Finland	FI
France	FR	Gabon	GA
Gambia	GM	Georgia	GE
Germany	DE	Great Britain (UK)	GB
Greece	GR	Greenland	GL
Guatemala	GT	Haiti	HT
Honduras	HN	Hong Kong	HK
Hungary	HU	Iceland	IS
India	IN	Indonesia	ID

Country	Code	Country	Code
Iran	IR	Iraq	IQ
Ireland	IE	Israel	IL
Italy	IT	Japan	JP
Jordan	JO	Kazachstan	KZ
Kenya	KE	Korea (North)	KP
Korea (South)	KR	Kyrgyz Republic	KG
Laos	LA	Latvia	LV
Lebanon	LB	Libya	LY
Liechtenstein	LI	Lithuania	LT
Luxembourg	LU	Mexico	MX
Mozambique	MZ	Morocco	MA
Netherlands	NL	New Zealand	NZ
Nicaragua	NI	Norway	NO
Pakistan	PA	Panama	PK

RDBMS Tables for Documentum Types

All the objects in Content Server, and their attribute values, are stored in a repository. This repository is part of a larger database, the relational database (RDBMS). This appendix contains an explanation and an example of the RDBMS tables that describe a Documentum type.

Note: The example tables assume that the underlying RDBMS is Oracle for the datatype descriptions.

Overview

Within the relational database, each object type is represented by two tables and two views:

- A table and a view for the object's single-valued attributes, and
- A table and a view for the object's repeating attributes.

The tables

The name of the table describing an object's single-valued attributes has the format:

typename_s

For example, the table that contains the single-valued attributes for the document type is the `dm_document_s` table.

The names of the tables describing the repeating attributes have the following format:

typename_r

For example, the table that contains the repeating attributes of the document type is the `dm_document_r` table.

The views

The views join the a type's tables with the tables of its supertypes. For example, the view of the single-valued attributes for folders joins the dm_folder_s and dm_sysobject_s tables. For cabinets, the view joins the dm_cabinet_s, dm_folder_s, and dm_sysobject_s tables.

The names of the views that are built on the single-valued attribute tables have the following format:

typename_sp

For example, the single-valued attribute view associated for dm_document type is named dm_document_sp.

Returning database table information

To obtain a listing of a table's columns and their datatypes in an Oracle database, you use the SQL Describe statement. For example, the following statement returns a description of the table that stores the SysObject type's single-valued attributes:

```
sql> describe dm_sysobject_s;
```

The user type tables in the Oracle RDBMS

This section contains examples of the _s and _r tables, with the dm_user type as the demonstrative type. The views are not shown as they do not differ for this type, as it has no supertypes.

[Table C-1, page 568](#) lists the results of the following statement, which retrieves the columns of the dm_user_s table:

```
sql>describe dm_user_s
```

Table C-1. Columns of the dm_user_s table

Column Name	Datatype	Nullable (Yes/No)
R_OBJECT_ID	CHAR(16)	N
GROUP_NAME	CHAR(32)	N
GROUP_ADDRESS	CHAR(80)	N
I_VSTAMP	NUMBER(10)	N

Column Name	Datatype	Nullable (Yes/No)
I_ATTR_EXTRA2	CHAR(64)	Y
I_ATTR_INT1	NUMBER(10)	Y
I_ATTR_INT2	NUMBER(10)	Y
I_VSTAMP	NUMBER(10)	Y

Table C-2, page 569 shows the results of the following statement, which retrieves a description of the dm_user_r table:

```
sql>describe dm_user_r;
```

Table C-2. Columns of the dm_user_r table

Column Name	Datatype	Nullable (Yes/No)
R_OBJECT_ID	NUMBER(10)	N
I_POSITION	NUMBER(6)	N
USER_NAMES	CHAR(32)	Y
GROUPS_NAMES	CHAR(32)	Y
I_ALL_USERS_NAMES	CHAR(16)	Y

@ (at sign) object ID format, 30
, (comma) in method commands, 29
_ (underbar)
 in table and column names, 32
_ (underscore), in type names, 31

A

a_ prefix, 15
_accessor_app_permit (computed attribute), 17
_accessor_name (computed attribute), 17
_accessor_permit (computed attribute), 17
_accessor_permit_type (computed attribute), 18
_accessor_xpermit (computed attribute), 18
_accessor_xpermit_names (computed attribute), 18
ACL object type, 42
_acl_ref_valid (computed attribute), 18
ACLs
 aliases in, 66
 dm_audittrail_acl type, 94
 dmi_audittrail_attrs type, 97
acs config object type, 47
activity object type, 52
agent exec process
 dm_agent_exec utility, 270
 method name in server config object, 426
aggr domain object type, 64
alias set object type, 66
_alias_set (computed attribute), 18
_alias_sets (computed attribute), 25
_all_users_names (computed attribute), 18
_allow_change_location (computed attribute), 19
_allow_change_owner (computed attribute), 19

_allow_change_permit (computed attribute), 19
_allow_change_state (computed attribute), 19
_allow_execute_proc (computed attribute), 19
annotations
 note object type, 323
API (Application Programming Interface)
 functions, 29
 methods, 28
 syntax, 29
api config object type, 68
app ref object type, 80
application object type, 81
aspect relation object type, 84
aspect type object type, 85
assembly object type, 86
attachments
 wf attachment object type, 520
_attribute_list_values (computed attribute), 19
attributes
 a_ prefix, 15
 aggr_domain object type, 64
 _attribute_list_values (computed attr), 19
 computed, 16
 computed, for lifecycles, 25
 datatypes, 16, 27
 dd attr info object type, 156
 dd common info object type, 162
 dm_display_config type, 178
 domain object type, 200
 domains, 16
 global, 15
 global/local determination, 16
 i_ prefix, 15
 index value, 14
 local, 16
 naming rules, 31

- persistent, 14
 - r_prefix, 15
 - read and write, 15
 - read-only, 15
 - referencing, 28
 - repeating, 14
 - single-valued, 14
 - storage, 14
 - audit trail acl object type, 94
 - audit trail attrs object type, 97
 - audit trail entries
 - _sign_data computed attribute, 23
 - audit trail group object type, 98
 - audit trail object type, 88
 - audit trail signatures
 - _sign_data computed attribute, 23
 - auditing
 - audit trail objects, 88
 - auth config object type, 101
- B**
- backing up
 - dumping repositories, 226
 - blob storage areas
 - object type, 102
 - renditions in, 436
 - blobstore object type, 102
 - BOF modules
 - module object type, 311
 - builtin expr object type, 103
- C**
- CA store object type, 104
 - cabinet object type, 107
 - cabinets
 - privileges to change, 107
 - cache config object type, 108
 - _cached (computed attribute), 19
 - caches
 - client_cache_size attribute, 428
 - dump process, 226
 - query, 446
 - size described in server config object, 426
 - size in session config object, 442
 - case sensitivity
 - table and column names, 32
 - type and attribute names, 31
 - user and group names, 32
 - category assign object type, 113
 - category class object type, 116
 - category object type, 110
 - change record object type, 120
 - _changed (computed attribute), 19
 - child security for relationships, 36
 - CI config object type, 122
 - client applications
 - date formats, 446
 - client cache
 - maximum size, defining, 428
 - client caches
 - cache config object type, 108
 - client_cache_size attribute, 428
 - collection object identifiers, 32
 - columns
 - naming rules, 32
 - commas in method commands, 29
 - comment object type, 125
 - completed workflow object type, 127
 - completed workitem object type, 131
 - component object type, 136
 - component routine, 348
 - _componentID (computed attribute), 20
 - composite predicate object type, 137
 - computed attributes
 - _accessor_app_permit, 17
 - _accessor_permit_type, 18
 - _is_restricted_session, 22
 - _istransactionopen, 22
 - lifecycles, 25
 - list of, 16
 - referencing, 28
 - _sign_data, 23
 - cond expr object type, 138
 - cond ID expr object type, 139
 - configuration objects
 - api config, 68
 - connection config, 140
 - docbase config, 181
 - location, 294
 - mount point, 316
 - server config, 426
 - session config, 441
 - connection brokers
 - docbase locator object type, 191
 - docbroker locator object type, 194
 - server locator object type, 437
 - connection config objects, 140

- `_containID` (computed attribute), 20
 - containment object type, 143
 - content assignment policies
 - `dm_relation_ssa_policy` type, 373
 - `dm_ssa_policy` type, 452
 - content object type, 146
 - Content Server
 - server config object type, 426
 - server locator object type, 437
 - content-addressed storage
 - `dm_ca_store` type, 104
 - `_content_buffer` (computed attribute), 20
 - `_content_state` (computed attribute), 20
 - `copy_child` attribute, 36
 - `_count` (computed attribute), 20
 - country codes, 563
 - `_current_state` (computed attribute), 21
- D**
- data dictionary
 - aggr domain object type, 64
 - dd attr info object type, 156
 - dd common info object type, 162
 - dd info object type, 168
 - dd type info object type, 175
 - domain object type, 200
 - expr code object type, 234
 - nls dd info object type, 320
 - value assist object type, 502
 - value func object type, 504
 - value list object type, 505
 - value query object type, 506
 - datatypes
 - attribute, 16
 - attributes, 24, 27
 - default values, 27
 - dates
 - format for clients, 446
 - dd attr info object type, 156
 - dd common info object type, 162
 - dd info object type, 168
 - dd type info object type, 175
 - default values
 - datatypes, 27
 - procedure timeout period, 308
 - repository scope, 30
 - `direction_kind` attribute, 38
 - discussions
 - `dmc_comment` type, 125
 - `dmc_readcomment` type, 360
 - `dmc_topic` type, 478
 - display config type, 178
 - distributed repositories
 - local and global attributes, 15
 - distributed store object type, 180
 - `dm_prefix`, 31
 - `dm_acl` type, 42
 - `dm_acs_config` type, 47
 - `dm_activity` type, 52
 - `dm_agent_exec` utility, 270
 - `dm_aggr_domain` type, 64
 - `dm_alias_set` type, 66
 - `dm_app_ref` type, 80
 - `dm_application` type, 81
 - `dm_assembly` type, 86
 - `dm_audittrail` type, 88
 - `dm_audittrail_acl` type, 94
 - `dm_audittrail_group` type, 98
 - `dm_auth_config` type, 101
 - `dm_blob_store` type, 102
 - `dm_builtin_expr` type, 103
 - `dm_ca_store` type, 104
 - `dm_cabinet` type, 107
 - `dm_cache_config` type, 108
 - `dm_category` type, 110
 - `dm_category_assign` type, 113
 - `dm_category_class` type, 116
 - `dm_ci_config` type, 122
 - `dm_component` type, 136
 - `dm_cond_expr` type, 138
 - `dm_cond_id_expr` type, 139
 - `dm_dd_info` type, 168
 - `dm_display_config` type, 178
 - `dm_distributedstore` type, 180
 - `dm_docbase_config` type, 181
 - `dm_docset` type, 195
 - `dm_docset_run` type, 197
 - `dm_document` type, 199
 - `dm_domain` type, 200
 - `dm_dump_record` type, 226
 - `dm_email_address_table` type, 229
 - `dm_email_message` type, 231
 - `dm_esign_template` type, 232
 - `dm_expression` type, 235
 - `dm_extern_file` type, 237
 - `dm_extern_free` type, 238
 - `dm_extern_store` type, 239
 - `dm_extern_url` type, 241
 - `dm_federation` type, 242

- dm_filestore type, 245
- dm_folder type, 246
- dm_foreign_key type, 247
- dm_format type, 248
- dm_ftengine_config type, 256
- dm_ftindex_agent_config type, 252
- dm_fulltext_index type, 257
- dm_func_expr type, 259
- dm_group type, 262
- dm_job type, 270
- dm_job_request type, 275
- dm_job_sequence type, 277
- dm_key type, 279
- dm_ldap_config type, 280
- dm_linkedstore type, 286
- dm_literal_expr type, 289
- dm_load_record type, 292
- dm_location type, 294
- dm_locator type, 296
- dm_mail_message type, 297
- dm_media_profile type, 304
- dm_method type, 305
- dm_mount_point type, 316
- dm_network_location_map type, 318
- dm_nls_dd_info type, 320
- dm_note type, 323
- dm_outputdevice type, 326
- dm_plugin type, 330
- dm_policy type, 332
- dm_procedure type, 341
- dm_process type, 342
- dm_public_key_certificate, 346
- dm_qual_comp type, 348
- dm_query type, 351
- dm_queue (view), 553
- dm_reference type, 361
- dm_registered type, 366
- dm_relation type, 371
- dm_relation_ssa_policy type, 373
- dm_relation_type type, 374
- dm_retainer type, 380
- dm_route_table type, 393
- dm_scope_config type, 423
- dm_script type, 424
- dm_server_config type, 426
- dm_smart_list type, 449
- dm_ssa_policy type, 452
- dm_state_extension type, 450
- dm_state_type type, 451
- dm_store type, 453
- dm_sysobject type, 461
- dm_tasks_dequeued (view), 559
- dm_tasks_queued (view), 556
- dm_taxonomy type, 475
- dm_type type, 483
- dm_user type, 488
- dm_userdata_table type, 497
- dm_value_assist type, 502
- dm_value_func type, 504
- dm_value_list type, 505
- dm_value_query type, 506
- dm_webc_config object type, 509
- dm_webc_target type, 515
- dm_WfReporting job
 - completed workflow object type, 127
- dm_workflow type, 526
- dm_xfm_form type, 543
- dm_xfm_instance, 545
- dm_xml_application type, 547
- dm_xml_config type, 549
- dm_xml_custom_code type, 550
- dm_xml_style_sheet type, 551
- dm_xml_zone type, 552
- dmAPIExec (API function), 29
- dmAPIGet (API function), 29
- dmAPISet (API function), 29
- dmc dsm application object type, 202
- dmc dsm backbone object type, 203
- dmc dsm doc propertie subject type, 204
- dmc dsm drug product object type, 206
- dmc dsm drug substance object typte, 207
- dmc dsm excipient object type, 208
- dmc dsm facilities equip object type, 209
- dmc dsm indication object type, 210
- dmc dsm m1 backbone object type, 211
- dmc dsm safety eval object type, 212
- dmc dsm sect doc attributes object type, 213
- dmc dsm section extension object type, 217
- dmc dsm section object type, 215
- dmc dsm sft section object type, 220
- dmc dsm stf backbone object type, 218
- dmc dsm study attributes, 221
- dmc dsm study report object type, 223
- dmc dsm submission object type, 224
- dmc rps authority object type, 395
- dmc rps execution rule object type, 409
- dmc rps notification object type, 412
- dmc_aspect_relation type, 84

- dmc_aspect_type type, 85
- dmc_comment type, 125
- dmc_completed_workflow type, 127
- dmc_completed_workitem type, 131
- dmc_composite_predicate type, 137
- dmc_dsm_application type, 202
- dmc_dsm_backbone type, 203
- dmc_dsm_doc_properties type, 204
- dmc_dsm_drug_product type, 206
- dmc_dsm_drug_substance type, 207
- dmc_dsm_excipient type, 208
- dmc_dsm_facilities_equip, 209
- dmc_dsm_indication type, 210
- dmc_dsm_m1_backbone type, 211
- dmc_dsm_sect_doc_attributes type, 213
- dmc_dsm_section type, 215
- dmc_dsm_section_extension type, 217
- dmc_dsm_stf_backbone type, 218
- dmc_dsm_stf_section type, 220
- dmc_dsm_study_attributes, 221
- dmc_dsm_study_report type, 223
- dmc_dsm_submission type, 224
- dmc_jar type, 268
- dmc_java_library type, 269
- dmc_module type, 311
- dmc_module_config type, 314
- dmc_notepage type, 324
- dmc_reaadcomment type, 360
- dmc_richtext type, 385
- dmc_room type, 386
- dmc_routecase_condition type, 389
- dmc_rps_authority type, 395, 397
- dmc_rps_base_date type, 399
- dmc_rps_child_strategy type, 401
- dmc_rps_condition type, 403
- dmc_rps_contact type, 404
- dmc_rps_disposition_method type, 405
- dmc_rps_event type, 407
- dmc_rps_execution_rule type, 409
- dmc_rps_hold type, 410
- dmc_rps_notification type, 412
- dmc_rps_phase_rel, 414
- dmc_rps_retainer type, 416
- dmc_rps_retainer_event_rel type, 419
- dmc_rps_retention_policy type, 420
- dmc_tcf_activity type, 476
- dmc_tcf_activity_template type, 477
- dmc_topic type, 478
- dmc_transition_condition type, 479
- dmc_validation_module type, 498
- dmc_validation_relation type, 500
- dmc_wf_package_schema type, 521
- dmc_wf_package_skill type, 522
- dmc_workqueue type, 536
- dmc_workqueue_category type, 537
- dmc_workqueue_doc_profile type, 538
- dmc_workqueue_policy type, 539
- dmc_workqueue_user_profile type, 541
- DMCL (client library)
 - api config objects and, 68
 - Docbroker locator object, 194
 - server locator object, 437
 - version number in api config object, 77
- dmcl.ini file
 - max_connection_per_session key, 33
 - max_session_count key, 33
- dmi_audittrail_attrs type, 97
- dmi_change_record type, 120
- dmi_dd_attr_info type, 156
- dmi_dd_common_info type, 162
- dmi_dd_type_info type, 175
- dmi_dump_object_record type, 225
- dmi_expr_code type, 234
- dmi_index object type, 266
- dmi_linkrecord type, 287
- dmi_load_object_record type, 290
- dmi_otherfile type, 325
- dmi_package type, 327
- dmi_queue_item type, 352
- dmi_registry type, 368
- dmi_replica_record type, 378
- dmi_sequence type, 425
- dmi_session type, 440
- dmi_subcontent type, 459
- dmi_type_info type, 485
- dmi_vstamp type, 508
- dmi_wf_attachment type, 520
- dmi_wf_timer type, 524
- dmi_workitem type, 531
- dmr_containment type, 143
- dmr_content type, 146
- DO_METHOD procedures
 - object type for, 305
- docbase config object type, 181
- docbase locator object type, 191
- _docbase_id (computed attribute), 21
- docbase_scope attribute, 30
- Docbasic
 - method object and, 307

- plugins and, 331
- procedure objects and, 341
- docbroker locator object type, 194
- docset object type, 195
- docset run object type, 197
- document object type, 199
- Document Set, *see* docset object type
- documents
 - language and country codes, 563
- Documentum Collaborative Services
 - room object type, 386
- Documentum Resource Locator, object
 - type for, 296
- domain controllers
 - identifying, 101
- domain object type, 200
- domain-required mode, 183
- domains, 16, 200
- DQL (Document Query Language)
 - names, 31
 - naming rules, 31
 - query objects and, 351
 - referencing attributes, 28
 - reserved words in queries, 28
- dsm_dmc_safety_eval type, 212
- _dump (computed attribute), 21
- dump object record object, 225
- dump record object type, 226

E

- electronic mail address
 - group, 262
 - user, 491
- electronic signatures
 - Esign template object type, 232
- email message object type, 231
- email message table object type, 229
- EMC Documentum Forms Builder
 - xfm form object type, 543
 - xfm instance object type, 545
- _entry_criteria (computed attribute), 26
- error level in _status attribute, 24
- Esign template object type, 232
- events
 - registry object type, 368
 - SysObject attributes, 473
- explicit transactions
 - _isdeadlocked, 22

- _istransactionopen (computed
 - attribute), 22
- expr code object type, 234
- expression object type, 235
- extended permissions, computed
 - attributes for, 25
- external applications
 - app ref object type, 80
 - application object type, 81
 - methods, referencing, 29
 - qual comp object type, 348
- external file store object type, 237
- external free store object type, 238
- external procedures
 - method object type and, 305
- external storage
 - external free store object type, 238
 - external store object type, 239
 - external URL object type, 241
 - plug-in libraries, 330
- external store object type, 239
- external URL object type, 241

F

- federation object type, 242
- file store object type, 245
- folder object type, 246
- foreign key object type, 247
- format object type, 248
- ft index agent config object type, 252
- ftengine config object type, 256
- full-text indexes
 - fulltext index object type, 257
- fulltext index object type, 257
- fulltext indexes
 - ftengine config object type, 256
- func expr object type, 259

G

- global attributes, 15
- group object type, 262
- groups
 - aliases, 66
 - _all_users_names (computed attr), 18
 - dm_audittrail_group type, 98
 - group object type, 262
 - naming rules, 32

H

- `_has_config_audit` (computed attribute), 21
- `_has_create_cabinet` (computed attribute), 21
- `_has_create_group` (computed attribute), 21
- `_has_create_type` (computed attribute), 21
- `_has_purge_audit` (computed attribute), 21
- `_has_superuser` (computed attribute), 21
- `_has_sysadmin` (computed attribute), 21
- `_has_view_audit` (computed attribute), 21

I

- `i_prefix`, 15
- `i_is_replica` attribute, 14, 41
- `i_vstamp` attribute, 13, 41
- `_id` (computed attribute), 21
- identifiers
 - collection object, 32
 - names, 31
 - object, 32
 - session, 33
 - subconnection, 33
 - type, 33
- inboxes
 - `dm_queue` (view), 553
 - `dm_tasks_queued` (view), 556
- `_included_types` (computed attribute), 26
- index agents
 - ft index agent config object type, 252
- index object type, 266
- indirect references, 30
- `integrity_kind` attribute, 38
- `_is_restricted_session` (computed attribute), 22
- `_isdeadlocked` (computed attribute), 22
- `_isnew` (computed attribute), 22
- `_isreplica` (computed attribute), 22
- `_istransactionopen` (computed attribute), 22

J

- jar files
 - jar object type, 268
- jar object type, 268
- Java

- java library object type, 269
- java library object type, 269
- job object type, 270
- job request object type, 275
- job sequence object type, 277
- jobs
 - job object type, 270
 - job sequence object type, 277

K

- key object type, 279
- keys
 - foreign key object type, 247
 - key object type, 279

L

- language codes, 563
- ldap config object type, 280
- LDAP directory server
 - ldap config object type, 280
- `_lengths` (computed attribute), 22
- lifecycle states
 - attachability, 332
- lifecycles
 - computed attributes, 25
 - object type for, 332
 - `_policy_name` (computed attr), 23
 - state extension object type, 450
 - state type object type, 451
- link record object type, 287
- linked storage areas
 - link record object type, 287
 - linked store object type, 286
- linked store object type, 286
- links
 - reference object type, 361
- literal expr object type, 289
- load object record objects, 290
- load record object type, 292
- local attributes, 16
- `locally_managed` attribute, 16
- location object type, 294
- location objects
 - events directory, 141
 - external file stores, 237
 - file stores, 245
 - linked stores, 286
- locator object type, 296

locking
 SysObject attribute for, 472
 login ticket key
 ticket_crypto_key attribute, 185
 login tickets
 login_ticket_cutoff attribute, 185
 trust_by_default attribute, 189
 trusted_docbases attribute, 189
 login_ticket_cutoff attribute, 185
 logs
 API tracing, 77

M

mail message object type, 297
 _masterdocbase (computed attribute), 22
 max_connection_per_session (dmcl.ini file key), 33
 max_session_count (dmcl.ini file key), 33
 maximums
 cache size in server config object, 426
 characters in table and column names, 32
 characters in type and attribute names, 31
 characters in user and group names, 32
 sequence number, 425
 media profile object type, 304
 method object type, 305
 methods
 attributes, referencing, 28
 repository scope
 default, 30
 described, 29
 scoping arguments, 29
 syntax, general, 29
 minimum values
 login tickets, issuance date, 185
 module config object type, 314
 module object type, 311
 mount point object type, 316

N

names
 attribute, 28, 31
 column, 32
 described, 31
 DQL reserved words as, 31

group, 32
 repository, 186
 rules for, 31
 server config objects, 431
 table, 32
 type, 31
 user, 32
 _names (computed attribute), 22
 network location map object type, 318
 _next_state (computed attribute), 26
 nls dd info object type, 320
 non-persistent types
 api config, 68
 connection config, 140
 docbase locator, 191
 docbroker locator, 194
 server locator, 437
 session config, 441
 none security for relationships, 36
 note object type, 323
 notepage object type, 324
 notes
 note object type, 323
 notification in registry object type, 368

O

object identifiers, 13, 32
 object IDs
 indirect reference, 30
 sequence numbers, 425
 object types
 dd type info, 175
 dm_type type, 483
 dmi_type_info type, 485
 naming rules, 31
 persistent object type, 13
 relation, 34
 relation type, 34
 objects
 _cached (computed attr), 19
 _current_state (computed attr), 21
 persistent attributes, 14
 _policy_name (computed attr), 23
 relation type objects, 39
 relationships between, 34
 other file object type, 325
 output device object type, 326

P

package object type, 327
 packages
 dmi_package type, 327
 skill levels for, 522
 wf package schema object type, 521
 parent security for relationships, 36
 permanent_link attribute, 36
 _permit (computed attribute), 23
 persistent attributes
 datatypes, 16
 defined, 14
 domains, 16
 local and global, 15
 read and write, 15
 read-only, 15
 single-valued, 14
 persistent object type, 13, 41
 plug-in libraries
 plugin object type, 330
 plugin object type, 330
 policy object type, 332
 _policy_name (computed attribute), 23
 _previous_state (computed attribute), 26
 primary session identifiers, 33
 private
 activities, 54
 cabinets, 107
 groups, 265
 mount point objects, 317
 process definitions, 343
 procedure object type, 341
 procedures
 timeout period, 308
 process object type, 342
 public
 activities, 54
 cabinets, 107
 groups, 265
 mount point objects, 317
 objects, 471
 process definitions, 343
 public key certificate object type, 346

Q

qual comp object type, 348
 queries
 caching, 446
 query object type, 351

queue item object type, 352
 quotes
 enclosing reserved words in, 31

R

r_prefix, 15
 r_object_id attribute, 13, 41
 RDBMS
 datatypes, 27
 Documentum tables in, 567
 index storage areas, 185
 reserved words, 32
 user name, 491
 views, 568
 readcomment object type, 360
 reference object type, 361
 registered object type, 366
 registered tables
 dm_registered type, 366
 registry object type, 368
 relation object type, 371
 relation objects
 described, 34
 relation SSA policy object type, 373
 relation type object type, 374
 relation type objects
 creating, 34
 described, 34
 destroying, 39
 relationships
 child security level, 36
 copy behavior, release
 compatibility, 37
 copy_child attribute, 36
 delete behavior, defining, 38
 described, 34
 direction_kind attribute, 38
 integrity_kind attribute, 38
 none security level, 36
 parent security level, 36
 permanent_link attribute, 36
 relation object, 34
 relation type objects, 34
 relationship types, creating, 34
 security, 35
 system security level, 35
 user-defined, 34
 remote repositories
 i_is_replica attribute, 41

- renditions
 - storage of, 436
 - `_repeating` (computed attribute), 23
 - repeating attributes
 - index position, 14
 - number of in `_values` attribute, 25
 - storing values, 14
 - replica record object type, 378
 - repositories
 - docbase config object type, 181
 - names, restrictions, 186
 - RDBMS and, 567
 - reference object type, 361
 - `ticket_crypto_key` attribute, 185
 - `trust_by_default` attribute, 189
 - `trusted_docbases` attribute, 189
 - underlying RDBMS table
 - descriptions, 567
 - repository scope
 - default, 30
 - described, 29
 - subconnection identifiers and, 33
 - reserved words
 - as attribute names, 28
 - in queries, 28, 31
 - names and DQL, 31
 - RDBMS, 31
 - `_resume_state` (computed attribute), 23
 - retainer object type, 380
 - retention policies
 - `dm_retainer` type, 380
 - richtext content
 - `dmc_notepage` type, 324
 - richtext object type, 385
 - room object type, 386
 - route cases
 - `dmc_composite_predicate` type, 137
 - `dmc_transition_condition` type, 479
 - route table object type, 393
 - routecase condition object type, 389
 - routines, component, 348
 - rps authority object type, 397
 - rps base date object type, 399
 - rps child strategy object type, 401
 - rps condition object type, 403
 - rps contact object type, 404
 - rps disposition method object type, 405
 - rps event object type, 407
 - rps hold object type, 410
 - rps phase rel object type, 414
 - rps retainer event rel object type, 419
 - rps retainer object type, 416
 - rps retention policy object type, 420
- ## S
- scope
 - method, 29
 - repository, 29
 - scope config object type, 423
 - script object type, 424
 - security
 - ACL object type, 42
 - attributes controlling, 470
 - directory in mount point object, 317
 - directory or file in Location type
 - object, 295
 - folder, 184
 - level for repository, 188
 - public key certificate object type, 346
 - public objects, 471
 - relationships, 35
 - sequence object type, 425
 - server config object type, 426
 - server config objects
 - names, restrictions, 431
 - server locator object type, 437
 - session config object type, 441
 - session identifiers, 33
 - session object type, 440
 - sessions
 - api config type and, 68
 - connection config objects and, 140
 - identifiers, 33
 - subconnection identifiers, 33
 - `_sign_data` (computed attribute), 23
 - Smart List object type, 449
 - state extension object type, 450
 - state type object type, 451
 - `_state_extension_obj` (computed attribute), 26
 - `_state_type` (computed attribute), 26
 - states
 - computed attributes describing, 26
 - `_status` (computed attribute), 24
 - storage areas, 238 to 239, 452
 - See also* content assignment policies blob, 102
 - distributed store object type, 180
 - external URL object type, 241

- file store object type, 245
- linked store object type, 286
- store object type, 453
- store object type, 453
- subconnections
 - identifiers, 33
- subcontent object type, 459
- syntax
 - method, 29
- SysObject object type, 461
- SysObjects
 - _current_state (computed attr), 21
 - event-related attribute, 473
 - folder attributes, 464
 - general attributes, 461
 - lifecycle-related attributes, 474
 - security-related attributes, 470
 - version-related attributes, 472
 - virtual document attributes, 465
- system security for relationships, 35

T

- tables
 - naming rules, 32
- tablespaces
 - index location, 266
 - type index storage, 185
- taxonomy object type, 475
- TCF Activity object type, 476
- TCF activity template type, 477
- timeouts
 - automatic activities, 54
 - backup connection broker, 69
 - client network requests, 72
 - connection broker client
 - connection, 77
 - procedure, 308
- topic object type, 478
- tracing
 - API, 77
 - error level in _status attribute, 24
- transactions
 - _isdeadlocked, 22
- transformation profiles
 - media profile object type, 304
- transition condition object type, 479
- translation relationships
 - language and country codes, 563
- trust_by_default attribute, 189

- trusted_docbases attribute, 189
- turbo storage
 - renditions in, 436
 - subcontent object type, 459
- type identifiers, 33
- type indexes
 - index object type, 266
- type info object type, 485
- type object type, 483
- _type_id (computed attribute), 24
- _type_name (computed attribute), 24
- types
 - naming rules, 31
 - persistent attributes, 14
 - RDBMS table descriptions of, 567
 - _types (computed attribute), 24
 - _typestring (computed attribute), 25

U

- underbar (_)
 - in table and column names, 32
- underscore (_), in type names, 31
- user authentication
 - UNIX users against domains, 101
- user object type, 488
- user privileges
 - user object, 493
- user-defined types
 - identifiers, 33
 - naming rules, 31
- userdata table object type, 497
- users
 - aliases, 66
 - naming rules, 32
 - workqueue user profile object
 - type, 541

V

- validation module object type, 498
- validation relation object type, 500
- value assist object type, 502
- value func object type, 504
- value list object type, 505
- value query object type, 506
- _values (computed attribute), 25
- versions
 - SysObject attributes, 472
- views

- dm_queue, 553
- dm_tasks_dequeued, 559
- dm_tasks_queued view, 556
- Documentum, 568
- RDBMS, 567
- system-defined, 553
- virtual documents
 - assembly objects and, 86
 - SysObject attributes, 465
- vstamp object type, 508

W

- webc config object type, 509
- webc target object type, 515
- wf attachment object type, 520
- wf package schema object type, 521
- WF package skill object type, 522
- wf timer object type, 524
- work item object type, 531
- work items
 - completed workitem object type, 131
 - workqueue object type, 536
- work queues
 - workqueue doc profile object type, 538
 - workqueue object type, 536
 - workqueue policy object type, 539
 - workqueue user profile object type, 541
- workflow definitions
 - wf timer object type, 524
- workflow object type, 526

- workflows
 - completed workflows object type, 127
 - completed workitem object type, 131
 - module config object type, 314
 - process object type, 342
 - wf attachment object type, 520
 - wf package schema object type, 521
 - work item object type, 531
 - workqueue object type, 536
- workqueue category object type, 537
- workqueue doc profile object type, 538
- workqueue object type, 536
- workqueue policy object type, 539
- workqueue user profile object type, 541

X

- xfm form object type, 543
- xfm instance object type, 545
- XML (Extensible Markup Language)
 - xml application object type, 547
 - xml config object type, 549
 - xml custom code object type, 550
 - xml style sheet object type, 551
 - xml zone object type, 552
- xml application object type, 547
- xml config object type, 549
- xml custom code object type, 550
- xml style sheet object type, 551
- xml zone object type, 552
- _xpermit (computed attribute), 25
- _xpermit_list (computed attribute), 25
- _xpermit_names (computed attribute), 25